

EY-2278E-MM-0001

# LISTINGS BOOK

Prepared by Educational Services  
of  
Digital Equipment Corporation

Copyright © 1984 by Digital Equipment Corporation  
All Rights Reserved

The reproduction of this material, in part or whole, is strictly prohibited. For copy information, contact the Educational Services Department, Digital Equipment Corporation, Bedford, Massachusetts 01730.

Printed in U.S.A.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may not be used or copied except in accordance with the terms of such license.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

**The manuscript for this book was created using DIGITAL Standard Runoff. Book production was done by Educational Services Development and Publishing in Nashua, NH.**

The following are trademarks of Digital Equipment Corporation:

<b>digital</b> ™	DECtape	Rainbow
DATATRIEVE	DECUS	RSTS
DEC	DECwriter	RSX
DECmate	DIBOL	UNIBUS
DECnet	MASSBUS	VAX
DECset	PDP	VMS
DECsystem-10	P/OS	VT
DECSYSTEM-20	Professional	Work Processor

## APPENDIX

### Supplemental Listings

The following listings are provided for your reference. Some are executive source modules. Others are solutions to Module Test problems.

#### Executive Source Modules

1. CRDRIVER - Card Reader Driver
2. SYSQIOREQ - \$QIO System Service Procedure
3. SYSQIOFDT - System \$QIO FDT Routines
4. IOSUBNPAG - Nonpaged I/O Subroutines
5. FORKCNTRL - Fork Process Dispatcher
6. INITADP - UNIBUS Adapter Initialization Routines only  
(entry INI\$UBADP)
7. LIOSUB780 - Purge Data Path Routine
8. LOADMREG - Load Mapping Registers Routines
9. IOCIOPST - I/O Post-Processing Routines
10. COMDRVSUB - Communication Routines

## Solutions to Module Test Problems

1.   PCDRIVER.LIS           PC11 Reader/Punch Driver  
     PCDRIVER.COM       Assembles and Links Driver  
     PCDRIVER.OPT       Used for Linking Driver  
     LOADER.COM          Loads Driver and Connects Units  
     READV.FOR          Test Program  
     RANDWV.FOR         Test Program  
     MODE.FOR            Test Program
  
2.   ACPMNT.LIS           Mount Image for ACP  
     ACPMNT.COM         Assembles and Links Mount Image  
     PCACP.LIS           PC11 ACP  
     PCACP.COM          Assembles and Links ACP  
     PCDMT.LIS          Dismount Image for ACP  
     PCDMT.COM          Assembles and Links Dismount Image  
     TESTPROG.FOR       Test Program
  
3.   PCDRIVER.LIS       Set Attention PC11 Driver  
     PCDRIVER.COM       Assembles and Links Driver  
     PCDRIVER.OPT       Used for Linking Driver  
     PCLOAD.COM         Loads Driver and Connects Units  
     ATTN.FOR            Test Program -- Controls PC11
  
4.   PRINTER.LIS         Connect to Interrupt Example  
     PRINTER.COM        Assembles and Links Program  
     CINLOAD.COM        Connects Printer with Connect to  
                          Interrupt Driver



**CRDRIVER**

(1)	686	CR11 FUNCTION DECISION TABLE
(1)	724	CANCEL I/O ON CHANNEL
(1)	752	READ FUNCTION PROCESSING
(1)	824	START I/O OPERATION ON CR11 CARD READER
(1)	1007	CR11 CARD READER INTDERRUPTS
(1)	1063	CARD READER INITIALIZATION
(1)	1085	CARD READER UNIT INITIALIZATION

```
0000 1 .TITLE CRDRIVER - CR11 CARD READER DRIVER
;EMD0087 0000 .1 .IDENT 'V03-002'
-1 0000 3
0000 4 ;
0000 5 ;*****
0000 6 ;*
0000 7 ;* COPYRIGHT (c) 1978, 1980, 1982 BY *
0000 8 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 9 ;* ALL RIGHTS RESERVED. *
0000 10 ;*
0000 11 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 12 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 13 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 14 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 15 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 16 ;* TRANSFERRED. *
0000 17 ;*
0000 18 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 19 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 20 ;* CORPORATION. *
0000 21 ;*
0000 22 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 23 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 24 ;*
0000 25 ;*
0000 26 ;*****
0000 27 ;
0000 28 ; D. N. CUTLER 1-SEP-77
0000 29 ;
;EMD0087 0000 .1 ; MODIFICATION HISTORY:
;EMD0087 0000 .2 ;
;EMD0087 0000 .3 ; V03-002 EMD0087 Ellen M. Dusseault 30-Apr-1984
;EMD0087 0000 .4 ; Add DEV$M_NNM characteristic to DEVCHAR2 so that these
;EMD0087 0000 .5 ; devices will have the "node$" prefix.
;EMD0087 0000 .6 ;
;EMD0087 0000 .7 ; V03-001 KDM0002 Kathleen D. Morse 28-Jun-1982
;EMD0087 0000 .8 ; Added $DYNDEF, $SSDEF, $DCDEF, and $PRDEF.
;EMD0087 0000 .9 ;
-6 0000 36 ;
0000 37 ; MACRO LIBRARY CALLS
0000 38 ;
0000 39
0000 40 $CRBDEF ;DEFINE CRB OFFSETS
0000 41 $CRDEF ;DEFINE CARD READER STATUS BITS
;KDM0002 0000 .1 $DCDEF ;DEFINE ADAPTER TYPES
;EMD0087 0000 .2 $DDBDEF ;DEFINE DDB OFFSETS
;EMD0087 0000 .3 $DPTDEF ;DEFINE DPT OFFSETS
;KDM0002 0000 .4 $DYNDEF ;DEFINE DYNAMIC DATA STRUCTURE TYPES
-2 0000 44 $IOBDEF ;DEFINE IDB OFFSETS
0000 45 $IODEF ;DEFINE I/O FUNCTION CODES
0000 46 $IRPDEF ;DEFINE IRP OFFSETS
0000 47 $JIBDEF ;DEFINE JIB OFFSETS
0000 48 $MSGDEF ;DEFINE SYSTEM MESSAGE TYPES
0000 49 $PCBDEF ;DEFINE PCB OFFSETS
;KDM0002 0000 .1 $PRDEF ;DEFINE PROCESSOR REGISTERS
;KDM0002 0000 .2 $UCBDEF ;DEFINE UCB OFFSETS
;KDM0002 0000 .3 $SSDEF ;DEFINE STATUS CODES
```

```
-1          0000 51          $VECDEF                                ;DEFINE VEC OFFSETS
          0000 52
          0000 53 ;
          0000 54 ; LOCAL SYMBOLS
          0000 55 ;
          0000 56 ; ARGUMENT LIST OFFSET DEFINITIONS
          0000 57 ;
          0000 58
00000000 0000 59 P1=0                                ;FIRST FUNCTION DEPENDENT PARAMETER
00000004 0000 60 P2=4                                ;SECOND FUNCTION DEPENDENT PARAMETER
00000008 0000 61 P3=8                                ;THIRD FUNCTION DEPENDENT PARAMETER
0000000C 0000 62 P4=12                               ;FOURTH FUNCTION DEPENDENT PARAMETER
00000010 0000 63 P5=16                               ;FIFTH FUNCTION DEPENDENT PARAMETER
00000014 0000 64 P6=20                               ;SIZTH FUNCTION DEPENDENT PARAMETER
          0000 65
          0000 66 ;
          0000 67 ; SPECIAL CARD COLUMN PATTERNS
          0000 68 ;
          0000 69
00000F0F 0000 70 CR_EOF=^B111100001111              ;END OF FILE (12-11-0-1-6-7-8-9)
000008A2 0000 71 CR_026=^B100010100010            ;TRANSLATE 026 CARD CODE (12-2-4-8)
00000AAA 0000 72 CR_029=^B101010101010            ;TRANSLATE 029 CARD CODE (12-0-2-4-6-8)
          0000 73
          0000 74 ;
          0000 75 ; CR11 CONTROLLER REGISTER OFFSET DEFINITIONS
          0000 76 ;
          0000 77
          0000 78          $DEFINI CR
          0000 79
          0000 80 $DEF CR_CSR .BLKW 1                ;CONTROL STATUS REGISTER
          0002 81          _VFIELD CR_CSR,0,<-        ; CONTROL STATUS REGISTER FIELD DEFINITIONS
          0002 82          <READ,,M>,-              ; READ CARD
          0002 83          <EJECT,,M>,-             ; EJECT CARD
          0002 84          <,4>,-                   ; RESERVED BITS
          0002 85          <IE,,M>,-                 ; INTERRUPT ENABLE
          0002 86          <CLDONE,,M>,-             ; COLUMN DONE
          0002 87          <OFFLIN,,M>,-             ; READER OFFLINE
          0002 88          <BUSY,,M>,-               ; CARD BEING READ
          0002 89          <ONLINE,,M>,-             ; READER ONLINE
          0002 90          <TIMERR,,M>,-             ; TIMING ERROR
          0002 91          <MCHECK,,M>,-             ; MOTION CHECK
          0002 92          <HCHECK,,M>,-            ; HOPPER CHECK
          0002 93          <CRDONE,,M>,-            ; CARD DONE
          0002 94          <ERROR,,M>,-             ; ERROR CONDITION
          0002 95          >
          0002 96 $DEF CR_CRB1 .BLKW 1                ;CARD READ DATA BUFFER 1 (BINARY)
          0004 97 $DEF CR_CRB2 .BLKW 1                ;CARD READ DATA BUFFER 2 (PACKED)
          0006 98
          0006 99          $DEFEND CR
          0000 100
          0000 101 ;
          0000 102 ; DEFINE DEVICE DEPENDENT UNIT CONTROL BLOCK OFFSETS
          0000 103 ;
          0000 104
          0000 105          $DEFINI UCB
          0000 106
00000090 0000 107 .=UCB$K_LENGTH
```

```

0090 108
0090 109 $DEF UCB$B_CR_COLCNT .BLKB 1 ;CURRENT COLUMN COUNT
0091 110 $DEF UCB$B_CR_EOFcnt .BLKB 1 ;END OF FILE PUNCH COUNT
0092 111 $DEF UCB$B_CR_EOFcol .BLKB 1 ;NUMBER OF END OF FILE PUNCHES REQUIRED
0093 112 $DEF UCB$B_CR_OFcnt .BLKB 1 ;OFFLINE TIME COUNTER
0094 113 $DEF UCB$W_CR_FSTCOL .BLKW 1 ;FIRST COLUMN BINARY DATA
0096 114 $DEF UCB$W_CR_CSR .BLKW 1 ;SAVED FINAL CONTROL STATUS REGISTER
0098 115
00000098 0098 116 UCB$K_CR_LENGTH=.
0098 117
0098 118 $DEFEND UCB
0000 119
0000 120 ;
0000 121 ; LOCAL DATA
0000 122 ;
0000 123 ; DRIVER PROLOGUE TABLE
0000 124 ;
0000 125
0000 126 DPTAB - ;DEFINE DRIVER PROLOGUE TABLE
0000 127 END=CR_END,- ;END OF DRIVER
0000 128 ADAPTER=UBA,- ;ADAPTER TYPE
0000 129 UCBSIZE=UCB$K_CR_LENGTH,- ;UCB SIZE
0000 130 NAME=CRDRIVER ;DRIVER NAME
0038 131 DPT_STORE INIT ;CONTROL BLOCK INIT VALUES
0038 132 DPT_STORE UCB,UCB$B_FIPL,B,8 ;FORK IPL
003C 133 DPT_STORE UCB,UCB$L_DEVCHAR,L,- ;DEVICE CHARACTERISTICS
003C 134 <DEV$M_REC- ; RECORD ORIENTED
003C 135 !DEV$M_AVL- ; AVAILABLE
003C 136 !DEV$M_IDV> ; INPUT DEVICE
;EMD0087 0043 .1 DPT_STORE UCB,UCB$L_DEVCHAR2,L,- ; DEVICE CHARACTERISTIC
;EMD0087 0043 .2 <DEV$M_NNM> ; PREFIX WITH "NODE$"
004A 137 DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$_CARD ;DEVICE CLASS
004E 138 DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$_CR11 ;DEVICE TYPE
0052 139 DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,80 ;DEFAULT BUFFER SIZE
0057 140 DPT_STORE UCB,UCB$L_DEVDEPEND,L,CR$_T029 ;DEFAULT TRANSLATION MODE
005E 141 DPT_STORE UCB,UCB$B_DIPL,B,22 ;DEVICE IPL
0062 142 DPT_STORE REINIT ;CONTROL BLOCK RE-INIT VALUES
0062 143 DPT_STORE CRB,CRB$L_INTD+4,D,CR$_INT ;INTERRUPT SERVICE ROUTINE ADDRESS
0067 144 DPT_STORE CRB,CRB$L_INTD+VEC$L_INITIAL,D,CR_INITIAL ;CONTROLLER INIT
006C 145 DPT_STORE CRB,CRB$L_INTD+VEC$L_UNITINIT,D,CR_CR11_INIT ;UNIT INIT
0071 146 DPT_STORE DDB,DOB$L_DDT,D,CR$_DDT ;DDT ADDRESS
0076 147 DPT_STORE END ;
0000 148
0000 149 ;
0000 150 ; DRIVER DISPATCH TABLE
0000 151 ;
0000 152
0000 153 DDTAB CR,- ;DRIVER DISPATCH TABLE
0000 154 CR_STARTIO,- ;START I/O OPERATION
0000 155 0,- ;UNSOLICITED INTERRUPT
0000 156 CR_FUNCTABLE,- ;FUNCTION DECISION TABLE
0000 157 CR_CANCELIO,- ;CANCEL I/O OPERATION
0000 158 0,- ;REGISTER DUMP ROUTINE
0000 159 0,- ;SIZE OF DIAGNOSTIC BUFFER
0000 160 0 ;SIZE OF ERROR LOG BUFFER
0038 161
0038 162 ;

```

```
0038 163 ; 029 CONVERSION TABLE
0038 164 ;
0038 165
0038 166 CR_CVT029: ;029 TRANSLATE TABLE
20 0038 167 .BYTE ^A/ / ; 0 ^X00 ^0000
31 0039 168 .BYTE ^A/1/ ; 1 ^X01 ^0001
32 003A 169 .BYTE ^A/2/ ; 2 ^X02 ^0002
33 003B 170 .BYTE ^A/3/ ; 3 ^X03 ^0003
34 003C 171 .BYTE ^A/4/ ; 4 ^X04 ^0004
35 003D 172 .BYTE ^A/5/ ; 5 ^X05 ^0005
36 003E 173 .BYTE ^A/6/ ; 6 ^X06 ^0006
37 003F 174 .BYTE ^A/7/ ; 7 ^X07 ^0007
38 0040 175 .BYTE ^A/8/ ; 8 ^X08 ^0010
60 0041 176 .BYTE ^A/' / ; 9 ^X09 ^0011
3A 0042 177 .BYTE ^A/: / ; 10 ^X0A ^0012
23 0043 178 .BYTE ^A/#/ ; 11 ^X0B ^0013
40 0044 179 .BYTE ^A/@/ ; 12 ^X0C ^0014
27 0045 180 .BYTE ^A/' / ; 13 ^X0D ^0015
3D 0046 181 .BYTE ^A/= / ; 14 ^X0E ^0016
22 0047 182 .BYTE ^A/" / ; 15 ^X0F ^0017
39 0048 183 .BYTE ^A/9/ ; 16 ^X10 ^0020
5C 0049 184 .BYTE ^A/\ / ; 17 ^X11 ^0021
16 004A 185 .BYTE ^X16 ; 18 ^X12 ^0022
5C 004B 186 .BYTE ^A\/ / ; 19 ^X13 ^0023
5C 004C 187 .BYTE ^A\/ / ; 20 ^X14 ^0024
5C 004D 188 .BYTE ^A\/ / ; 21 ^X15 ^0025
5C 004E 189 .BYTE ^A\/ / ; 22 ^X16 ^0026
04 004F 190 .BYTE ^X04 ; 23 ^X17 ^0027
5C 0050 191 .BYTE ^A\/ / ; 24 ^X18 ^0030
5C 0051 192 .BYTE ^A\/ / ; 25 ^X19 ^0031
5C 0052 193 .BYTE ^A\/ / ; 26 ^X1A ^0032
5C 0053 194 .BYTE ^A\/ / ; 27 ^X1B ^0033
14 0054 195 .BYTE ^X14 ; 28 ^X1C ^0034
15 0055 196 .BYTE ^X15 ; 29 ^X1D ^0035
5C 0056 197 .BYTE ^A\/ / ; 30 ^X1E ^0036
1A 0057 198 .BYTE ^X1A ; 31 ^X1F ^0037
30 0058 199 .BYTE ^A/0/ ; 32 ^X20 ^0040
2F 0059 200 .BYTE ^A\/ / ; 33 ^X21 ^0041
53 005A 201 .BYTE ^A/S/ ; 34 ^X22 ^0042
54 005B 202 .BYTE ^A/T/ ; 35 ^X23 ^0043
55 005C 203 .BYTE ^A/U/ ; 36 ^X24 ^0044
56 005D 204 .BYTE ^A/V/ ; 37 ^X25 ^0045
57 005E 205 .BYTE ^A/W/ ; 38 ^X26 ^0046
58 005F 206 .BYTE ^A/X/ ; 39 ^X27 ^0047
59 0060 207 .BYTE ^A/Y/ ; 40 ^X28 ^0050
5C 0061 208 .BYTE ^A\/ / ; 41 ^X29 ^0051
5C 0062 209 .BYTE ^A\/ / ; 42 ^X2A ^0052
2C 0063 210 .BYTE ^A/, / ; 43 ^X2B ^0053
25 0064 211 .BYTE ^A/%/ ; 44 ^X2C ^0054
5F 0065 212 .BYTE ^A/_ / ; 45 ^X2D ^0055
3E 0066 213 .BYTE ^A/>/ ; 46 ^X2E ^0056
3F 0067 214 .BYTE ^A/?/ ; 47 ^X2F ^0057
5A 0068 215 .BYTE ^A/Z/ ; 48 ^X30 ^0060
5C 0069 216 .BYTE ^A\/ / ; 49 ^X31 ^0061
5C 006A 217 .BYTE ^A\/ / ; 50 ^X32 ^0062
5C 006B 218 .BYTE ^A\/ / ; 51 ^X33 ^0063
5C 006C 219 .BYTE ^A\/ / ; 52 ^X34 ^0064
```

0A	006D	220	.BYTE	^X0A	;	53	^X35	^0065
17	006E	221	.BYTE	^X17	;	54	^X36	^0066
1B	006F	222	.BYTE	^X1B	;	55	^X37	^0067
5C	0070	223	.BYTE	^A/\	;	56	^X38	^0070
5C	0071	224	.BYTE	^A/\	;	57	^X39	^0071
5C	0072	225	.BYTE	^A/\	;	58	^X3A	^0072
5C	0073	226	.BYTE	^A/\	;	59	^X3B	^0073
5C	0074	227	.BYTE	^A/\	;	60	^X3C	^0074
05	0075	228	.BYTE	^X05	;	61	^X3D	^0075
06	0076	229	.BYTE	^X06	;	62	^X3E	^0076
07	0077	230	.BYTE	^X07	;	63	^X3F	^0077
2D	0078	231	.BYTE	^A/-	;	64	^X40	^0100
4A	0079	232	.BYTE	^A/J	;	65	^X41	^0101
4B	007A	233	.BYTE	^A/K	;	66	^X42	^0102
4C	007B	234	.BYTE	^A/L	;	67	^X43	^0103
4D	007C	235	.BYTE	^A/M	;	68	^X44	^0104
4E	007D	236	.BYTE	^A/N	;	69	^X45	^0105
4F	007E	237	.BYTE	^A/O	;	70	^X46	^0106
50	007F	238	.BYTE	^A/P	;	71	^X47	^0107
51	0080	239	.BYTE	^A/Q	;	72	^X48	^0110
5C	0081	240	.BYTE	^A/\	;	73	^X49	^0111
5D	0082	241	.BYTE	^A/]	;	74	^X4A	^0112
24	0083	242	.BYTE	^A/\$	;	75	^X4B	^0113
2A	0084	243	.BYTE	^A/*	;	76	^X4C	^0114
29	0085	244	.BYTE	^A/)	;	77	^X4D	^0115
3B	0086	245	.BYTE	^A;/	;	78	^X4E	^0116
5E	0087	246	.BYTE	^A/^	;	79	^X4F	^0117
52	0088	247	.BYTE	^A/R	;	80	^X50	^0120
11	0089	248	.BYTE	^X11	;	81	^X51	^0121
12	008A	249	.BYTE	^X12	;	82	^X52	^0122
13	008B	250	.BYTE	^X13	;	83	^X53	^0123
5C	008C	251	.BYTE	^A/\	;	84	^X54	^0124
5C	008D	252	.BYTE	^A/\	;	85	^X55	^0125
08	008E	253	.BYTE	^X08	;	86	^X56	^0126
5C	008F	254	.BYTE	^A/\	;	87	^X57	^0127
18	0090	255	.BYTE	^X18	;	88	^X58	^0130
19	0091	256	.BYTE	^X19	;	89	^X59	^0131
5C	0092	257	.BYTE	^A/\	;	90	^X5A	^0132
5C	0093	258	.BYTE	^A/\	;	91	^X5B	^0133
1C	0094	259	.BYTE	^X1C	;	92	^X5C	^0134
1D	0095	260	.BYTE	^X1D	;	93	^X5D	^0135
1E	0096	261	.BYTE	^X1E	;	94	^X5E	^0136
1F	0097	262	.BYTE	^X1F	;	95	^X5F	^0137
7D	0098	263	.BYTE	^A/}	;	96	^X60	^0140
7E	0099	264	.BYTE	^A/~	;	97	^X61	^0141
73	009A	265	.BYTE	^A/s	;	98	^X62	^0142
74	009B	266	.BYTE	^A/t	;	99	^X63	^0143
75	009C	267	.BYTE	^A/u	;	100	^X64	^0144
76	009D	268	.BYTE	^A/v	;	101	^X65	^0145
77	009E	269	.BYTE	^A/w	;	102	^X66	^0146
78	009F	270	.BYTE	^A/x	;	103	^X67	^0147
79	00A0	271	.BYTE	^A/y	;	104	^X68	^0150
5C	00A1	272	.BYTE	^A/\	;	105	^X69	^0151
5C	00A2	273	.BYTE	^A/\	;	106	^X6A	^0152
5C	00A3	274	.BYTE	^A/\	;	107	^X6B	^0153
5C	00A4	275	.BYTE	^A/\	;	108	^X6C	^0154
5C	00A5	276	.BYTE	^A/\	;	109	^X6D	^0155

5C	00A6	277	.BYTE	^A/\	; 110	^X6E	^0156
5C	00A7	278	.BYTE	^A/\	; 111	^X6F	^0157
7A	00A8	279	.BYTE	^A/z/	; 112	^X70	^0160
5C	00A9	280	.BYTE	^A/\	; 113	^X71	^0161
5C	00AA	281	.BYTE	^A/\	; 114	^X72	^0162
5C	00AB	282	.BYTE	^A/\	; 115	^X73	^0163
5C	00AC	283	.BYTE	^A/\	; 116	^X74	^0164
5C	00AD	284	.BYTE	^A/\	; 117	^X75	^0165
5C	00AE	285	.BYTE	^A/\	; 118	^X76	^0166
5C	00AF	286	.BYTE	^A/\	; 119	^X77	^0167
5C	00B0	287	.BYTE	^A/\	; 120	^X78	^0170
5C	00B1	288	.BYTE	^A/\	; 121	^X79	^0171
5C	00B2	289	.BYTE	^A/\	; 122	^X7A	^0172
5C	00B3	290	.BYTE	^A/\	; 123	^X7B	^0173
5C	00B4	291	.BYTE	^A/\	; 124	^X7C	^0174
5C	00B5	292	.BYTE	^A/\	; 125	^X7D	^0175
5C	00B6	293	.BYTE	^A/\	; 126	^X7E	^0176
5C	00B7	294	.BYTE	^A/\	; 127	^X7F	^0177
26	00B8	295	.BYTE	^A/&/	; 128	^X80	^0200
41	00B9	296	.BYTE	^A/A/	; 129	^X81	^0201
42	00BA	297	.BYTE	^A/B/	; 130	^X82	^0202
43	00BB	298	.BYTE	^A/C/	; 131	^X83	^0203
44	00BC	299	.BYTE	^A/D/	; 132	^X84	^0204
45	00BD	300	.BYTE	^A/E/	; 133	^X85	^0205
46	00BE	301	.BYTE	^A/F/	; 134	^X86	^0206
47	00BF	302	.BYTE	^A/G/	; 135	^X87	^0207
48	00C0	303	.BYTE	^A/H/	; 136	^X88	^0210
5C	00C1	304	.BYTE	^A/\	; 137	^X89	^0211
5B	00C2	305	.BYTE	^A/[/	; 138	^X8A	^0212
2E	00C3	306	.BYTE	^A/. /	; 139	^X8B	^0213
3C	00C4	307	.BYTE	^A/</	; 140	^X8C	^0214
28	00C5	308	.BYTE	^A/( /	; 141	^X8D	^0215
2B	00C6	309	.BYTE	^A/+ /	; 142	^X8E	^0216
21	00C7	310	.BYTE	^A/! /	; 143	^X8F	^0217
49	00C8	311	.BYTE	^A/I /	; 144	^X90	^0220
01	00C9	312	.BYTE	^X01	; 145	^X91	^0221
02	00CA	313	.BYTE	^X02	; 146	^X92	^0222
03	00CB	314	.BYTE	^X03	; 147	^X93	^0223
5C	00CC	315	.BYTE	^A/\	; 148	^X94	^0224
09	00CD	316	.BYTE	^X09	; 149	^X95	^0225
5C	00CE	317	.BYTE	^A/\	; 150	^X96	^0226
FF	00CF	318	.BYTE	^XFF	; 151	^X97	^0227
5C	00D0	319	.BYTE	^A/\	; 152	^X98	^0230
5C	00D1	320	.BYTE	^A/\	; 153	^X99	^0231
5C	00D2	321	.BYTE	^A/\	; 154	^X9A	^0232
0B	00D3	322	.BYTE	^X0B	; 155	^X9B	^0233
0C	00D4	323	.BYTE	^X0C	; 156	^X9C	^0234
0D	00D5	324	.BYTE	^X0D	; 157	^X9D	^0235
0E	00D6	325	.BYTE	^X0E	; 158	^X9E	^0236
0F	00D7	326	.BYTE	^X0F	; 159	^X9F	^0237
7B	00D8	327	.BYTE	^A/{ /	; 160	^XA0	^0240
61	00D9	328	.BYTE	^A/a /	; 161	^XA1	^0241
62	00DA	329	.BYTE	^A/b /	; 162	^XA2	^0242
63	00DB	330	.BYTE	^A/c /	; 163	^XA3	^0243
64	00DC	331	.BYTE	^A/d /	; 164	^XA4	^0244
65	00DD	332	.BYTE	^A/e /	; 165	^XA5	^0245
66	00DE	333	.BYTE	^A/f /	; 166	^XA6	^0246



67	00DF	334	.BYTE	AA/g/		; 167	AXA7	^0247
68	00E0	335	.BYTE	AA/h/		; 168	AXA8	^0250
5C	00E1	336	.BYTE	AA//		; 169	AXA9	^0251
5C	00E2	337	.BYTE	AA//		; 170	AXAA	^0252
5C	00E3	338	.BYTE	AA//		; 171	AXAB	^0253
5C	00E4	339	.BYTE	AA//		; 172	AXAC	^0254
5C	00E5	340	.BYTE	AA//		; 173	AXAD	^0255
5C	00E6	341	.BYTE	AA//		; 174	AXAE	^0256
5C	00E7	342	.BYTE	AA//		; 175	AXAF	^0257
69	00E8	343	.BYTE	AA/i/		; 176	AXB0	^0260
5C	00E9	344	.BYTE	AA//		; 177	AXB1	^0261
5C	00EA	345	.BYTE	AA//		; 178	AXB2	^0262
5C	00EB	346	.BYTE	AA//		; 179	AXB3	^0263
5C	00EC	347	.BYTE	AA//		; 180	AXB4	^0264
5C	00ED	348	.BYTE	AA//		; 181	AXB5	^0265
5C	00EE	349	.BYTE	AA//		; 182	AXB6	^0266
5C	00EF	350	.BYTE	AA//		; 183	AXB7	^0267
5C	00F0	351	.BYTE	AA//		; 184	AXB8	^0270
5C	00F1	352	.BYTE	AA//		; 185	AXB9	^0271
5C	00F2	353	.BYTE	AA//		; 186	AXBA	^0272
5C	00F3	354	.BYTE	AA//		; 187	AXBB	^0273
5C	00F4	355	.BYTE	AA//		; 188	AXBC	^0274
5C	00F5	356	.BYTE	AA//		; 189	AXBD	^0275
5C	00F6	357	.BYTE	AA//		; 190	AXBE	^0276
5C	00F7	358	.BYTE	AA//		; 191	AXBF	^0277
7C	00F8	359	.BYTE	AA/j/		; 192	AXC0	^0300
6A	00F9	360	.BYTE	AA/k/		; 193	AXC1	^0301
6B	00FA	361	.BYTE	AA/l/		; 194	AXC2	^0302
6C	00FB	362	.BYTE	AA/m/		; 195	AXC3	^0303
6D	00FC	363	.BYTE	AA/n/		; 196	AXC4	^0304
6E	00FD	364	.BYTE	AA/o/		; 197	AXC5	^0305
6F	00FE	365	.BYTE	AA/p/		; 198	AXC6	^0306
70	00FF	366	.BYTE	AA/q/		; 199	AXC7	^0307
71	0100	367	.BYTE	AA//		; 200	AXC8	^0310
5C	0101	368	.BYTE	AA//		; 201	AXC9	^0311
5C	0102	369	.BYTE	AA//		; 202	AXCA	^0312
5C	0103	370	.BYTE	AA//		; 203	AXCB	^0313
5C	0104	371	.BYTE	AA//		; 204	AXCC	^0314
5C	0105	372	.BYTE	AA//		; 205	AXCD	^0315
5C	0106	373	.BYTE	AA//		; 206	AXCE	^0316
5C	0107	374	.BYTE	AA//		; 207	AXCF	^0317
72	0108	375	.BYTE	AA/r/		; 208	AXD0	^0320
5C	0109	376	.BYTE	AA//		; 209	AXD1	^0321
5C	010A	377	.BYTE	AA//		; 210	AXD2	^0322
5C	010B	378	.BYTE	AA//		; 211	AXD3	^0323
5C	010C	379	.BYTE	AA//		; 212	AXD4	^0324
5C	010D	380	.BYTE	AA//		; 213	AXD5	^0325
5C	010E	381	.BYTE	AA//		; 214	AXD6	^0326
5C	010F	382	.BYTE	AA//		; 215	AXD7	^0327
5C	0110	383	.BYTE	AA//		; 216	AXD8	^0330
10	0111	384	.BYTE	AX10		; 217	AXD9	^0331
5C	0112	385	.BYTE	AA//		; 218	AXDA	^0332
5C	0113	386	.BYTE	AA//		; 219	AXDB	^0333
5C	0114	387	.BYTE	AA//		; 220	AXDC	^0334
5C	0115	388	.BYTE	AA//		; 221	AXDD	^0335
5C	0116	389	.BYTE	AA//		; 222	AXDE	^0336
5C	0117	390	.BYTE	AA//		; 223	AXDF	^0337

```

5C 0118 391 .BYTE ^A/\ ; 224 ^XE0 ^0340
5C 0119 392 .BYTE ^A/\ ; 225 ^XE1 ^0341
5C 011A 393 .BYTE ^A/\ ; 226 ^XE2 ^0342
5C 011B 394 .BYTE ^A/\ ; 227 ^XE3 ^0343
5C 011C 395 .BYTE ^A/\ ; 228 ^XE4 ^0344
5C 011D 396 .BYTE ^A/\ ; 229 ^XE5 ^0345
5C 011E 397 .BYTE ^A/\ ; 230 ^XE6 ^0346
5C 011F 398 .BYTE ^A/\ ; 231 ^XE7 ^0347
5C 0120 399 .BYTE ^A/\ ; 232 ^XE8 ^0350
5C 0121 400 .BYTE ^A/\ ; 233 ^XE9 ^0351
5C 0122 401 .BYTE ^A/\ ; 234 ^XEA ^0352
5C 0123 402 .BYTE ^A/\ ; 235 ^XEB ^0353
5C 0124 403 .BYTE ^A/\ ; 236 ^XEC ^0354
5C 0125 404 .BYTE ^A/\ ; 237 ^XED ^0355
5C 0126 405 .BYTE ^A/\ ; 238 ^XEE ^0356
5C 0127 406 .BYTE ^A/\ ; 239 ^XEF ^0357
5C 0128 407 .BYTE ^A/\ ; 240 ^XF0 ^0360
5C 0129 408 .BYTE ^A/\ ; 241 ^XF1 ^0361
5C 012A 409 .BYTE ^A/\ ; 242 ^XF2 ^0362
5C 012B 410 .BYTE ^A/\ ; 243 ^XF3 ^0363
5C 012C 411 .BYTE ^A/\ ; 244 ^XF4 ^0364
5C 012D 412 .BYTE ^A/\ ; 245 ^XF5 ^0365
5C 012E 413 .BYTE ^A/\ ; 246 ^XF6 ^0366
5C 012F 414 .BYTE ^A/\ ; 247 ^XF7 ^0367
5C 0130 415 .BYTE ^A/\ ; 248 ^XF8 ^0370
5C 0131 416 .BYTE ^A/\ ; 249 ^XF9 ^0371
5C 0132 417 .BYTE ^A/\ ; 250 ^XFA ^0372
5C 0133 418 .BYTE ^A/\ ; 251 ^XFB ^0373
5C 0134 419 .BYTE ^A/\ ; 252 ^XFC ^0374
5C 0135 420 .BYTE ^A/\ ; 253 ^XFD ^0375
5C 0136 421 .BYTE ^A/\ ; 254 ^XFE ^0376
5C 0137 422 .BYTE ^A/\ ; 255 ^XFF ^0377
0138 423
0138 424 ;
0138 425 ; 026 CONVERSION TABLE
0138 426 ;
0138 427
0138 428 CR_CVT026: ;026 TRANSLATE TABLE
20 0138 429 .BYTE ^A/ / ; 0 ^X00 ^0000
31 0139 430 .BYTE ^A/1/ ; 1 ^X01 ^0001
32 013A 431 .BYTE ^A/2/ ; 2 ^X02 ^0002
33 013B 432 .BYTE ^A/3/ ; 3 ^X03 ^0003
34 013C 433 .BYTE ^A/4/ ; 4 ^X04 ^0004
35 013D 434 .BYTE ^A/5/ ; 5 ^X05 ^0005
36 013E 435 .BYTE ^A/6/ ; 6 ^X06 ^0006
37 013F 436 .BYTE ^A/7/ ; 7 ^X07 ^0007
38 0140 437 .BYTE ^A/8/ ; 8 ^X08 ^0010
60 0141 438 .BYTE ^A/ / ; 9 ^X09 ^0011
5F 0142 439 .BYTE ^A/ / ; 10 ^X0A ^0012
3D 0143 440 .BYTE ^A/= / ; 11 ^X0B ^0013
40 0144 441 .BYTE ^A/@/ ; 12 ^X0C ^0014
5E 0145 442 .BYTE ^A/^ / ; 13 ^X0D ^0015
27 0146 443 .BYTE ^A/' / ; 14 ^X0E ^0016
5C 0147 444 .BYTE ^A/\ / ; 15 ^X0F ^0017
39 0148 445 .BYTE ^A/9/ ; 16 ^X10 ^0020
5C 0149 446 .BYTE ^A/\ / ; 17 ^X11 ^0021
16 014A 447 .BYTE ^X16 ; 18 ^X12 ^0022

```

5C	014B	448	.BYTE	^A/\	;	19	^X13	^0023
5C	014C	449	.BYTE	^A/\	;	20	^X14	^0024
5C	014D	450	.BYTE	^A/\	;	21	^X15	^0025
5C	014E	451	.BYTE	^A/\	;	22	^X16	^0026
04	014F	452	.BYTE	^X04	;	23	^X17	^0027
5C	0150	453	.BYTE	^A/\	;	24	^X18	^0030
5C	0151	454	.BYTE	^A/\	;	25	^X19	^0031
5C	0152	455	.BYTE	^A/\	;	26	^X1A	^0032
5C	0153	456	.BYTE	^A/\	;	27	^X1B	^0033
14	0154	457	.BYTE	^X14	;	28	^X1C	^0034
15	0155	458	.BYTE	^X15	;	29	^X1D	^0035
5C	0156	459	.BYTE	^A/\	;	30	^X1E	^0036
1A	0157	460	.BYTE	^X1A	;	31	^X1F	^0037
30	0158	461	.BYTE	^A/O/	;	32	^X20	^0040
2F	0159	462	.BYTE	^A/\	;	33	^X21	^0041
53	015A	463	.BYTE	^A/S/	;	34	^X22	^0042
54	015B	464	.BYTE	^A/T/	;	35	^X23	^0043
55	015C	465	.BYTE	^A/U/	;	36	^X24	^0044
56	015D	466	.BYTE	^A/V/	;	37	^X25	^0045
57	015E	467	.BYTE	^A/W/	;	38	^X26	^0046
58	015F	468	.BYTE	^A/X/	;	39	^X27	^0047
59	0160	469	.BYTE	^A/Y/	;	40	^X28	^0050
5C	0161	470	.BYTE	^A/\	;	41	^X29	^0051
3B	0162	471	.BYTE	^A/;/	;	42	^X2A	^0052
2C	0163	472	.BYTE	^A/,/	;	43	^X2B	^0053
28	0164	473	.BYTE	^A/(/	;	44	^X2C	^0054
22	0165	474	.BYTE	^A/"/	;	45	^X2D	^0055
23	0166	475	.BYTE	^A/#/	;	46	^X2E	^0056
25	0167	476	.BYTE	^A/%/	;	47	^X2F	^0057
5A	0168	477	.BYTE	^A/Z/	;	48	^X30	^0060
5C	0169	478	.BYTE	^A/\	;	49	^X31	^0061
5C	016A	479	.BYTE	^A/\	;	50	^X32	^0062
5C	016B	480	.BYTE	^A/\	;	51	^X33	^0063
5C	016C	481	.BYTE	^A/\	;	52	^X34	^0064
0A	016D	482	.BYTE	^X0A	;	53	^X35	^0065
17	016E	483	.BYTE	^X17	;	54	^X36	^0066
1B	016F	484	.BYTE	^X1B	;	55	^X37	^0067
5C	0170	485	.BYTE	^A/\	;	56	^X38	^0070
5C	0171	486	.BYTE	^A/\	;	57	^X39	^0071
5C	0172	487	.BYTE	^A/\	;	58	^X3A	^0072
5C	0173	488	.BYTE	^A/\	;	59	^X3B	^0073
5C	0174	489	.BYTE	^A/\	;	60	^X3C	^0074
05	0175	490	.BYTE	^X05	;	61	^X3D	^0075
06	0176	491	.BYTE	^X06	;	62	^X3E	^0076
07	0177	492	.BYTE	^X07	;	63	^X3F	^0077
2D	0178	493	.BYTE	^A/-/	;	64	^X40	^0100
4A	0179	494	.BYTE	^A/J/	;	65	^X41	^0101
4B	017A	495	.BYTE	^A/K/	;	66	^X42	^0102
4C	017B	496	.BYTE	^A/L/	;	67	^X43	^0103
4D	017C	497	.BYTE	^A/M/	;	68	^X44	^0104
4E	017D	498	.BYTE	^A/N/	;	69	^X45	^0105
4F	017E	499	.BYTE	^A/O/	;	70	^X46	^0106
50	017F	500	.BYTE	^A/P/	;	71	^X47	^0107
51	0180	501	.BYTE	^A/Q/	;	72	^X48	^0110
5C	0181	502	.BYTE	^A/\	;	73	^X49	^0111
3A	0182	503	.BYTE	^A/:/	;	74	^X4A	^0112
24	0183	504	.BYTE	^A/\$/	;	75	^X4B	^0113

2A	0184	505	.BYTE	^A/*/	;	76	^X4C	^O114
5B	0185	506	.BYTE	^A/[/	;	77	^X4D	^O115
3E	0186	507	.BYTE	^A/>/	;	78	^X4E	^O116
26	0187	508	.BYTE	^A/&/	;	79	^X4F	^O117
52	0188	509	.BYTE	^A/R/	;	80	^X50	^O120
11	0189	510	.BYTE	^X11	;	81	^X51	^O121
12	018A	511	.BYTE	^X12	;	82	^X52	^O122
13	018B	512	.BYTE	^X13	;	83	^X53	^O123
5C	018C	513	.BYTE	^A/\//	;	84	^X54	^O124
5C	018D	514	.BYTE	^A/\//	;	85	^X55	^O125
08	018E	515	.BYTE	^X08	;	86	^X56	^O126
5C	018F	516	.BYTE	^A/\//	;	87	^X57	^O127
18	0190	517	.BYTE	^X18	;	88	^X58	^O130
19	0191	518	.BYTE	^X19	;	89	^X59	^O131
5C	0192	519	.BYTE	^A/\//	;	90	^X5A	^O132
5C	0193	520	.BYTE	^A/\//	;	91	^X5B	^O133
1C	0194	521	.BYTE	^X1C	;	92	^X5C	^O134
1D	0195	522	.BYTE	^X1D	;	93	^X5D	^O135
1E	0196	523	.BYTE	^X1E	;	94	^X5E	^O136
1F	0197	524	.BYTE	^X1F	;	95	^X5F	^O137
7D	0198	525	.BYTE	^A/}/	;	96	^X60	^O140
7E	0199	526	.BYTE	^A/~//	;	97	^X61	^O141
73	019A	527	.BYTE	^A/s/	;	98	^X62	^O142
74	019B	528	.BYTE	^A/t/	;	99	^X63	^O143
75	019C	529	.BYTE	^A/u/	;	100	^X64	^O144
76	019D	530	.BYTE	^A/v/	;	101	^X65	^O145
77	019E	531	.BYTE	^A/w/	;	102	^X66	^O146
78	019F	532	.BYTE	^A/x/	;	103	^X67	^O147
79	01A0	533	.BYTE	^A/y/	;	104	^X68	^O150
5C	01A1	534	.BYTE	^A/\//	;	105	^X69	^O151
5C	01A2	535	.BYTE	^A/\//	;	106	^X6A	^O152
5C	01A3	536	.BYTE	^A/\//	;	107	^X6B	^O153
5C	01A4	537	.BYTE	^A/\//	;	108	^X6C	^O154
5C	01A5	538	.BYTE	^A/\//	;	109	^X6D	^O155
5C	01A6	539	.BYTE	^A/\//	;	110	^X6E	^O156
5C	01A7	540	.BYTE	^A/\//	;	111	^X6F	^O157
7A	01A8	541	.BYTE	^A/z/	;	112	^X70	^O160
5C	01A9	542	.BYTE	^A/\//	;	113	^X71	^O161
5C	01AA	543	.BYTE	^A/\//	;	114	^X72	^O162
5C	01AB	544	.BYTE	^A/\//	;	115	^X73	^O163
5C	01AC	545	.BYTE	^A/\//	;	116	^X74	^O164
5C	01AD	546	.BYTE	^A/\//	;	117	^X75	^O165
5C	01AE	547	.BYTE	^A/\//	;	118	^X76	^O166
5C	01AF	548	.BYTE	^A/\//	;	119	^X77	^O167
5C	01B0	549	.BYTE	^A/\//	;	120	^X78	^O170
5C	01B1	550	.BYTE	^A/\//	;	121	^X79	^O171
5C	01B2	551	.BYTE	^A/\//	;	122	^X7A	^O172
5C	01B3	552	.BYTE	^A/\//	;	123	^X7B	^O173
5C	01B4	553	.BYTE	^A/\//	;	124	^X7C	^O174
5C	01B5	554	.BYTE	^A/\//	;	125	^X7D	^O175
5C	01B6	555	.BYTE	^A/\//	;	126	^X7E	^O176
5C	01B7	556	.BYTE	^A/\//	;	127	^X7F	^O177
2B	01B8	557	.BYTE	^A/+/	;	128	^X80	^O200
41	01B9	558	.BYTE	^A/A/	;	129	^X81	^O201
42	01BA	559	.BYTE	^A/B/	;	130	^X82	^O202
43	01BB	560	.BYTE	^A/C/	;	131	^X83	^O203
44	01BC	561	.BYTE	^A/D/	;	132	^X84	^O204

45	01BD	562	.BYTE	^A/E/	; 133	^X85	^0205
46	01BE	563	.BYTE	^A/F/	; 134	^X86	^0206
47	01BF	564	.BYTE	^A/G/	; 135	^X87	^0207
48	01C0	565	.BYTE	^A/H/	; 136	^X88	^0210
5C	01C1	566	.BYTE	^A/I/	; 137	^X89	^0211
3F	01C2	567	.BYTE	^A/?/	; 138	^X8A	^0212
2E	01C3	568	.BYTE	^A/. /	; 139	^X8B	^0213
29	01C4	569	.BYTE	^A/)/	; 140	^X8C	^0214
5D	01C5	570	.BYTE	^A/] /	; 141	^X8D	^0215
3C	01C6	571	.BYTE	^A/</	; 142	^X8E	^0216
21	01C7	572	.BYTE	^A!/ /	; 143	^X8F	^0217
49	01C8	573	.BYTE	^A/I /	; 144	^X90	^0220
01	01C9	574	.BYTE	^X01	; 145	^X91	^0221
02	01CA	575	.BYTE	^X02	; 146	^X92	^0222
03	01CB	576	.BYTE	^X03	; 147	^X93	^0223
5C	01CC	577	.BYTE	^A/\ /	; 148	^X94	^0224
09	01CD	578	.BYTE	^X09	; 149	^X95	^0225
5C	01CE	579	.BYTE	^A/\ /	; 150	^X96	^0226
FF	01CF	580	.BYTE	^XFF	; 151	^X97	^0227
5C	01D0	581	.BYTE	^A/\ /	; 152	^X98	^0230
5C	01D1	582	.BYTE	^A/\ /	; 153	^X99	^0231
5C	01D2	583	.BYTE	^A/\ /	; 154	^X9A	^0232
0B	01D3	584	.BYTE	^X0B	; 155	^X9B	^0233
0C	01D4	585	.BYTE	^X0C	; 156	^X9C	^0234
0D	01D5	586	.BYTE	^X0D	; 157	^X9D	^0235
0E	01D6	587	.BYTE	^X0E	; 158	^X9E	^0236
0F	01D7	588	.BYTE	^X0F	; 159	^X9F	^0237
7B	01D8	589	.BYTE	^A/{ /	; 160	^XA0	^0240
61	01D9	590	.BYTE	^A/a /	; 161	^XA1	^0241
62	01DA	591	.BYTE	^A/b /	; 162	^XA2	^0242
63	01DB	592	.BYTE	^A/c /	; 163	^XA3	^0243
64	01DC	593	.BYTE	^A/d /	; 164	^XA4	^0244
65	01DD	594	.BYTE	^A/e /	; 165	^XA5	^0245
66	01DE	595	.BYTE	^A/f /	; 166	^XA6	^0246
67	01DF	596	.BYTE	^A/g /	; 167	^XA7	^0247
68	01E0	597	.BYTE	^A/h /	; 168	^XA8	^0250
5C	01E1	598	.BYTE	^A/\ /	; 169	^XA9	^0251
5C	01E2	599	.BYTE	^A/\ /	; 170	^XAA	^0252
5C	01E3	600	.BYTE	^A/\ /	; 171	^XAB	^0253
5C	01E4	601	.BYTE	^A/\ /	; 172	^XAC	^0254
5C	01E5	602	.BYTE	^A/\ /	; 173	^XAD	^0255
5C	01E6	603	.BYTE	^A/\ /	; 174	^XAE	^0256
5C	01E7	604	.BYTE	^A/\ /	; 175	^XAF	^0257
69	01E8	605	.BYTE	^A/i /	; 176	^XB0	^0260
5C	01E9	606	.BYTE	^A/\ /	; 177	^XB1	^0261
5C	01EA	607	.BYTE	^A/\ /	; 178	^XB2	^0262
5C	01EB	608	.BYTE	^A/\ /	; 179	^XB3	^0263
5C	01EC	609	.BYTE	^A/\ /	; 180	^XB4	^0264
5C	01ED	610	.BYTE	^A/\ /	; 181	^XB5	^0265
5C	01EE	611	.BYTE	^A/\ /	; 182	^XB6	^0266
5C	01EF	612	.BYTE	^A/\ /	; 183	^XB7	^0267
5C	01F0	613	.BYTE	^A/\ /	; 184	^XB8	^0270
5C	01F1	614	.BYTE	^A/\ /	; 185	^XB9	^0271
5C	01F2	615	.BYTE	^A/\ /	; 186	^XBA	^0272
5C	01F3	616	.BYTE	^A/\ /	; 187	^XBB	^0273
5C	01F4	617	.BYTE	^A/\ /	; 188	^XBC	^0274
5C	01F5	618	.BYTE	^A/\ /	; 189	^XBD	^0275

5C 01F6	619	.BYTE	^A//	; 190	^XBE	^0276
5C 01F7	620	.BYTE	^A//	; 191	^XBF	^0277
7C 01F8	621	.BYTE	^A/ /	; 192	^XC0	^0300
6A 01F9	622	.BYTE	^A/j/	; 193	^XC1	^0301
6B 01FA	623	.BYTE	^A/k/	; 194	^XC2	^0302
6C 01FB	624	.BYTE	^A/l/	; 195	^XC3	^0303
6D 01FC	625	.BYTE	^A/m/	; 196	^XC4	^0304
6E 01FD	626	.BYTE	^A/n/	; 197	^XC5	^0305
6F 01FE	627	.BYTE	^A/o/	; 198	^XC6	^0306
70 01FF	628	.BYTE	^A/p/	; 199	^XC7	^0307
71 0200	629	.BYTE	^A/q/	; 200	^XC8	^0310
5C 0201	630	.BYTE	^A//	; 201	^XC9	^0311
5C 0202	631	.BYTE	^A//	; 202	^XCA	^0312
5C 0203	632	.BYTE	^A//	; 203	^XCB	^0313
5C 0204	633	.BYTE	^A//	; 204	^XCC	^0314
5C 0205	634	.BYTE	^A//	; 205	^XCD	^0315
5C 0206	635	.BYTE	^A//	; 206	^XCE	^0316
5C 0207	636	.BYTE	^A//	; 207	^XCF	^0317
72 0208	637	.BYTE	^A/r/	; 208	^XD0	^0320
5C 0209	638	.BYTE	^A//	; 209	^XD1	^0321
5C 020A	639	.BYTE	^A//	; 210	^XD2	^0322
5C 020B	640	.BYTE	^A//	; 211	^XD3	^0323
5C 020C	641	.BYTE	^A//	; 212	^XD4	^0324
5C 020D	642	.BYTE	^A//	; 213	^XD5	^0325
5C 020E	643	.BYTE	^A//	; 214	^XD6	^0326
5C 020F	644	.BYTE	^A//	; 215	^XD7	^0327
5C 0210	645	.BYTE	^A//	; 216	^XD8	^0330
10 0211	646	.BYTE	^X10	; 217	^XD9	^0331
5C 0212	647	.BYTE	^A//	; 218	^XDA	^0332
5C 0213	648	.BYTE	^A//	; 219	^XDB	^0333
5C 0214	649	.BYTE	^A//	; 220	^XDC	^0334
5C 0215	650	.BYTE	^A//	; 221	^XDD	^0335
5C 0216	651	.BYTE	^A//	; 222	^XDE	^0336
5C 0217	652	.BYTE	^A//	; 223	^XDF	^0337
5C 0218	653	.BYTE	^A//	; 224	^XE0	^0340
5C 0219	654	.BYTE	^A//	; 225	^XE1	^0341
5C 021A	655	.BYTE	^A//	; 226	^XE2	^0342
5C 021B	656	.BYTE	^A//	; 227	^XE3	^0343
5C 021C	657	.BYTE	^A//	; 228	^XE4	^0344
5C 021D	658	.BYTE	^A//	; 229	^XE5	^0345
5C 021E	659	.BYTE	^A//	; 230	^XE6	^0346
5C 021F	660	.BYTE	^A//	; 231	^XE7	^0347
5C 0220	661	.BYTE	^A//	; 232	^XE8	^0350
5C 0221	662	.BYTE	^A//	; 233	^XE9	^0351
5C 0222	663	.BYTE	^A//	; 234	^XEA	^0352
5C 0223	664	.BYTE	^A//	; 235	^XEB	^0353
5C 0224	665	.BYTE	^A//	; 236	^XEC	^0354
5C 0225	666	.BYTE	^A//	; 237	^XED	^0355
5C 0226	667	.BYTE	^A//	; 238	^XEE	^0356
5C 0227	668	.BYTE	^A//	; 239	^XEF	^0357
5C 0228	669	.BYTE	^A//	; 240	^XF0	^0360
5C 0229	670	.BYTE	^A//	; 241	^XF1	^0361
5C 022A	671	.BYTE	^A//	; 242	^XF2	^0362
5C 022B	672	.BYTE	^A//	; 243	^XF3	^0363
5C 022C	673	.BYTE	^A//	; 244	^XF4	^0364
5C 022D	674	.BYTE	^A//	; 245	^XF5	^0365
5C 022E	675	.BYTE	^A//	; 246	^XF6	^0366

CRDRIVER  
V03-002

- CR11 CARD READER DRIVER

3-JUN-1984 11:02:03 VAX-11 Macro V03-01 Page 13  
10-MAR-1982 20:31:59 DISK\$VMSMASTER:[DRIVER.SRC]CRDRIVE(1)

5C	022F	676	.BYTE	^A/\		; 247	^XF7	^0367
5C	0230	677	.BYTE	^A/\		; 248	^XF8	^0370
5C	0231	678	.BYTE	^A/\		; 249	^XF9	^0371
5C	0232	679	.BYTE	^A/\		; 250	^XFA	^0372
5C	0233	680	.BYTE	^A/\		; 251	^XFB	^0373
5C	0234	681	.BYTE	^A/\		; 252	^XFC	^0374
5C	0235	682	.BYTE	^A/\		; 253	^XFD	^0375
5C	0236	683	.BYTE	^A/\		; 254	^XFE	^0376
5C	0237	684	.BYTE	^A/\		; 255	^XFF	^0377

```

0238 686 .SBTTL CR11 FUNCTION DECISION TABLE
0238 687 ;
0238 688 ; CR11 FUNCTION DECISION TABLE
0238 689 ;
0238 690
0238 691 CR_FUNCABLE: ;FUNCTION DECISION TABLE
0238 692 FUNCTAB ,- ;LEGAL FUNCTION
0238 693 <READLBLK,- ;READ LOGICAL BLOCK
0238 694 READPBLK,- ;READ PHYSICAL BLOCK
0238 695 READVBLK,- ;READ VIRTUAL BLOCK
0238 696 SENSEMODE,- ;SENSE READ MODE
0238 697 SENSECHAR,- ;SENSE READER CHARACTERISTICS
0238 698 SETMODE,- ;SET READER MODE
0238 699 SETCHAR,- ;SET READER CHARACTERISTICS
0238 700 > ;
0240 701 FUNCTAB ,- ;BUFFERED I/O FUNCTIONS
0240 702 <READLBLK,- ;READ LOGICAL BLOCK
0240 703 READPBLK,- ;READ PHYSICAL BLOCK
0240 704 READVBLK,- ;READ VIRTUAL BLOCK
0240 705 SENSEMODE,- ;SENSE READ MODE
0240 706 SENSECHAR,- ;SENSE READER CHARACTERISTICS
0240 707 SETMODE,- ;SET READER MODE
0240 708 SETCHAR,- ;SET READER CHARACTERISTICS
0240 709 > ;
0248 710 FUNCTAB CR_READ,- ;READ FUNCTIONS
0248 711 <READLBLK,- ;READ LOGICAL BLOCK
0248 712 READPBLK,- ;READ PHYSICAL BLOCK
0248 713 READVBLK,- ;READ VIRTUAL BLOCK
0248 714 > ;
0254 715 FUNCTAB +EXE$SETMODE,- ;SET MODE/CHARACTERISTICS FUNCTIONS
0254 716 <SETCHAR,- ;SET READER CHARACTERISTICS
0254 717 SETMODE,- ;SET READER MODE
0254 718 > ;
0260 719 FUNCTAB +EXE$SENSEMODE,- ;SENSE MODE/CHARACTERISTICS FUNCTIONS
0260 720 <SENSECHAR,- ;SENSE READER CHARACTERISTICS
0260 721 SENSEMODE,- ;SENSE READER MODE
0260 722 > ;

```



```

026C 724 .SBTTL CANCEL I/O ON CHANNEL
026C 725 ;+
026C 726 ; CR_CANCELIO - CANCEL I/O ON CHANNEL
026C 727 ;
026C 728 ; THIS ROUTINE IS CALLED WHEN THE LAST CHANNEL ASSIGNED TO A DEVICE IS DEASSIGNED,
026C 729 ; THE DEVICE IS DEALLOCATED, AND WHEN THE CANCEL I/O ON CHANNEL SYSTEM SERVICE IS
026C 730 ; EXECUTED.
026C 731 ;
026C 732 ; INPUTS:
026C 733 ;
026C 734 ; R2 = NEGATIVE CHANNEL NUMBER.
026C 735 ; R3 = ADDRESS OF CURRENT I/O REQUEST PACKET.
026C 736 ; R4 = CURRENT PROCESS PCB ADDRESS.
026C 737 ; R5 = DEVICE UCB ADDRESS.
026C 738 ;
026C 739 ; OUTPUTS:
026C 740 ;
026C 741 ; THE DEVICE INDEPENDENT CANCEL I/O ROUTINE IS CALLED AND A CHECK IS MADE
026C 742 ; TO SEE IF THE UCB REFERENCE COUNT IS ZERO. IF THE REFERENCE COUNT IS ZERO,
026C 743 ; THEN THE MESSAGE SENT TO JOB CONTROLLER BIT IS CLEARED.
026C 744 ;-
026C 745
026C 746 CR_CANCELIO: ;CANCEL I/O ON CHANNEL
5C A5 B5 026C 747 TSTW UCB$W_REFC(R5) ;REFERENCE COUNT ZERO?
04 12 026F 748 BNEQ 10$ ;IF NEQ NO
68 A5 01 AA 0271 749 BICW #UCB$M_JOB,UCB$W_DEVSTS(R5) ;CLEAR MESSAGE SENT BIT
00000000'GF 17 0275 750 10$: JMP G^IIOC$CANCELIO ;CANCEL I/O ON CHANNEL

```

```

027B 752          .SBTTL  READ FUNCTION PROCESSING
027B 753 ;+
027B 754 ; CR_READ - READ FUNCTION PROCESSING
027B 755 ;
027B 756 ; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO PROCESS
027B 757 ; A READ LOGICAL, READ PHYSICAL, OR READ VIRTUAL FUNCTION TO A CARD READER.
027B 758 ;
027B 759 ; INPUTS:
027B 760 ;
027B 761 ;     R0 = SCRATCH.
027B 762 ;     R1 = SCRATCH.
027B 763 ;     R2 = SCRATCH.
027B 764 ;     R3 = ADDRESS OF I/O REQUEST PACKET.
027B 765 ;     R4 = CURRENT PROCESS PCB ADDRESS.
027B 766 ;     R5 = ASSIGNED DEVICE UCB ADDRESS.
027B 767 ;     R6 = ADDRESS OF CCB.
027B 768 ;     R7 = I/O FUNCTION CODE.
027B 769 ;     R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
027B 770 ;     R9 = SCRATCH.
027B 771 ;     R10 = SCRATCH.
027B 772 ;     R11 = SCRATCH.
027B 773 ;     AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
027B 774 ;
027B 775 ; OUTPUTS:
027B 776 ;
027B 777 ;     THE FUNCTION PARAMETERS ARE CHECKED AND A BUFFER IS ALLOCATED FOR THE
027B 778 ;     CARD READER DRIVER TO READ A CARD IMAGE INTO.
027B 779 ;-
027B 780
027B 781 CR_READ:
027B 782          MOVL   P1(AP),R0          ;READ FUNCTION PROCESSING
027E 783          MOVZWL P2(AP),R1        ;GET ADDRESS OF USER BUFFER
0282 784          BEQL   30$              ;GET LENGTH OF USER BUFFER
0284 785          JSB    G^EXE$READCHK    ;IF EQL ZERO LENGTH TRANSFER
028A 786          MOVW   R1,IRP$W_BCNT(R3) ;CHECK ACCESSIBILITY OF USER BUFFER
028E 787          PUSHR  #AM<R0,R3>      ;INSERT LENGTH OF USER BUFFER
0290 788          MOVZBL #80,R1          ;SAVE BUFFER AND I/O PACKET ADDRESSES
0294 789          BBC    #IO$V_BINARY,IRP$W_FUNC(R3),10$ ;SET LENGTH REQUIRED FOR ASCII READ
0299 790          MULL   #2,R1          ;IF CLR, ASCII READ
029C 791          CMPW   R1,IRP$W_BCNT(R3) ;SET LENGTH REQUIRED FOR BINARY READ
02A0 792          BGEQU  20$              ;LENGTH OF READ LARGER THAN USER BUFFER?
02A2 793          MOVW   R1,IRP$W_BCNT(R3) ;IF GEQU YES
02A6 794          ADDL   #12,R1          ;SET LENGTH OF USER BUFFER TO SIZE OF READ
02A9 795          JSB    G^EXE$BUFFERQUOTA ;ACCOUNT FOR BUFFER OVERHEAD
02AF 796          BLBC   R0,40$          ;CHECK IF PROCESS HAS SUFFICIENT QUOTA
02B2 797          JSB    G^EXE$ALLOCBUF   ;IF LBC QUOTA CHECK FAILURE
02B8 798          BLBC   R0,40$          ;ALLOCATE BUFFER FOR CARD READ
02BB 799          POPR   #AM<R0,R3>      ;IF LBC ALLOCATION FAILURE
02BD 800          MOVL   R2,IRP$L_SVAPTE(R3) ;RETRIEVE BUFFER AND I/O PACKET ADDRESSES
02C1 801          MOVW   R1,IRP$W_BOFF(R3) ;INSERT ADDRESS OF READ BUFFER
02C5 802          PUSHL  R0              ;INSERT NUMBER OF QUOTA BYTES CHARGED
02C7 803          MOVL   PCB$L_JIB(R4),R0 ;SAVE BUFFER ADDRESS
02CC 804          SUBL   R1,JIB$L_BYTCNT(R0) ;GET JIB ADDRESS
02D0 805          POPL   R0              ;CHARGE PROCESS FOR BUFFER
02D3 806          MOVAB  12(R2),(R2)+     ;RESTORE BUFFER ADDRESS
02D7 807          MOVL   R0,(R2)         ;INSERT ADDRESS OF DATA AREA
02DA 808          JMP    G^EXE$QIODRVPKT ;SAVE ADDRESS OF USER BUFFER
00000000'GF 17 02DA 808          JMP    G^EXE$QIODRVPKT ;QUEUE DRIVER PACKET
50 6C D0 027B 782          MOVL   P1(AP),R0          ;READ FUNCTION PROCESSING
51 04 AC 3C 027E 783          MOVZWL P2(AP),R1        ;GET ADDRESS OF USER BUFFER
5C 13 0282 784          BEQL   30$              ;GET LENGTH OF USER BUFFER
00000000'GF 16 0284 785          JSB    G^EXE$READCHK    ;IF EQL ZERO LENGTH TRANSFER
32 A3 51 B0 028A 786          MOVW   R1,IRP$W_BCNT(R3) ;CHECK ACCESSIBILITY OF USER BUFFER
09 BB 028E 787          PUSHR  #AM<R0,R3>      ;INSERT LENGTH OF USER BUFFER
51 50 8F 9A 0290 788          MOVZBL #80,R1          ;SAVE BUFFER AND I/O PACKET ADDRESSES
03 20 A3 06 E1 0294 789          BBC    #IO$V_BINARY,IRP$W_FUNC(R3),10$ ;SET LENGTH REQUIRED FOR ASCII READ
51 02 C4 0299 790          MULL   #2,R1          ;IF CLR, ASCII READ
32 A3 51 B1 029C 791          CMPW   R1,IRP$W_BCNT(R3) ;SET LENGTH REQUIRED FOR BINARY READ
04 1E 02A0 792          BGEQU  20$              ;LENGTH OF READ LARGER THAN USER BUFFER?
32 A3 51 B0 02A2 793          MOVW   R1,IRP$W_BCNT(R3) ;IF GEQU YES
51 0C C0 02A6 794          ADDL   #12,R1          ;SET LENGTH OF USER BUFFER TO SIZE OF READ
00000000'GF 16 02A9 795          JSB    G^EXE$BUFFERQUOTA ;ACCOUNT FOR BUFFER OVERHEAD
37 50 E9 02AF 796          BLBC   R0,40$          ;CHECK IF PROCESS HAS SUFFICIENT QUOTA
00000000'GF 16 02B2 797          JSB    G^EXE$ALLOCBUF   ;IF LBC QUOTA CHECK FAILURE
2E 50 E9 02B8 798          BLBC   R0,40$          ;ALLOCATE BUFFER FOR CARD READ
09 BA 02BB 799          POPR   #AM<R0,R3>      ;IF LBC ALLOCATION FAILURE
2C A3 52 D0 02BD 800          MOVL   R2,IRP$L_SVAPTE(R3) ;RETRIEVE BUFFER AND I/O PACKET ADDRESSES
30 A3 51 B0 02C1 801          MOVW   R1,IRP$W_BOFF(R3) ;INSERT ADDRESS OF READ BUFFER
50 DD 02C5 802          PUSHL  R0              ;INSERT NUMBER OF QUOTA BYTES CHARGED
50 0080 C4 D0 02C7 803          MOVL   PCB$L_JIB(R4),R0 ;SAVE BUFFER ADDRESS
20 A0 51 C2 02CC 804          SUBL   R1,JIB$L_BYTCNT(R0) ;GET JIB ADDRESS
50 8ED0 02D0 805          POPL   R0              ;CHARGE PROCESS FOR BUFFER
82 0C A2 9E 02D3 806          MOVAB  12(R2),(R2)+     ;RESTORE BUFFER ADDRESS
62 50 D0 02D7 807          MOVL   R0,(R2)         ;INSERT ADDRESS OF DATA AREA
00000000'GF 17 02DA 808          JMP    G^EXE$QIODRVPKT ;SAVE ADDRESS OF USER BUFFER
00000000'GF 17 02DA 808          JMP    G^EXE$QIODRVPKT ;QUEUE DRIVER PACKET

```

```

      02E0 809
      02E0 810 ;
      02E0 811 ; ZERO LENGTH TRANSFER
      02E0 812 ;
      02E0 813
50 01 3C 02E0 814 30$: MOVZWL #SS$ NORMAL,R0 ;SET NORMAL COMPLETION STATUS
00000000'GF 17 02E3 815 JMP G^EXE$FINISHIOC ;FINISH I/O
      02E9 816
      02E9 817 ;
      02E9 818 ; QUOTA OR BUFFER ALLOCATION FAILURE
      02E9 819 ;
      02E9 820
      0C BA 02E9 821 40$: POPR #^M<R2,R3> ;RETRIEVE I/O PACKET ADDRESS
00000000'GF 17 02EB 822 JMP G^EXE$ABORTIO ;ABORT I/O OPERATION
```

```

02F1 824          .SBTTL  START I/O OPERATION ON CR11 CARD READER
02F1 825 ;+
02F1 826 ; CR_STARTIO - START I/O OPERATION ON CR11 CARD READER
02F1 827 ;
02F1 828 ; THIS ROUTINE IS ENTERED WHEN THE ASSOCIATED UNIT IS IDLE AND A PACKET IS
02F1 829 ; AVAILABLE FOR PROCESSING.
02F1 830 ;
02F1 831 ; INPUTS:
02F1 832 ;
02F1 833 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
02F1 834 ;      R5 = ADDRESS OF DEVICE UNIT UCB.
02F1 835 ;
02F1 836 ; OUTPUTS:
02F1 837 ;
02F1 838 ;      CARD MOTION IS STARTED BY SETTING THE APPROPRIATE FUNCTION BITS IN THE
02F1 839 ;      CONTROL STATUS REGISTER. AS EACH COLUMN INTERRUPT OCCURS, THE DATA FROM
02F1 840 ;      THE DATA BUFFER REGISTER(S) IS STORED IN THE BUFFER ALLOCATED BY THE
02F1 841 ;      FDT ROUTINE. WHEN ALL 80 COLUMNS HAVE BEEN READ, A FORK PROCESS IS CREAT-
02F1 842 ;      ED, THE COLUMN DATA IS CONVERTED ACCORDING TO THE I/O FUNCTION CODE, AND
02F1 843 ;      REQUEST COMPLETE IS CALLED FOR POST PROCESSING.
02F1 844 ;-
02F1 845
02F1 846 CR_STARTIO:                                ;START I/O OPERATION
02F1 847      CMPZV      #IRP$V_FCODE,#IRP$$_FCODE,- ;SET MODE FUNCTION?
02F1 848      IRP$W_FUNC(R3),#IO$_SETMODE ;
02F1 849      BEQL      10$ ;IF EQL YES
02F1 850      CMPZV      #IRP$V_FCODE,#IRP$$_FCODE,- ;SET CHARACTERISTICS FUNCTION?
02F1 851      IRP$W_FUNC(R3),#IO$_SETCHAR ;
02F1 852      BNEQ      20$ ;IF NEQ NO
0301 853
0301 854 ;
0301 855 ; SET READER CHARACTERISTICS
0301 856 ;
0301 857
40 A5 38 A3 B0 0301 858      MOVW      IRP$L_MEDIA(R3),UCB$B_DEVCLASS(R5) ;SET DEVICE CLASS AND TYPE
0306 859
0306 860 ;
0306 861 ; SET READER MODE
0306 862 ;
0306 863
42 A5 3A A3 B0 0306 864 10$:      MOVW      IRP$L_MEDIA+2(R3),UCB$W_DEVBUFSIZ(R5) ;SET DEFAULT BUFFER SIZE
44 A5 3C A3 D0 030B 865      MOVL      IRP$L_MEDIA+4(R3),UCB$L_DEVDEPEND(R5) ;SET DEVICE DEPENDENT FLAGS
0310 866      BRW      140$ ;
0313 867
0313 868 ;
0313 869 ; SET UP PARAMETERS AND READ CARD
0313 870 ;
0313 871
78 A5 2C B3 D0 0313 872 20$:      MOVL      @IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5) ;SET ADDRESS OF BUFFER
0091 C5 01 90 0318 873      MOVVB      #1,UCB$B_CR_EOFcnt(R5) ;SET END OF FILE COUNT FOR ASCII
05 20 A3 06 E1 031D 874      BBC      #IO$V_BINARY,IRP$W_FUNC(R3),30$ ;IF CLR, ASCII READ
0091 C5 08 90 0322 875      MOVVB      #8,UCB$B_CR_EOFcnt(R5) ;SET END OF FILE COUNT FOR BINARY
0092 C5 09 90 0327 876 30$:      MOVVB      UCB$B_CR_EOFcnt(R5),UCB$B_CR_EOFcol(R5) ;SET REQUIRED NUMBER
0090 C5 01 8E 032E 877      MNEGB      #1,UCB$B_CR_COLcnt(R5) ;SET INITIAL COLUMN COUNT
0093 C5 94 0333 878      CLRFB      UCB$B_CR_OFLCnt(R5) ;SET INITIAL OFFLINE COUNT
54 24 A5 D0 0337 879      MOVL      UCB$L_CRB(R5),R4 ;GET ADDRESS OF CRB

```

```

54 2C B4 D0 033B 880      MOVL  @CRB$L_INTD+VEC$L_IDB(R4),R4 ;GET DEVICE CSR ADDRESS
      033F 881 40$:      DSBINT ;IPL^31 ;DISABLE INTERRUPTS
64 0100 8F B3 0345 882      BITW  #CR_CSR_M_OFFLIN,CR_CSR(R4) ;READER OFFLINE?
      5D 13 034A 883      BEQL  70$ ;IF EQL NO
      034C 884      WFIKPCH 50$,#2 ;WAIT FOR TIMEOUT
      0356 885      IOFORK ;CREATE FORK PROCESS
      E1 11 035C 886      BRB  40$ ;
      035E 887 ;
      035E 888 ;
      035E 889 ; READER TIME OUT OR DEVICE ERROR
      035E 890 ;
      035E 891 ;
64 40 8F 9B 035E 892 50$:  MOVZBW #CR_CSR_M_IE,CR_CSR(R4) ;CLEAR READER ERRORS
      0362 893      SETIPL  UCB$B_FIPL(R5) ;LOWER TO DEVICE FORK LEVEL
22 64 A5 03 E0 0366 894      BBS  #UCB$V_CANCEL,UCB$W_STS(R5),60$ ;IF SET, CANCEL I/O REQUESTED
      01 0F 9D 036B 895      ACBB  #15,#1,UCB$B_CR_OFLCNT(R5),40$ ;IF SET, NOT TIME FOR MESSAGE
FFCC 0093 C5 036E ;
      0093 C5 94 0373 896      CLRBR  UCB$B_CR_OFLCNT(R5) ;CLEAR OFFLINE COUNT
      18 BB 0377 897      PUSHR  #^M<R3,R4> ;SAVE REGISTERS
      54 05 9A 0379 898      MOVZBL #MSG$_DEVOFFLIN,R4 ;SET DEVICE MESSAGE NUMBER
00000000 GF 9E 037C 899      MOVAB  G^SYS$GL_OPRMBX,R3 ;GET ADDRESS OF OPERATOR MAILBOX
      53 0382 ;
00000000 GF 16 0383 900      JSB  G^EXE$$NDEVMSG ;SEND MESSAGE TO OPERATOR
      18 BA 0389 901      POPR  #^M<R3,R4> ;RESTORE REGISTERS
      B2 11 038B 902      BRB  40$ ;
      038D 903 ;
      038D 904 ;
      038D 905 ; CANCEL CURRENT READ REQUEST
      038D 906 ;
      038D 907 ;
50 2C 3C 038D 908 60$:  MOVZWL #SS$_ABORT,R0 ;SET ABORT STATUS
      00F3 31 0390 909      BRW  150$ ;
      0393 910 ;
      0393 911 ; DATA OVERFLOW (MORE THAN 80 COL) DETECTED
      0393 912 ;
      0393 913 ;
64 40 8F 9B 0393 914 65$:  MOVZBW #CR_CSR_M_IE,CR_CSR(R4) ;INHIBIT READS
      0397 915      IOFORK ;
      039D 916      INCW  UCB$W_ERRCNT(R5) ;INCREMENT HARDWARE ERROR COUNT
50 0082 C5 B6 03A1 917      MOVZWL #SS$_CTRLERR,R0 ;RETURN HARDWARE ERROR STATUS
      0054 8F 3C 03A1 917      MOVZWL #SS$_CTRLERR,R0 ;RETURN HARDWARE ERROR STATUS
      00DD 31 03A6 918      BRW  150$ ;
      03A9 919 ;
      03A9 920 ;
      03A9 921 ; INITIATE READ
      03A9 922 ;
      03A9 923 ;
      02 A4 B5 03A9 924 70$:  TSTW  CR_CRB1(R4) ;CLEAR COLUMN BUFFER
      41 8F 9B 03AC 925      MOVZBW #CR_CSR_M_IE!CR_CSR_M_READ,- ;ENABLE INTERRUPTS AND START READ
      64 03AF 926 ;
      03B0 927 80$:  WFIKPCH 50$,#3 ;WAITFOR INTERRUPT OR TIMEOUT
64 C000 8F B3 03BA 928      BITW  #CR_CSR_M_CRDONE!CR_CSR_M_ERROR,CR_CSR(R4) ;CARD DONE OR ERROR?
      4A 12 03BF 929      BNEQ  120$ ;IF NEQ YES
      51 02 A4 B0 03C1 930      MOVW  CR_CRB1(R4),R1 ;READ BINARY COLUMN
      52 04 A4 B0 03C5 931      MOVW  CR_CRB2(R4),R2 ;READ PACKED COLUMN
      0090 C5 96 03C9 932      INCB  UCB$B_CR_COLCNT(R5) ;INCREMENT COLUMN COUNT
      05 12 03CD 933      BNEQ  90$ ;IF NEQ NOT FIRST COLUMN
0094 C5 51 B0 03CF 934      MOVW  R1,UCB$W_CR_FSTCOL(R5) ;SAVE FIRST COLUMN BINARY DATA

```

```

0092 C5 91 03D4 935 90$: CMPB UCB$B_CR_EOFCOL(R5),UCB$B_CR_COLCNT(R5) ;PAST END OF FILE DATA?
0090 C5 03D8
51 0F0F 0B 15 03DB 936 BLEQ 100$ ;IF LEQ YES
04 12 03E2 938 CMPW #CR_EOF,R1 ;END OF FILE PUNCH?
0091 C5 97 03E4 939 BNEQ 100$ ;IF NEQ NO
50 8F 91 03E8 940 100$: DECB UCB$B_CR_EOFcnt(R5) ;DECREMENT END OF FILE COUNT
0090 C5 03EB 940 100$: CMPB #80,UCB$B_CR_COLCNT(R5) ;DATA OVERFLOW (MORE THAN 80 COL) ?
A3 1B 03EE 941 BLEQU 65$ ;IF LEQU, YES
78 B5 52 90 03F0 942 MOVB R2,@UCB$L_SVAPTE(R5) ;STORE PACKED COLUMN
07 20 A3 06 E1 03F4 943 BBC #IO$V_BINARY,IRP$W_FUNC(R3),110$ ;IF CLR, ASCII READ
78 B5 51 B0 03F9 944 MOVW R1,@UCB$L_SVAPTE(R5) ;STORE BINARY COLUMN
78 A5 D6 03FD 945 INCL UCB$L_SVAPTE(R5) ;UPDATE BUFFER ADDRESS
78 A5 D6 0400 946 110$: INCL UCB$L_SVAPTE(R5) ;UPDATE BUFFER ADDRESS
A5 11 0403 947 DSBINT ;DISABLE INTERRUPTS
040B 948 BRB 80$ ;
040B 949 ;
040B 950 ;
040B 951 ; SPECIAL CONDITION
040B 952 ;
040B 953 ;
0096 C5 64 B0 040B 954 120$: MOVW CR_CSR(R4),UCB$W_CR_CSR(R5) ;SAVE READER STATUS
64 40 8F 9B 0410 955 MOVZBW #CR_CSR_M_IE,CR_CSR(R4) ;CLEAR READER ERRORS
0414 956 IOFORK ;CREATE FORK PROCESS
50 0870 8F 3C 041A 957 MOVZWL #SS$_ENDOFFILE,R0 ;ASSUME END OF FILE ENCOUNTERED
0091 C5 95 041F 958 TSTB UCB$B_CR_EOFcnt(R5) ;END OF FILE?
61 13 0423 959 BEQL 150$ ;IF EQL YES
50 0838 8F 3C 0425 960 MOVZWL #SS$_DATAOVERUN,R0 ;ASSUME TIMING ERROR
51 0096 C5 3C 042A 961 MOVZWL UCB$W_CR_CSR(R5),R1 ;GET READER STATUS
53 51 0B E0 042F 962 BBS #CR_CSR_V_TIMERR,R1,150$ ;IF SET, TIMING ERROR - EXIT
65 51 0C E0 0433 963 BBS #CR_CSR_V_MCHECK,R1,180$ ;IF SET, MOTION CHECK - RETRY
0437 964 ;
0437 965 ;*** NOTE: SINCE HOPPER CHECK SETS ERROR, A READ CHECK IS NOT DETECTABLE
0437 966 ;*** IF HOPPER CHECK IS ALSO SET, IE. NO READ CHECK RETRIES ON LAST CARD
0437 967 ;
04 51 0D E0 0437 968 BBS #CR_CSR_V_HCCHK,R1,125$ ;IF SET, HOPPER CHECK - OK
5D 51 0F E0 043B 969 BBS #CR_CSR_V_ERROR,R1,180$ ;IF SET, READ CHECK - RETRY
043F 970 125$:
00C0 8F B3 043F 971 BITW #IO$M_BINARY!IO$M_PACKED,IRP$W_FUNC(R3) ;BINARY OR PACKED READ?
20 A3 0443
36 12 0445 972 BNEQ 140$ ;IF NEQ YES
0AAA 8F B1 0447 973 CMPW #CR_029,UCB$W_CR_FSTCOL(R5) ;CHANGE MODE TO 029 TRANSLATION?
0094 C5 044B
3E 13 044E 974 BEQL 160$ ;IF EQL YES
08A2 8F B1 0450 975 CMPW #CR_026,UCB$W_CR_FSTCOL(R5) ;CHANGE MODE TO 026 TRANSLATION?
0094 C5 0454
3D 13 0457 976 BEQL 170$ ;IF EQL YES
50 FBDB CF 9E 0459 977 MOVAB CR_CVT029,R0 ;GET ADDRESS OF 029 TRANSLATION TABLE
04 00 ED 045E 978 CMPZV #CR$V_TMODE,#CR$$_TMODE,- ;029 TRANSLATION MODE?
01 44 A5 0461 979 UCB$L_DEVDEPEND(R5),#CR$K_T029 ;
05 13 0464 980 BEQL 130$ ;IF EQL YES
50 FCCE CF 9E 0466 981 MOVAB CR_CVT026,R0 ;GET ADDRESS OF 026 TRANSLATION TABLE
55 DD 046B 982 130$: PUSHL R5 ;SAVE ADDRESS OF UCB
51 2C B3 D0 046D 983 MOVL @IRP$L_SVAPTE(R3),R1 ;GET ADDRESS OF I/O BUFFER
61 32 A3 2E 0471 984 MOVTC IRP$W_BCNT(R3),(R1),#0,(R0),IRP$W_BCNT(R3),(R1) ;TRANSLATE
60 00 0475
61 32 A3 0477

```

```
55 8ED0 047A 985 POPL R5 ;RETRIEVE ADDRESS OF UCB
50 01 3C 047D 986 140$: MOVZWL #SS$_NORMAL,R0 ;SET NORMAL COMPLETION
10 7E A5 F0 0480 987 INSV UCB$_BCNT(R5),#16,#16,R0 ;INSERT TRANSFER BYTE COUNT
50 10 . 0484
51 D4 0486 988 150$: CLRL R1 ;CLEAR SECOND I/O LONGWORD
0488 989 REQCOM ;COMPLETE REQUEST
048E 990
048E 991 ;
048E 992 ; SET 029 TRANSLATION MODE
048E 993 ;
048E 994
00 01 F0 048E 995 160$: INSV #CR$_T029,#CR$_TMODE,- ;SET 029 TRANSLATION MODE
44 A5 04 0491 996 #CR$_TMODE,UCB$_DEVDEPEND(R5) ;
06 11 0494 997 BRB 180$ ;
0496 998
0496 999 ;
0496 1000 ; SET 026 TRANSLATION MODE
0496 1001 ;
0496 1002
00 00 F0 0496 1003 170$: INSV #CR$_T026,#CR$_TMODE,- ;SET 026 TRANSLATION MODE
44 A5 04 0499 1004 #CR$_TMODE,UCB$_DEVDEPEND(R5) ;
FE74 31 049C 1005 180$: BRW 20$ ;
```

```

049F 1007          .SBTTL  CR11 CARD READER INTDERRUPTS
049F 1008 ;+
049F 1009 ; CR$INT - CR11 CARD READER INTERRUPTS
049F 1010 ;
049F 1011 ; THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN INTERRUPT OCCURS ON A
049F 1012 ; CR11 CARD READER CONTROLLER. THE STATE OF THE STACK ON ENTRY IS:
049F 1013 ;
049F 1014 ;          00(SP) = ADDRESS OF IDB ADDRESS.
049F 1015 ;          04(SP) - 24(SP) = SAVED R0 - R5.
049F 1016 ;          28(SP) = INTERRUPT PC.
049F 1017 ;          32(SP) = INTERRUPT PSL.
049F 1018 ;
049F 1019 ; INTERRUPT DISPATCHING OCCURS AS FOLLOWS:
049F 1020 ;
049F 1021 ;          IF THE INTERRUPT IS EXPECTED, THE DRIVER IS CALLED AT ITS
049F 1022 ;          INTERRUPT RETURN ADDRESS (UCB$L_FPC). IF THE INTERRUPT IS
049F 1023 ;          NOT EXPECTED AND THE DEVICE IS NOT ALLOCATED, A MESSAGE IS
049F 1024 ;          SENT TO THE JOB CONTROLLER TO INFORM IT THAT AN INPUT
049F 1025 ;          SYMBIONT PROCESS SHOULD BE CREATED TO READ THE CARDS.
049F 1026 ;-
049F 1027
049F 1028 CR$INT::
                    ;CARD READER INTERRUPT
53  9E  D0 049F 1029      MOVL  @(SP)+,R3          ;GET ADDRESS OF IDB
54  63  7D 04A2 1030      MOVQ  IDB$L_CSR(R3),R4        ;GET CONTROLLER CSR AND OWNER UCB ADDRESS
11 64  A5 01  E5 04A5 1031      BCC  #UCB$V_INT,UCB$W_STS(R5),10$ ;IF CLR, INTERRUPT NOT EXPECTED
53  10  A5 D0 04AA 1032      MOVL  UCB$L_FR3(R5),R3      ;RESTORE REMAINING DRIVER CONTEXT
                    OC B5 16 04AE 1033      JSB  @UCB$L_FPC(R5)        ;CALL DRIVER
50  8E  7D 04B1 1034      MOVQ  (SP)+,R0          ;RESTORE REGISTERS
52  8E  7D 04B4 1035      MOVQ  (SP)+,R2          ;
54  8E  7D 04B7 1036      MOVQ  (SP)+,R4          ;
                    02 04BA 1037      REI                      ;
                    04BB 1038
                    04BB 1039 ;
                    04BB 1040 ; UNSOLICITED INTERRUPT
                    04BB 1041 ;
                    04BB 1042
                    50  64  3C 04BB 1043 10$:  MOVZWL  CR_CSR(R4),R0          ;GET READER STATUS
64  40  8F 9B 04BE 1044      MOVZBW  #CR_CSR_M_IE,CR_CSR(R4) ;CLEAR STATUS, ENABLE INTERRUPTS
50  0400 8F B3 04C2 1045      BITW  #CR_CSR_M_ONLINE,R0    ;READER TRANSITION TO ONLINE?
                    OC 13 04C7 1046      BEQL  20$                ;IF EQL NO
                    5C  A5  B5 04C9 1047      TSTW  UCB$W_REFC(R5)        ;DEVICE ASSIGNED OR ALLOCATED?
                    07  12 04CC 1048      BNEQ  20$                ;IF NEQ YES
02 68  A5  00  E2 04CE 1049      BBSS  #UCB$V_JOB,UCB$W_DEVSTS(R5),20$ ;IF SET, MESSAGE ALREADY SENT
                    50  8E  7D 04D3 1050      BSBB  30$                ;SEND MESSAGE TO JOB CONTROLLER
                    52  8E  7D 04D5 1051 20$:  MOVQ  (SP)+,R0          ;RESTORE REGISTERS
                    54  8E  7D 04D8 1052      MOVQ  (SP)+,R2          ;
                    54  8E  7D 04DB 1053      MOVQ  (SP)+,R4          ;
                    02 04DE 1054      REI                      ;
                    00000000'GF 16 04DF 1055 30$:  JSB  GAEXE$FORK          ;CREATE FORK PROCESS
                    54  02  9A 04E5 1056      MOVZBL #MSG$_CRUNSOLIC,R4    ;SET MESSAGE TYPE
00000000'GF 9E 04E8 1057      MOVAB  GASYS$GL_JOBCTLMB,R3 ;SET ADDRESS OF JOB CONTROLLER MAILBOX
                    53  04EE
00000000'GF 16 04EF 1058      JSB  GAEXE$SNDEVM$G      ;SENT MESSAGE TO JOB CONTROLLER
                    04 50  E8 04F5 1059      BLBS  R0,40$            ;IF LBS SUCCESSFUL NOTIFICATION
68  A5  01  AA 04F8 1060      BICW  #UCB$M_JOB,UCB$W_DEVSTS(R5) ;CLEAR MESSAGE SENT BIT
                    05 04FC 1061 40$:  RSB                      ;

```



```
04FD 1063 .SBTTL CARD READER INITIALIZATION
04FD 1064 ;+
04FD 1065 ; CR_INITIAL - CR11 CARD READER INITIALIZATION
04FD 1066 ;
04FD 1067 ; THIS ROUTINE IS CALLED AT SYSTEM STARTUP AND AFTER A POWER FAILURE. THE CSR
04FD 1068 ; ADDRESS OF THE RESPECTIVE CR11 CONTROLLER IS READ TO INSURE ITS PRESENCE ON
04FD 1069 ; THE UBA AND THEN CARD READER INTERRUPTS ARE ENABLED.
04FD 1070 ;
04FD 1071 ; INPUTS:
04FD 1072 ;
04FD 1073 ; R4 = CR11 CONTROLLER CSR ADDRESS.
04FD 1074 ; R5 = IDB ADDRESS OF DEVICE UNIT.
04FD 1075 ;
04FD 1076 ; OUTPUTS:
04FD 1077 ;
04FD 1078 ; ALL REGISTERS ARE PRESERVED.
04FD 1079 ;-
04FD 1080
04FD 1081 CR_INITIAL: ;CR11 INITIALIZATION
64 40 8F 9B 04FD 1082 MOVZBW #CR_CSR_M_IE,CR_CSR(R4) ;ENABLE CR11 INTERRUPTS.
05 0501 1083 RSB ;
```

```

0502 1085          .SBTTL  CARD READER UNIT INITIALIZATION
0502 1086 ;+
0502 1087 ; CR_CR11_INIT - CARD READER UNIT INITIALIZATION
0502 1088 ;
0502 1089 ; THIS ROUTINE IS CALLED AT SYSTEM STARTUP AND AFTER A POWER FAILURE. THE
0502 1090 ; ONLINE BIT IS SET IN THE DEVICE UCB.
0502 1091 ;
0502 1092 ; INPUTS:
0502 1093 ;
0502 1094 ;          R5 = ADDRESS OF DEVICE UCB.
0502 1095 ;
0502 1096 ; OUTPUTS:
0502 1097 ;
0502 1098 ;          THE ONLINE BIT IS SET IN THE DEVICE UCB AND THE ADDRESS OF THE UCB
0502 1099 ;          IS FILLED INTO THE OWNER FIELD OF THE IDB.
0502 1100 ; -
0502 1101
0502 1102 CR_CR11_INIT:                                ;CARD READER UNIT INITIALIZATION
64 A5  10  A8 0502 1103          BISW      #UCB$M_ONLINE,UCB$W_STS(R5) ;SET UNIT ONLINE
50  24 A5  D0 0506 1104          MOVL     UCB$L_CRB(R5),R0           ;GET ADDRESS OF CRB
50  2C A0  D0 050A 1105          MOVL     CRB$L_INTD+VEC$L_IDB(R0),R0 ;GET ADDRESS OF IDB
04 A0  55  D0 050E 1106          MOVL     R5,IDB$L_OWNER(R0)       ;SET ADDRESS OF DEVICE UCB
05 0512 1107          RSB
0513 1108 CR_END:
0513 1109
0513 1110          .END

```

CRDRIVER  
Symbol table

- CR11 CARD READER DRIVER

\$\$\$	=	00000020	R	02	EXE\$QIODRVPKT	*****	X	03
\$\$OP	=	00000002			EXE\$READCHK	*****	X	03
AT\$UBA	=	00000001			EXE\$SENSEMODE	*****	X	03
CR\$DDT	=	00000000	RG	03	EXE\$SETMODE	*****	X	03
CR\$INT	=	0000049F	RG	03	EXE\$SNDEVMMSG	*****	X	03
CR\$K_T026	=	00000000			FUNCTAB_LEN	=	00000034	
CR\$K_T029	=	00000001			IDB\$L_CSR	=	00000000	
CR\$S_TMODE	=	00000004			IDB\$L_OWNER	=	00000004	
CR\$V_TMODE	=	00000000			IO\$M_BINARY	=	00000040	
CRB\$L_INTD	=	00000024			IO\$M_PACKED	=	00000080	
CR_026	=	000008A2			IO\$V_BINARY	=	00000006	
CR_029	=	00000AAA			IO\$_READLBLK	=	00000021	
CR_CANCELIO	=	0000026C	R	03	IO\$_READPBLK	=	0000000C	
CR_CR11_INIT	=	00000502	R	03	IO\$_READVBLK	=	00000031	
CR_CRB1	=	00000002			IO\$_SENSECHAR	=	0000001B	
CR_CRB2	=	00000004			IO\$_SENSEMODE	=	00000027	
CR_CSR	=	00000000			IO\$_SETCHAR	=	0000001A	
CR_CSR_M_CRDONE	=	00004000			IO\$_SETMODE	=	00000023	
CR_CSR_M_ERROR	=	00008000			IO\$_VIRTUAL	=	0000003F	
CR_CSR_M_IE	=	00000040			IOC\$CANCELIO	*****	X	03
CR_CSR_M_OFFLIN	=	00000100			IOC\$MNTVER	*****	X	03
CR_CSR_M_ONLINE	=	00000400			IOC\$REQCOM	*****	X	03
CR_CSR_M_READ	=	00000001			IOC\$RETURN	*****	X	03
CR_CSR_V_ERROR	=	0000000F			IOC\$WFIKPC	*****	X	03
CR_CSR_V_HC	=	0000000D			IRP\$L_MEDIA	=	00000038	
CR_CSR_V_MC	=	0000000C			IRP\$L_SVA	=	0000002C	
CR_CSR_V_TMERR	=	0000000B			IRP\$S_FCODE	=	00000006	
CR_CVT026	=	00000138	R	03	IRP\$V_FCODE	=	00000000	
CR_CVT029	=	00000038	R	03	IRP\$W_BCNT	=	00000032	
CR_END	=	00000513	R	03	IRP\$W_BOFF	=	00000030	
CR_EOF	=	00000F0F			IRP\$W_FUNC	=	00000020	
CR_FUNCTABLE	=	00000238	R	03	JIB\$L_BYTCNT	=	00000020	
CR_INITIAL	=	000004FD	R	03	MASKH	=	00000080	
CR_READ	=	0000027B	R	03	MASKL	=	08000000	
CR_STARTIO	=	000002F1	R	03	MSG\$_CRUNSOLIC	=	00000002	
DC\$_CARD	=	00000041			MSG\$_DEVOFFLIN	=	00000005	
DDB\$L_DDT	=	0000000C			P1	=	00000000	
DEV\$M_AVL	*****		X	02	P2	=	00000004	
DEV\$M_IDV	*****		X	02	P3	=	00000008	
DEV\$M_NNM	*****		X	02	P4	=	0000000C	
DEV\$M_REC	*****		X	02	P5	=	00000010	
DPT\$C_LENGTH	=	00000038			P6	=	00000014	
DPT\$C_VERSION	=	00000004			PCB\$L_JIB	=	00000080	
DPT\$INITAB	=	00000038	R	02	PR\$_IPL	=	00000012	
DPT\$REINITAB	=	00000062	R	02	SIZ...	=	00000001	
DPT\$TAB	=	00000000	R	02	SS\$_ABORT	=	0000002C	
DT\$_CR11	=	00000001			SS\$_CTRLERR	=	00000054	
DYN\$C_CRB	=	00000005			SS\$_DATAOVERUN	=	00000838	
DYN\$C_DDB	=	00000006			SS\$_ENDOFFILE	=	00000870	
DYN\$C_DPT	=	0000001E			SS\$_NORMAL	=	00000001	
DYN\$C_UCB	=	00000010			SYS\$GL_JOBCTLMB	*****	X	03
EXE\$ABORTIO	*****		X	03	SYS\$GL_OPRMBX	*****	X	03
EXE\$ALLOCBUF	*****		X	03	UCB\$B_CR_COLCNT	=	00000090	
EXE\$BUFRQUOTA	*****		X	03	UCB\$B_CR_EOF	=	00000091	
EXE\$FINISHIOC	*****		X	03	UCB\$B_CR_EOF	=	00000092	
EXE\$FORK	*****		X	03	UCB\$B_CR_OF	=	00000093	
EXE\$IOFORK	*****		X	03	UCB\$B_DEVCLASS	=	00000040	

```
UCB$B_DEVTYPE = 00000041
UCB$B_DIPL    = 0000005E
UCB$B_FIPL    = 00000008
UCB$K_CR_LENGTH = 00000098
UCB$K_LENGTH  = 00000090
UCB$L_CRB     = 00000024
UCB$L_DEVCHAR = 00000038
UCB$L_DEVCHAR2 = 0000003C
UCB$L_DEVDEPEND = 00000044
UCB$L_FPC     = 0000000C
UCB$L_FR3    = 00000010
UCB$L_SVAPTE = 00000078
UCB$M_JOB    = 00000001
UCB$M_ONLINE = 00000010
UCB$V_CANCEL = 00000003
UCB$V_INT    = 00000001
UCB$V_JOB    = 00000000
UCB$W_BCNT   = 0000007E
UCB$W_CR_CSR = 00000096
UCB$W_CR_FSTCOL = 00000094
UCB$W_DEVBUFSIZ = 00000042
UCB$W_DEVSTS = 00000068
UCB$W_ERRCNT = 00000082
UCB$W_REFC   = 0000005C
UCB$W_STS    = 00000064
VEC$L_IDB    = 00000008
VEC$L_INITIAL = 0000000C
VEC$L_UNITINIT = 00000018
```

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000098 ( 152.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	00000077 ( 119.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00000513 ( 1299.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.08	00:00:02.17
Command processing	97	00:00:00.51	00:00:06.20
Pass 1	524	00:00:23.55	00:01:16.65
Symbol table sort	0	00:00:03.25	00:00:10.16
Pass 2	199	00:00:04.98	00:00:17.35
Symbol table output	16	00:00:00.14	00:00:00.71
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	877	00:00:32.54	00:01:53.26

The working set limit was 1800 pages.

123849 bytes (242 pages) of virtual memory were used to buffer the intermediate code.  
There were 110 pages of symbol table space allocated to hold 2103 non-local and 29 local symbols.  
1119 source lines were read in Pass 1, producing 18 object records in Pass 2.  
36 pages of virtual memory were used to define 33 macros.

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

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	20
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	31

2350 GETS were required to define 31 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:CRDRIVER/OBJ=OBJ\$:CRDRIVER MSRC\$:CRDRIVER/UPDATE=(ENH\$:CRDRIVER)+EXECML\$/LIB



**SYSQIOREQ**

(1)	76	DECLARATIONS
(1)	131.32	QIO ERROR AND EXCEPTION HANDLING ROUTINES
(1)	165	QUEUE I/O REQUEST
(1)	583	BUILD I/O PACKET FOR PAGE READ/WRITE
(1)	701	COMPLETE I/O OPERATION
(1)	752	QUEUE I/O PACKET TO DRIVER
(1)	775	EXE\$ALTQUEPKT - Call driver ALTSTART entry point
(1)	810	QUEUE I/O PACKET TO ACP
(1)	845.10	EXE\$QXQPPKT - QUEUE I/O PACKET TO XQP
(1)	847	INSERT I/O PACKET IN UNIT QUEUE
(1)	870	INSERT I/O PACKET IN QUEUE BY PRIORITY



```
;LMP0221
-1
0000 1 .TITLE SYSQIOREQ - QUEUE I/O REQUEST SYSTEM SERVICE
0000 .1 .IDENT 'V03-012'
0000 3
0000 4 ;
0000 5 ;*****
0000 6 ;*
0000 7 ;* COPYRIGHT (c) 1978, 1980, 1982 BY
0000 8 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 ;* ALL RIGHTS RESERVED.
0000 10 ;*
0000 11 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 ;* TRANSFERRED.
0000 17 ;*
0000 18 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 ;* CORPORATION.
0000 21 ;*
0000 22 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 ;*
0000 25 ;*
0000 26 ;*****
0000 27 ;
0000 28 ;++
0000 29 ;
0000 30 ; AUTHOR:
0000 31 ;
0000 32 ; D. N. CUTLER, 14-JUN-76
0000 33 ;
0000 34 ; FACILITY:
0000 35 ;
0000 36 ; SYSTEM SERVICE QUEUE I/O REQUEST
0000 37 ;
0000 38 ; MODIFIED BY:
0000 39 ;
;LMP0221 0000 .1 ; V03-012 LMP0221 L. Mark Pilant, 30-Mar-1984 17:08
;LMP0221 0000 .2 ; Remove references to UCB$L_OWNUIC and UCNB$V_VPROT.
;LMP0221 0000 .3 ;
;SRB0118 0000 .4 ; V03-011 SRB0118 Steve Beckhardt 23-Mar-1984
;SRB0118 0000 .5 ; Changed waiting for DIOCNT or BIOCNT to wait at IPL 0.
;SRB0118 0000 .6 ;
;CDS0005 0000 .7 ; V03-010 CDS0005 Christian D. Saether 20-Mar-1984
;CDS0005 0000 .8 ; Use symbolic definition to locate XQP queue header.
;CDS0005 0000 .9 ; Give the XQP ast a priority boost.
;CDS0005 0000 .10 ;
;SSA0017 0000 .11 ; V03-009 SSA0017 Stan Amway 9-Mar-1984
;SSA0017 0000 .12 ; Maintain device queue length in UCB.
;SSA0017 0000 .13 ; Efficiently supports MONITORs disk class and provides
;SSA0017 0000 .14 ; accurate queue lengths for HSC and UDA disks.
;SSA0017 0000 .15 ;
;LMP0206 0000 .16 ; V03-008 LMP0206 L. Mark Pilant, 7-Mar-1984 12:20
;LMP0206 0000 .17 ; Only do an access check on the first QIO to the channel for
;LMP0206 0000 .18 ; logical and physical requests.
```

```
;LMP0206          0000 .19 ;
;LMP0185          0000 .20 ;      V03-007 LMP0185          L. Mark Pilant,          27-Jan-1984  11:05
;LMP0185          0000 .21 ;      Add support for ACLs on devices.
;LMP0185          0000 .22 ;
;CDS0004          0000 .23 ;      V03-006 CDS0004          Christian D. Saether    20-May-1983
;CDS0004          0000 .24 ;      Get the PID from the PCB instead of the IRP so that
;CDS0004          0000 .25 ;      journalling works (it uses the irp pid field for
;CDS0004          0000 .26 ;      something else).
;CDS0004          0000 .27 ;
;RLRMXBCNTC      0000 .28 ;      V03-005 RLRMXBCNTc      Robert L. Rappaport    28-Mar-1983
;RLRMXBCNTC      0000 .29 ;      Verify IRP$L_DIAGBUF is non-zero before assuming that it
;RLRMXBCNTC      0000 .30 ;      contains the original value of IRP$L_SVAPTE in VIRTUAL_LOGIO.
;RLRMXBCNTC      0000 .31 ;
;CDS0003          0000 .32 ;      V03-004 CDS0003          Christian D. Saether    14-Mar-1983
;CDS0003          0000 .33 ;      Return from EXE$QXQPPKT with status, rather than
;CDS0003          0000 .34 ;      assuming success.
;CDS0003          0000 .35 ;
;CDS0002          0000 .36 ;      V03-003 CDS0002          Christian D. Saether    12-Mar-1983
;CDS0002          0000 .37 ;      Do not insque packet to xqp work queue in EXE$QXQPPKT,
;CDS0002          0000 .38 ;      but rather pass it as the ast parameter and queue it
;CDS0002          0000 .39 ;      in the xqp, if necessary. This avoids a problem
;CDS0002          0000 .40 ;      where the packet is processed by the xqp before the
;CDS0002          0000 .41 ;      ast is dequeued.
;CDS0002          0000 .42 ;
;CDS0001          0000 .43 ;      V03-002 CDS0001          C Saether              13-Aug-1982
;CDS0001          0000 .44 ;      Changes to send file system packets to XQP.
;CDS0001          0000 .45 ;
;LJK0172         0000 .46 ;      V03-001 LJK0172          Lawrence J. Kenah       18-June-1982
;LJK0172         0000 .47 ;      Count I/O operations in EXE$BLDPKTxxxx to allow file
;LJK0172         0000 .48 ;      expiration to work correctly for mapped files.
-33              0000 73 ;
                0000 74 ;--
```

```
0000 76          .SBTTL  DECLARATIONS
0000 77
0000 78 ;
0000 79 ; MACRO LIBRARY CALLS
0000 80 ;
0000 81
0000 82          $ACBDEF          ;DEFINE ACB OFFSETS
0000 83          $AQBDEF          ;DEFINE AQB OFFSETS
0000 84          $CADEF           ;DEFINE CONDITIONAL ASSEMBLY PARAMETERS
0000 85          $CCBDEF          ;DEFINE CCB OFFSETS
;CDS0001 0000 .1          $CDRPDEF        ;DEFINE CDRP OFFSETS
;CDS0005 0000 .2          $ddbDEF        ;DEFINE DDB OFFSETS
;CDS0005 0000 .3          $DDTDEF        ;DEFINE DDT OFFSETS
;CDS0005 0000 .4          $DEVDEF        ;DEFINE DEV VALUES
;CDS0005 0000 .5          $DYNDEF        ;DEFINE DATA STRUCTURE TYPE CODES
-4      0000 .6          $F11BDEF       ;DEFINE F11BXQP OFFSETS
0000 90          $IODEF           ;DEFINE IO FUNCTION CODES
0000 91          $IPLDEF          ;DEFINE INTERRUPT PRIORITY LEVELS
0000 92          $IRPDEF          ;DEFINE IRP OFFSETS
0000 93          $PCBDEF          ;DEFINE PCB OFFSETS
0000 94          $PHDDEF          ;DEFINE PHD OFFSETS
0000 95          $PRDEF           ;DEFINE PROCESSOR REGISTERS
0000 96          $PRIDEF          ;DEFINE PRIORITY CLASS INCREMENTS
0000 97          $PRVDEF          ;DEFINE PRIVILEGE BITS
0000 98          $PSLDEF          ;DEFINE PROCESSOR STATUS FIELDS
0000 99          $RSNDEF          ;DEFINE RESOURCE WAIT NUMBERS
0000 100         $SECDDEF         ;DEFINE SEC OFFSETS
0000 101         $SSDEF           ;DEFINE STATUS VALUES
0000 102         $UCBDEF          ;DEFINE UCB OFFSETS
0000 103         $VCBDEF          ;DEFINE VCB OFFSETS
0000 104         $WCBDEF          ;DEFINE WINDOW CONTROL BLOCK OFFSETS
0000 105
0000 106 ;
0000 107 ; LOCAL SYMBOLS
0000 108 ;
0000 109 ; ARGUMENT LIST OFFSET DEFINITIONS
0000 110 ;
0000 111
00000004 0000 112 EFN=4          ;EVENT FLAG NUMBER
00000008 0000 113 CHAN=8        ;I/O CHANNEL NUMBER
0000000C 0000 114 FUNC=12       ;I/O FUNCTION CODE
00000010 0000 115 IOSB=16      ;ADDRESS OF I/O STATUS BLOCK
00000014 0000 116 ASTADR=20    ;ADDRESS OF AST SERVICE ROUTINE
00000018 0000 117 ASTPRM=24    ;AST SERVICE ROUTINE PARAMETER
0000001C 0000 118 P1=28        ;FIRST FUNCTION DEPENDENT PARAMETER
00000020 0000 119 P2=32        ;SECOND FUNCTION DEPENDENT PARAMETER
00000024 0000 120 P3=36        ;THIRD FUNCTION DEPENDENT PARAMETER
00000028 0000 121 P4=40        ;FOURTH FUNCTION DEPENDENT PARAMETER
0000002C 0000 122 P5=44        ;FIFTH FUNCTION DEPENDENT PARAMETER
00000030 0000 123 P6=48        ;SIXTH FUNCTION DEPENDENT PARAMETER
0000 124
0000 125 ;
0000 126 ; FUNCTION DECISION TABLE OFFSET DEFINITIONS
0000 127 ;
0000 128
00000000 0000 129 LEGAL=0      ;LEGAL FUNCTION MASK
00000008 0000 130 IOTYPE=8    ;I/O FUNCTION TYPE MASK
```

```

00000010 0000 131 FDTACT=16 ;ACTION ROUTINE MASKS
;LMP0185 0000 .1
;LMP0185 0000 .2 ;
;LMP0185 0000 .3 ; TABLES FOR DETERMINING THE ACCESS DESIRED, BASED UPON THE I/O FUNCTION
;LMP0185 0000 .4 ; CODE. THIS IS NECESSARY FOR THE FIRST TIME THROUGH PROTECTION CHECK DONE
;LMP0185 0000 .5 ; ON SHARABLE, NON-MOUNTABLE (NON-FILES ORIENTED) DEVICES.
;LMP0185 0000 .6 ;
;LMP0185 0000 .7
;LMP0185 0000 .8 .MACRO ACCMSK CODES
;LMP0185 0000 .9 MASKL = 0
;LMP0185 0000 .10 MASKH = 0
;LMP0185 0000 .11
;LMP0185 0000 .12 .IRP X,<CODES>
;LMP0185 0000 .13 .IF GT <IO$_'X&IO$_VIRTUAL>-31
;LMP0185 0000 .14 MASKH = MASKH!<1@<<IO$_'X&IO$_VIRTUAL>-32>>
;LMP0185 0000 .15 .IFF
;LMP0185 0000 .16 MASKL = MASKL!<1@<IO$_'X&IO$_VIRTUAL>>
;LMP0185 0000 .17 .ENDC; GT <IO$_'X&IO$_VIRTUAL>-31
;LMP0185 0000 .18 .ENDR ; X,<CODES>
;LMP0185 0000 .19 .LONG MASKL,MASKH
;LMP0185 0000 .20 .ENDM ACCMSK
;LMP0185 0000 .21
;LMP0185 0000 .22 READ_ACCESS:
;LMP0185 0000 .23 ACCMSK <READPBLK,READLBLK,READVBLK,-
;LMP0185 0000 .24 READHEAD,READTRACKD,REREADN,REREADP,-
;LMP0185 0000 .25 READPROMPT,TTYREADALL,TTYREADPALL>
;LMP0185 0008 .26
;LMP0185 0008 .27 WRITE_ACCESS:
;LMP0185 0008 .28 ACCMSK <WRITEPBLK,WRITELBLK,WRITEVBLK,-
;LMP0185 0008 .29 WRITECHECK,WRITECHECKH,-
;LMP0185 0008 .30 WRITEHEAD,WRITETRACKD,WRITERET>

```

```

;LMP0185          0010 .32          .SBTTL QIO ERROR AND EXCEPTION HANDLING ROUTINES
;LMP0185          0010 .33 ;
;LMP0185          0010 .34 ; Device is not mountable. See if it is necessary to do the protection check.
;LMP0185          0010 .35 ;
;LMP0185          0010 .36 NOT_FILE_DEV:
;LMP0185          1E 08 A6 02 E2 0010 .37          BBSS      #CCB$V_PROCHKDON,CCB$B_STS(R6),20$          ;XFER IF HERE ALREADY
;LMP0185          19 38 A5 10 E1 0015 .38          BBC       #DEV$V_SHR,UCB$L_DEVCHAR(R5),20$          ;XFER IF NOT SHARABLE
;LMP0185          50 0000'CF 9E 001A .39          MOVAB     W^EXE$CHKRDACCES,R0          ;SET ADDR OF CHECKING ROUTINE
;LMP0185          0A DD AF 57 E0 001F .40          BBS       R7,READ_ACCESS,10$          ;XFER IF A READ FUNCTION
;LMP0185          50 0000'CF 9E 0024 .41          MOVAB     W^EXE$CHKWRTACCES,R0          ;SET ADDR OF CHECKING ROUTINE
;LMP0185          05 DB AF 57 E1 0029 .42          BBC       R7,WRITE_ACCESS,20$          ;XFER IF NOT WRITE FUNCTION
;LMP0185          60 16 002E .43 10$: JSB       (R0)          ;DO PROTECTION CHECK
;LMP0185          12 50 E9 0030 .44          BLBC     R0,ERRORB          ;XFER IF NOT SUCCESSFUL
;LMP0185          0092 31 0033 .45 20$: BRW      CHKDON          ;ELSE REJOIN MAINLINE CODE
-2
          0036 134 ;
          0036 135 ; MISCELLANEOUS ERROR HANDLING AND EXCEPTION HANDLING ROUTINES. THESE HAVE
          0036 136 ; BEEN MOVED OUT OF LINE TO MAKE THE COMMON PATH NEARLY BRANCH FREE.
          0036 137 ;
          0036 138
          0036 139 CLREF:
          FFC7' 30 0036 140          BSBW      SCH$CLREF          ;CLEAR SPECIFIED EVENT FLAG
          40 11 0039 141          BRB       VCHAN          ;CONTINUE WITH QIO
50 013C 8F 3C 003B 142 IVCHAN: MOVZWL #SS$ IVCHAN,R0          ;SET ERROR STATUS
          03 11 0040 143          BRB       ERRORB          ;AND ERROR REQUEST
          50 24 3C 0042 144 PRIVERR: MOVZWL #SS$ NOPRIV,R0          ;SET ERROR STATUS
          OODF 31 0045 145 ERRORB: BRW      ERROR          ;AND ERROR REQUEST
          0048 146
          0048 147 ;
          0048 148 ; An access or deaccess operation is pending for this channel. Wait for
          0048 149 ; it to complete, then retry the QIO.
          0048 150 ;
          0048 151
          0048 152 DACSPND:SETIPL #IPL$ SYNCH          ;SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
          4C A4 01 3C 004B 153          MOVZWL #RSN$ ASTWAIT,PCB$L_EFWM(R4) ;SET AST WAIT RESOURCE NUMBER
52 0000'CF 7E 004F 154          MOVAQ     W^SCH$GQ_MWAIT,R2          ;SET ADDRESS OF WAIT QUEUE
          4C A4 E2 0054 155          BBSS     PCB$L_EFWM(R4),W^SCH$GL_RESMASK,10$ ;SET WAITING FLAG
          00 0000'CF 0057
          FFA2' 31 005B 156 10$: BRW      SCH$WAIT          ;WAIT FOR AST
          005E 157 ;
          005E 158 ; Device is marked spooled. Acquire intermediate UCB address if virtual funtion.
          005E 159 ;
          57 2F D1 005E 160 SPOOL: CMPL     S^#IO$ LOGICAL,R7          ;VIRTUAL I/O FUNCTION?
          60 18 0061 161          BGEQ     NSPOOL          ;IF GEQ NO
          55 60 A5 D0 0063 162          MOVL     UCB$L_AMB(R5),R5          ;GET INTERMEDIATE DEVICE UCB ADDRESS
          5A 11 0067 163          BRB       NSPOOL          ;
;LMP0185          0069 .1 ;
;LMP0185          0069 .2 ; Intermediate branch to the protection checking routine.
;LMP0185          0069 .3 ;
;LMP0185          0069 .4 NOT_FILE_DEVB:
;LMP0185          A5 11 0069 .5          BRB       NOT_FILE_DEV

```

```

006B 165 .SBTTL QUEUE I/O REQUEST
006B 166 ;+
006B 167 ; EXE$QIOREQ - QUEUE I/O REQUEST
006B 168 ;
006B 169 ; THIS SERVICE PROVIDES THE CAPABILITY TO INITIATE AN I/O OPERATION
006B 170 ; BY QUEUEING A REQUEST TO A DEVICE'S ASSOCIATED DRIVER. ONCE THE
006B 171 ; OPERATION HAS BEEN INITIATED, CONTROL WILL RETURN TO THE CALLER
006B 172 ; WHO CAN SYNCHRONIZE I/O COMPLETION IN ONE OF THREE WAYS:
006B 173 ;
006B 174 ; 1) SPECIFY THE ADDRESS OF AN AST ROUTINE THAT WILL BE
006B 175 ; EXECUTED WHEN THE I/O COMPLETES.
006B 176 ;
006B 177 ; 2) WAIT FOR THE SPECIFIED EVENT FLAG TO BE SET.
006B 178 ;
006B 179 ; 3) POLL THE SPECIFIED I/O STATUS BLOCK FOR A COMPLETION
006B 180 ; STATUS.
006B 181 ;
006B 182 ; THIS ROUTINE VERIFIES THE FUNCTION INDEPENDENT PARAMETERS, ALLOCATES
006B 183 ; AN I/O REQUEST PACKET, COPIES THE FUNCTION INDEPENDENT PARAMETERS AND
006B 184 ; PROCESS INFORMATION TO THE I/O PACKET, CHECKS ACCESS TO THE DEVICE,
006B 185 ; AND CALLS THE DRIVER'S FUNCTION DECISION TABLE ROUTINE(S) THAT CORRESPOND
006B 186 ; TO THE SPECIFIED FUNCTION. IT IS THEN UP TO THE FDT ROUTINE TO EITHER
006B 187 ; COMPLETE THE REQUEST IMMEDIATELY (EXE$ABORTIO OR EXE$FINISHIO) OR TO
006B 188 ; QUEUE THE I/O REQUEST FOR FURTHER PROCESSING BY THE DRIVER'S STARTIO
006B 189 ; ROUTINE (EXE$QIODRVPKT).
006B 190 ;
006B 191 ; INPUTS:
006B 192 ;
006B 193 ; EFN(AP) = EVENT FLAG NUMBER.
006B 194 ; CHAN(AP) = I/O CHANNEL NUMBER.
006B 195 ; FUNC(AP) = I/O FUNCTION CODE.
006B 196 ; IOSB(AP) = ADDRESS OF I/O STATUS BLOCK.
006B 197 ; ASTADR(AP) = ADDRESS OF AST SERVICE ROUTINE.
006B 198 ; ASTPRM(AP) = AST SERVICE ROUTINE PARAMETER.
006B 199 ; P1(AP) TO P6(AP) = FUNCTION DEPENDENT PARAMETERS.
006B 200 ;
006B 201 ; R4 = CURRENT PROCESS PCB ADDRESS.
006B 202 ;
006B 203 ; OUTPUTS:
006B 204 ;
006B 205 ; R0 LOW BIT CLEAR INDICATES FAILURE TO INITIATE THE I/O REQUEST.
006B 206 ;
006B 207 ; R0 = SS$_ABORT - A NETWORK LOGICAL LINK WAS BROKEN.
006B 208 ;
006B 209 ; R0 = SS$_ACCVIO - THE I/O STATUS BLOCK CANNOT BE WRITTEN BY
006B 210 ; THE CALLER.
006B 211 ;
006B 212 ; R0 = SS$_DEVOFFLINE - THE SPECIFIED DEVICE IS OFFLINE.
006B 213 ;
006B 214 ; R0 = SS$_EXQUOTA - THE PROCESS HAS EXCEEDED ITS BUFFERED I/O
006B 215 ; QUOTA, DIRECT I/O QUOTA, OR BUFFERED I/O BYTE COUNT
006B 216 ; QUOTA AND HAS DISABLED RESOURCE WAIT MODE. OR, THE
006B 217 ; PROCESS HAS EXCEEDED ITS AST LIMIT QUOTA.
006B 218 ;
006B 219 ; R0 = SS$_ILLEFC - AN ILLEGAL EVENT FLAG NUMBER WAS SPECIFIED.
006B 220 ;
006B 221 ; R0 = SS$_INSMEM - INSUFFICIENT DYNAMIC MEMORY IS AVAILABLE

```

```

006B 222 ;
006B 223 ;
006B 224 ;
006B 225 ;
006B 226 ;
006B 227 ;
006B 228 ;
006B 229 ;
006B 230 ;
006B 231 ;
006B 232 ;
006B 233 ;
006B 234 ;
006B 235 ;
006B 236 ;
006B 237 ;
006B 238
OFFC 006B 239
006D 240 ;
006D 241 ;
006D 242 ;
006D 243
53 04 AC 9A 006D 244 QIO: MOVZBL EFN(AP),R3 ;GET EVENT FLAG NUMBER
3F 53 91 0071 245 CMPB R3,#63 ;CHECK FOR LOCAL
CO 1A 0074 246 BGTRU CLREF ;IF NO. MUST DO FULL CLREF
00 50 A4 53 E5 0076 247 BCCC R3,PCB$L_EFCS(R4),VCHAN ;CLEAR SPECIFIED EVENT FLAG
007B 248
007B 249 ;
007B 250 ; Validate channel number, compute CCB address and acquire UCB address.
007B 251 ;
007B 252
FFFF000F 8F CB 007B 253 VCHAN: BICL3 #<^FFFFFF0000!<CCB$C_LENGTH-1>>,-;FETCH CHANNEL NUMBER AND
50 08 AC 0081 254 CHAN(AP),R0 ;CLEAR EXTRANEIOUS BITS
B5 13 0084 255 BEQL IVCHAN ;IF EQL INVALID CHANNEL
50 B1 0086 256 CMPW R0,@#CTL$GW_CHINDX ;LEGAL CHANNEL NUMBER?
00000000'9F 0088
AC 1A 008D 257 BGTRU IVCHAN ;IF GTRU NO
59 50 CE 008F 258 MNEGL R0,R9 ;CONVERT TO CHANNEL INDEX
00000000'FF49 9E 0092 259 MOVAB @CTL$GL_CCBASE[R9],R6 ;GET ADDRESS OF CORRESPONDING CCB
56 0099
53 DC 009A 260 MOVPSL R3 ;READ CURRENT PSL
53 02 16 EF 009C 261 EXTZV #PSL$V_PRVMOD,#PSL$$PRVMOD,R3,R11 ;EXTRACT PREVIOUS MODE FIELD
5B 00A0
59 59 10 78 00A1 262 ASHL #16,R9,R9 ;PREPARE CHANNEL INDEX FOR LATER MERGE
09 A6 5B 91 00A5 263 CMPB R11,CCB$B_AMOD(R6) ;CALLER HAVE PRIVILEGE TO ACCESS CHANNEL?
97 18 00A9 264 BGEQ PRIVERR ;IF GEQ NO
55 66 DO 00AB 265 MOVL CCB$L_UCB(R6),R5 ;GET ASSIGNED DEVICE UCB ADDRESS
96 04 A6 E8 00AE 266 BLBS CCB$L_WIND(R6),DACSPND ;IF LBS ACCESS/DEACCESS PENDING
00B2 267
00B2 268 ;
00B2 269 ; Isolate function code and begin decoding
00B2 270 ;
00B2 271
5A 0C AC 3C 00B2 272 MOVZWL FUNC(AP),R10 ;GET I/O FUNCTION CODE AND MODIFIERS
FFFFFC0 8F CB 00B6 273 BICL3 #AC<IO$M_FCODE>,R10,R7 ;CLEAR ALL BUT I/O FUNCTION CODE
57 5A 00BC
9B 38 A5 06 E0 00BE 274 BBS S^#DEV$V_SPL,UCB$L_DEVCHAR(R5),SPOOL ;IF SET, DEVICE IS SPOOLED

```

TO ALLOCATE AN I/O REQUEST PACKET AND THE PROCESS HAS  
DISABLED RESOURCE WAIT MODE.

RO = SS\$\_IVCHAN - AN INVALID CHANNEL NUMBER WAS SPECIFIED.

RO = SS\$\_NOPRIV - THE SPECIFIED CHANNEL DOES NOT EXIST OR WAS  
ASSIGNED FROM A MORE PRIVILEGED ACCESS MODE. OR, THE  
PROCESS DOES NOT HAVE THE PRIVILEGE TO PERFORM THE  
SPECIFIED TYPE OF I/O FUNCTION ON THE DEVICE.

RO = SS\$\_UNASEFC - UNASSOCIATED EVENT FLAG CLUSTER SPECIFIED.

RO LOW BIT SET INDICATES SUCCESSFUL COMPLETION.

RO = SS\$\_NORMAL - NORMAL COMPLETION.

.ENTRY EXE\$QIO,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>

Clear specified event flag. For local event flags, this is done in line.

```

00C3 275
00C3 276 ;
00C3 277 ; Acquire FDT address.
00C3 278 ;
00C3 279
;LMP0185 A1 38 A5 0E E1 00C3 .1 NSPOOL: BBC #DEV$V_FOD,UCB$L_DEVCHAR(R5),NOT_FILE_DEVB ;XFER IF NOT MOUNTAB
;LMP0185 50 0088 C5 D0 00C8 .2 CHKDON: MOVL UCB$L_DDT(R5),R0 ;GET ADDRESS OF DDT
-1 58 08 A0 D0 00CD 281 MOVL DDT$L_FDT(R0),R8 ;GET ADDRESS OF FDT
37 68 57 E1 00D1 282 BBC R7,LEGAL(R8),ILLIO ;IF CLR, ILLEGAL I/O FUNCTION
39 64 A5 04 E1 00D5 283 BBC #UCB$V_ONLINE,UCB$W_STS(R5),OFFLINE ;IF CLR, DEVICE OFFLINE
00DA 284
00DA 285 ;
00DA 286 ; Probe and clear IOSB if it is specified.
00DA 287 ;
00DA 288
51 10 AC D0 00DA 289 PRIOSB: MOVL IOSB(AP),R1 ;GET ADDRESS OF I/O STATUS BLOCK
08 13 00DE 290 BEQL NOIOSB ;IF EQL NONE SPECIFIED
00E0 291 IFNOWRT #8,(R1),ACCVIO ;CAN I/O STATUS BLOCK BE WRITTEN?
61 7C 00E6 292 CLRQ (R1) ;CLEAR I/O STATUS BLOCK
00E8 293
00E8 294 ;
00E8 295 ; Charge appropriate I/O counts depending upon type. Counts will have to
00E8 296 ; be backed out if no I/O packet is available. Set IPL to block process
00E8 297 ; deletion once we are committed.
00E8 298 ;
00E8 299
00E8 300 NOIOSB: SETIPL #IPL$ASTDEL ;PREVENT PROCESS DELETION
7B 08 A8 57 E1 00EB 301 BBC R7,IOTYPE(R8),DIRECT ;IF CLR, DIRECT I/O FUNCTION
00F0 302 ASSUME IRP$M_BUFIO EQ 1 ;TO ALLOW INCREMENT BELOW
59 B6 00F0 303 INCW R9 ;SET IRP$M_BUFIO
3A A4 B7 00F2 304 DECW PCB$W_BIOCNT(R4) ;CHARGE FOR ANOTHER BUFFERED I/O
79 18 00F5 305 BGEQ OK ;OK IF NOT NEGATIVE
3A A4 3F 00F7 306 PUSHAW PCB$W_BIOCNT(R4) ;SET ADDRESS OF QUOTA CELL
52 6E D0 00FA 307 NOCNT: MOVL (SP),R2 ;FETCH QUOTA ADDRESS
62 B6 00FD 308 INCW (R2) ;BACKOUT CHARGE
;SRB0118 00FF .1 SETIPL #0 ;LOWER IPL TO WAIT AT IPL 0
FEFB' 30 0102 309 BSBW EXE$SNGLEQUOTA ;CHECK UNIT QUOTA OF I/O FUNCTION TYPE
1F 50 E9 0105 310 BLBC R0,ERROR ;IF LBC QUOTA EXCEEDED
9E B7 0108 311 DECW @(SP)+ ;CHARGE FOR I/O OF TYPE
64 11 010A 312 BRB OK ;
010C 313
50 00F4 8F 3C 010C 314 ILLIO: MOVZWL #SS$_ILLIOFUNC,R0 ;SET ILLEGAL I/O FUNCTION STATUS
14 11 0111 315 BRB ERROR ;
0113 316
57 34 91 0113 317 OFFLINE: CMPB #IO$_DEACCESS,R7 ;CHECK FOR DEACCESS I/O FUNCTION
C2 13 0116 318 BEQL PRIOSB ;ALLOW IT TO PROCEED
57 38 91 0118 319 CMPB #IO$_ACPCONTROL,R7 ;LIKewise FOR ACP CONTROL
BD 13 011B 320 BEQL PRIOSB ;SO THAT A FILE ON AN OFFLINE DEVICE
011D 321 ;MAY BE CLOSED
50 0084 8F 3C 011D 322 MOVZWL #SS$_DEVOFFLINE,R0 ;SET DEVICE OFFLINE STATUS
03 11 0122 323 BRB ERROR ;
0124 324
50 0C 3C 0124 325 ACCVIO: MOVZWL S^#SS$_ACCVIO,R0 ;SET ACCESS VIOLATION STATUS
0127 326 ERROR: SETIPL #0 ;ALLOW INTERRUPTS
50 DD 012A 327 PUSHL R0 ;SAVE FINAL STATUS
51 60 A4 D0 012C 328 MOVL PCB$L_PID(R4),R1 ;GET PROCESS ID OF CURRENT PROCESS
52 D4 0130 329 CLRL R2 ;SET PRIORITY CLASS INCREMENT

```



```

53  04 AC  9A 0132 330      MOVZBL  EFN(AP),R3          ;GET SPECIFIED EVENT FLAG NUMBER
      FEC7 30 0136 331      BSBW    SCH$POSTEF        ;POST SPECIFIED EVENT FLAG
      01  BA 0139 332      POPR    #AM<R0>          ;RESTORE FINAL STATUS
      04  013B 333      RET                                ;
      013C 334
      013C 335 ;
      013C 336 ; ALLOCATE REQUEST I/O PACKET - WHEN THE LOOKASIDE LIST IS EMPTY.
      013C 337 ;
      013C 338
      FEC1' 30 013C 339 ALLOC: BSBW    EXE$ALLOCIRP        ;ALLOCATE I/O REQUEST PACKET
35 50  E8 013F 340      BLBS    R0,SUCCESS        ;IF LBS SUCCESSFUL ALLOCATION
3A A4  3F 0142 341      PUSHAW  PCB$W_BIOCNT(R4)    ;ASSUME BUFFERED I/O
03 59  E8 0145 342      BLBS    R9,NALLOC        ;IF SET, BUFFERED I/O
3E A4  3F 0148 343      PUSHAW  PCB$W_DIOCNT(R4)    ;ELSE DIRECT I/O
      9E  B6 014B 344 NALLOC: INCW   @(SP)+          ;RESTORE COUNT, SINCE NO I/O STARTED
      D8  11 014D 345      BRB     ERROR            ;
      014F 346
      014F 347 ;
      014F 348 ; Convert section index to window address.
      014F 349 ;
      014F 350
50  04 A6  32 014F 351 SECTION:CVTWL  CCB$L_WIND(R6),R0      ;SIGN EXTEND SECTION INDEX
00000000'9F D0 0153 352      MOVL    @#CTL$GL_PHD,R1      ;GET ADDRESS OF PROCESS HEADER
      51  0159
51  20 A1  C0 015A 353      ADDL    PHD$L_PSTBASOFF(R1),R1 ;CALCULATE BASE ADDRESS OF SECTION TABLE
      OC A140 D0 015E 354      MOVL    SEC$L_WINDOW(R1)[R0],-4(R2) ;GET ADDRESS OF REAL WINDOW
      FC A2  0162
      52  11 0164 355      BRB     NOSECT            ;
      0166 356
      3E A4  3F 0166 364 NODCNT: PUSHAW  PCB$W_DIOCNT(R4)    ;SET FOR DIRECT I/O FUNCTION
      8F  11 0169 365      BRB     NOCNT            ;
      016B 366
      3E A4  B7 016B 367 DIRECT: DECW   PCB$W_DIOCNT(R4)    ;CHARGE FOR ANOTHER DIRECT I/O
      F6  19 016E 368      BLSS   NODCNT            ;BR IF NONE ALLOWED
      0170 369 OK:
52  0000'DF 0F 0170 370 GTPKT: REMQUE  @W^IOC$GL_IRPFL,R2    ;GET I/O PACKET FROM LOOK ASIDE LIST
      C5  1D 0175 371      BVS    ALLOC            ;IF VS EMPTY LIST
      0177 372
      0177 373 ;
      0177 374 ; BUILD DEVICE INDEPENDENT PART OF I/O PACKET
      0177 375 ;
      0177 376 ; R2 - IRP Address
      0177 377 ; R4 - PCB Address
      0177 378 ; R5 - UCB Address
      0177 379 ; R6 - CCB Address
      0177 380 ; R7 - Function code (original)
      0177 381 ; R8 - FDT address
      0177 382 ; R9 - Channel index @ 16 + (IRP$M_BUFIO -- if buffered I/O)
      0177 383 ; R10 - Function code (transformed)
      0177 384 ; R11 - Access mode
      0177 385 ;
      0A A6  B6 0177 386 SUCCES: INCW   CCB$W_IOC(R6)        ;INCREMENT OUTSTANDING I/O ON CHANNEL
      017A 387      ASSUME  IRP$W_SIZE EQ 8        ;FOR FOLLOWING OPTIMIZATION
53  82  7E 017A 388      MOVAQ  (R2)+,R3        ;COPY ADDRESS AND ADD IRP$W_SIZE TO R2
50  14 AC  7D 017D 389      MOVQ   ASTADR(AP),R0      ;INSERT AST ADDRESS AND PARAMETER
008C C4  D0 0181 390      MOVL   PCB$L_ARB(R4),IRP$L_ARB(R3) ;COPY ACCESS RIGHTS BLOCK ADDRESS
      58 A3  0185

```



```

-13
                                01F9 443                                ;COMPARES BELOW
                                01F9 444 ;
                                01F9 445 ; CHECK IF REQUESTING PROCESS HAS PRIVILEGE TO ACCESS DEVICE
                                01F9 446 ;
                                01F9 447 ; NOTE: LOW BIT OF FUNCTION CODE WAS CLEARED ABOVE
                                01F9 448 ;
                                01F9 449
;SRB0118          59 30 D1 01F9 .1          ASSUME IO$_READVBLK-IO$_WRITEVBLK EQ 1
-1              35 12 01FC 451          CMPL S^#IO$_WRITEVBLK,R9          ;VIRTUAL READ OR WRITE?
OD 5B 0E E1 01FE 452          BNEQ 15$          ;IF NEQ NO
                                BBC S^#DEV$V_FOD,R11,5$          ;IF CLR, NOT FILE DEVICE
                                0202 453
                                0202 454 ;
                                0202 455 ; THE FOLLOWING TEST IS NECESSITATED BY THE SYSTEM INITIALIZATION SEQUENCE
                                0202 456 ;
                                0202 457
          18 A3 D5 0202 458          TSTL IRP$L_WIND(R3)          ;WINDOW ADDRESS SPECIFIED?
          18 12 0205 459          BNEQ 90$          ;IF NEQ YES
          53 5B 13 E1 0207 460          BBC S^#DEV$V_MNT,R11,60$          ;IF CLR, DEVICE NOT MOUNTED
          61 5B 18 E1 020B 461          BBC S^#DEV$V_FOR,R11,80$          ;IF CLR, MOUNTED STRUCTURED
                                020F 462
                                020F 463 ;
                                020F 464 ; CONVERT VIRTUAL READ/WRITE FUNCTION TO ITS LOGICAL COUNTERPART
                                020F 465 ;
                                020F 466
          57 10 C2 020F 467 5$:          SUBL S^#IO$_READVBLK-IO$_READLBLK,R7 ;CONVERT TO LOGICAL FUNCTION
          20 A3 10 A2 0212 468          SUBW S^#IO$_READVBLK-IO$_READLBLK,IRP$W_FUNC(R3) ;
          00014040 8F D3 0216 469          BITL #<DEV$M_SPL!-          ;NOT SPOOLED,
          5B 021C 470
                                DEV$M_FOD!-          ;NOT FILE DEVICE,
                                021D 471          DEV$M_SHR>,R11          ;AND NOT SHARABLE
          14 12 021D 472          BNEQ 15$          ;BR IF SATISFIED
                                021F 473 ;
                                021F 474 ; CHECK IF AST QUOTA IS EXCEEDED
                                021F 475 ;
                                021F 476
          38 A4 B5 021F 477 90$:          TSTW PCB$W_ASTCNT(R4)          ;AST QUEUE ENTRY QUOTA EXCEEDED?
          40 19 0222 478          BLSS 75$          ;IF LEQ YES
                                0224 479
                                0224 480 ;
                                0224 481 ; SCAN FUNCTION DECISION TABLE CALLING EACH SELECTED ACTION ROUTINE WITH:
                                0224 482 ;
                                0224 483 ; R0 = ADDRESS OF ACTION ROUTINE ENTRY POINT.
                                0224 484 ; R1 = SCRATCH.
                                0224 485 ; R2 = SCRATCH.
                                0224 486 ; R3 = ADDRESS OF I/O REQUEST PACKET.
                                0224 487 ; R4 = CURRENT PROCESS PCB ADDRESS.
                                0224 488 ; R5 = ASSIGNED DEVICE UCB ADDRESS.
                                0224 489 ; R6 = ADDRESS OF CCB.
                                0224 490 ; R7 = I/O FUNCTION CODE BIT NUMBER.
                                0224 491 ; R8 = FDT DISPATCH ADDRESS. (UPDATED TO POINT TO ACTION ROUTINE MASKS)
                                0224 492 ; R9 = SCRATCH.
                                0224 493 ; R10 = SCRATCH.
                                0224 494 ; R11 = SCRATCH.
                                0224 495 ; AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
                                0224 496 ;
                                0224 497 ; NB: in the Guide to Writing a Device Driver, we document the contents
                                0224 498 ; of R0 as being the address of the FDT action routine entry point.

```

```

0224 499 ; This is the only reason that the dispatch code below does not read:
0224 500 ; JSB @8(R8)
0224 501 ; Should future generations wish to modify FDT dispatching to use the
0224 502 ; single dispatch instruction, they must bear the responsibility for
0224 503 ; breaking user written drivers.
0224 504
58 0C C0 0224 505 110$: ADDL #12,R8 ;POINT TO NEXT FUNCTION MASK
F9 68 57 E1 0227 506 BBC R7,(R8),110$ ;IF CLR, THEN ACTION NOT SELECTED
50 08 A8 D0 022B 507 MOVL 8(R8),R0 ;GET ADDRESS OF ACTION ROUTINE
60 16 022F 508 JSB (R0) ;CALL ACTION ROUTINE
F1 11 0231 509 BRB 110$ ;
0233 510
0233 511 ;
0233 512 ; CONTINUE DECODING FUNCTIONS OTHER THAN VIRTUAL READ/WRITE
0233 513 ;
0233 514
57 2F D1 0233 515 15$: CMPL SA#IO$_LOGICAL,R7 ;VIRTUAL I/O FUNCTION?
38 19 0236 516 BLSS 80$ ;IF LSS YES
0238 517
0238 518 ;
0238 519 ; LOGICAL OR PHYSICAL I/O FUNCTION
0238 520 ;
0238 521
0238 522 IFNPRIV PHY_IO,80$ ;PROCESS HAVE PHYSICAL I/O PRIVILEGE?
57 1F D1 023E 523 CMPL SA#IO$_PHYSICAL,R7 ;PHYSICAL I/O FUNCTION?
6E 19 0241 524 BLSS 20$ ;IF LSS NO
0243 525 IFNPRIV LOG_IO,60$ ;PROCESS HAVE LOGICAL I/O PRIVILEGE?
22 08 A6 04 E2 0249 .1 BBSS #CCB$_PHYCHKDON,CCB$_STS(R6),80$ ;XFER IF CHECK DONE
;LMP0206 5A 0000 CF 9E 024E .2 MOVAB WAEXE$CHKPHYACCES,R10 ;SET FOR PHYSICAL I/O FUNCTION CHECK
;LMP0221 6C 11 0253 .3 BRB 30$ ;
;LMP0221 0255 .4
;LMP0221 17 5B 10 E1 0255 .5 40$: BBC SA#DEV$_SHR,R11,80$ ;IF CLR, DEVICE NOT SHAREABLE
;LMP0221 0259 .6 50$: ; R4 - PCB ADDRESS
;LMP0221 0259 .7 ; R5 - UCB ADDRESS
-7 6A 16 0259 533 JSB (R10) ;CHECK ACCESS TO VOLUME
12 50 E8 025B 534 BLBS R0,80$ ;IF LBS ACCESS ALLOWED
50 24 3C 025E 535 60$: MOVZWL #SS$_NOPRIV,R0 ;SET NO PRIVILEGE STATUS
00FC 31 0261 536 70$: BRW EXE$ABORTIO ;
10 A3 D5 0264 537 75$: TSTL IRP$_AST(R3) ;DOES THIS REQUEST NEED AN AST?
BB 13 0267 538 BEQL 110$ ;NO, THEN CAN'T BE QUOTA EXCEEDED.
50 1C 3C 0269 539 MOVZWL #SS$_EXQUOTA,R0 ;AST QUOTA EXCEEDED
F3 11 026C 540 BRB 70$ ;
;LMP0206 E5 11 026E 540 BRB 40$ ;INTERMEDIATE BRANCH
0270 541 ;
0270 542 ; PROCESS HAS ACCESS TO DEVICE
0270 543 ;
0270 544
57 1F D1 0270 545 80$: CMPL SA#IO$_PHYSICAL,R7 ;LOGICAL OR VIRTUAL I/O FUNCTION?
AA 19 0273 546 BLSS 90$ ;IF LSS YES
0100 8F A8 0275 547 BISW #IRP$_PHYSIO,IRP$_STS(R3) ;SET PHYSICAL I/O FLAG
2A A3 0279
59 14 AC D0 027B 548 MOVL <P6-P1>(AP),R9 ;GET ADDRESS OF DIAGNOSTIC BUFFER
9E 13 027F 549 BEQL 90$ ;IF EQL THEN NOT SPECIFIED
0281 550 ;
0281 551 ; Process diagnostic buffer parameter
0281 552 ;
0281 553 IFNPRIV DIAGNOSE,60$ ;PROCESS HAVE PRIVILEGE TO DIAGNOSE?

```

```

;LMP0206
-1
51 0088 C5 D0 0287 554      MOVL   UCB$_DDT(R5),R1      ;GET ADDRESS OF DDT
51 14 A1 3C 028C 555      MOVZWL DDT$_DIAGBUF(R1),R1  ;GET SIZE OF DIAGNOSTIC BUFFER
      1C 13 0290 .1        BEQL   130$                ;IF EQL NO DIAGNOSTIC FUNCTIONS
      53 DD 0292 557      PUSHL  R3                  ;SAVE I/O PACKET ADDRESS
      FD69 30 0294 558     BSBW   EXE$ALLOCFBUF        ;ALLOCATE DIAGNOSTIC BUFFER
      53 8ED0 0297 559     POPL   R3                      ;RETRIEVE I/O PACKET ADDRESS
      C4 50 E9 029A 560     BLBC   R0,70$              ;IF LBC ALLOCATION FAILURE
4C A3 52 D0 029D 561      MOVL   R2,IRP$_DIAGBUF(R3)  ;SAVE ADDRESS OF DIAGNOSTIC BUFFER
82 0C A2 9E 02A1 562      MOVAB  12(R2),(R2)+         ;SET POINTER TO DATA AREA
      62 59 D0 02A5 563     MOVL   R9,(R2)             ;SAVE USER ADDRESS OF DIAGNOSTIC BUFFER
      0080 8F A8 02A8 564     BISW   #IRP$_DIAGBUF,IRP$_STS(R3) ;SET DIAGNOSTIC BUFFER PRESENT
      2A A3 02AC
      FF6E 31 02AE 565 130$: BRW   90$
      02B1 566 ;
      02B1 567 ; LOGICAL I/O FUNCTION
      02B1 568 ;
      02B1 569
      02B1 570 20$: IFPRIV LOG_IO,130$ ;PROCESS HAVE LOGICAL I/O PRIVILEGE?
;LMP0206
B4 08 A6 03 E2 02B7 .1      BBSS   #CCB$_LOGCHKDON,CCB$_STS(R6),80$ ;XFER IF CHECK DONE
5A 0000'CF 9E 02BC 571      MOVAB  W^EXE$CHKLOGACCES,R10 ;SET FOR LOGICAL I/O FUNCTION CHECK
      02C1 572
      02C1 573 ;
      02C1 574 ; PHYSICAL OR LOGICAL I/O FUNCTION - CHECK ACCESSIBILITY OF DEVICE
      02C1 575 ;
      02C1 576
;LMP0206
-1
99 5B 06 E0 02C1 577 30$: BBS   S^#DEV$_SPL,R11,60$ ;IF SET, SPOOLED DEVICE
A5 5B 0E E1 02C5 .1      BBC    S^#DEV$_FOD,R11,77$ ;IF CLR, NOT FILE DEVICE
91 5B 13 E1 02C9 579     BBC    S^#DEV$_MNT,R11,60$ ;IF CLR, DEVICE NOT MOUNTED
8D 5B 18 E1 02CD 580     BBC    S^#DEV$_FOR,R11,60$ ;IF CLR, MOUNTED STRUCTURED
      FF85 31 02D1 581     BRW   50$
      ;

```

```

02D4 583          .SBTTL  BUILD I/O PACKET FOR PAGE READ/WRITE
02D4 584 ;+
02D4 585 ; EXE$BUILDPKTR - BUILD I/O PACKET FOR PAGE READ
02D4 586 ; EXE$BUILDPKTW - BUILD I/O PACKET FOR PAGE WRITE
02D4 587 ; EXE$BLDPKTSWPR - BUILD I/O PACKET FOR SWAP READ
02D4 588 ; EXE$BLDPKTSWPW - BUILD I/O PACKET FOR SWAP WRITE
02D4 589 ; EXE$BLDPKTGSR - BUILD I/O PACKET FOR SHARED MEMORY GLOBAL SECTION READ
02D4 590 ; EXE$BLDPKTGSW - BUILD I/O PACKET FOR SHARED MEMORY GLOBAL SECTION WRITE
02D4 591 ;
02D4 592 ; THIS ROUTINE IS CALLED TO FILL OUT AND QUEUE AN I/O PACKET
02D4 593 ; FOR A SWAPPING OR PAGING READ OR WRITE.
02D4 594 ;
02D4 595 ; INPUTS:
02D4 596 ;
02D4 597 ;         R0 = VIRTUAL BLOCK NUMBER
02D4 598 ;         R1 = NUMBER OF BYTES TO TRANSFER (PAGE INCREMENTS)
02D4 599 ;         R2 = WINDOW ADDRESS FOR MAPPING VBN TO LBN
02D4 600 ;         R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
02D4 601 ;         R4 = CURRENT PROCESS CONTROL BLOCK ADDRESS
02D4 602 ;             PCB$W_DIOCNT(R4) IS ASSUMED GREATER THAN ZERO
02D4 603 ;             AND MUST BE CHECKED BY THE CALLER.
02D4 604 ;         R5 = I/O REQUEST PACKET ADDRESS
02D4 605 ;             WITH THE FOLLOWING FIELDS ALREADY FILLED IN
02D4 606 ;
02D4 607 ;         IRP$W_SIZE(R5) AND IRP$B_TYPE(R5)
02D4 608 ;             FOR ENTRY AT EXE$BUILDPKTW, EXE$BLDPKTGSR, AND EXE$BLDPKTGSW,
02D4 609 ;             THESE ARE FILLED IN BY THE CALLER FOR ALL OTHER ENTRY POINTS
02D4 610 ;             THEY ARE FILLED IN BY THIS CODE.
02D4 611 ;         IRP$L_AST(R5) =
02D4 612 ;             FOR PAGE READ CASE - SYSTEM VIRTUAL ADDRESS OF SLAVE (PROCESS)
02D4 613 ;             PAGE TABLE ENTRY FOR THE CASE OF A GLOBAL PAGE READ.
02D4 614 ;             THIS MUST BE 0 FOR A SYSTEM OR PROCESS PAGE READ.
02D4 615 ;             FOR PAGE WRITE CASE - STANDARD QI/O AST ADDRESS
02D4 616 ;             FOR SWAPIO CASE - THIS PARAMETER IS CURRENTLY NOT USED
02D4 617 ;
02D4 618 ;         IRP$L_ASTPRM(R5) =
02D4 619 ;             FOR PAGE READ CASE - THE CONTENTS OF THE FAULTED PAGE TABLE ENTRY
02D4 620 ;             USED TO RECOVER THE ORIGINAL BACKING STORE ADDRESS WHEN A PAGE
02D4 621 ;             READ ERROR OCCURRED FOR A COPY ON REFERENCE PAGE.
02D4 622 ;             FOR PAGE WRITE CASE - STANDARD QI/O AST PARAMETER
02D4 623 ;             FOR SWAPIO CASE - ADDRESS OF KERNEL AST ROUTINE TO CALL
02D4 624 ;
02D4 625 ;         IRP$B_PRI(R5) = THE PRIORITY AT WHICH THE TRANSFER IS TO BE QUEUED
02D4 626 ;
02D4 627 ;         IRP$B_RMOD(R5) =
02D4 628 ;             FOR PAGE WRITE CASE - STANDARD QI/O MODE OF REQUESTER
02D4 629 ;             FOR ALL OTHER CASES - CONTAINS GARBAGE WHICH IS IGNORED
02D4 630 ;
02D4 631 ;         IRP$B_EFN(R5) =
02D4 632 ;             FOR PAGE WRITE CASE - STANDARD QI/O EVENT FLAG NUMBER
02D4 633 ;             FOR ALL OTHER CASES - CONTAINS GARBAGE WHICH IS IGNORED
02D4 634 ;
02D4 635 ;         IRP$L_IOSB(R5) =
02D4 636 ;             FOR PAGE WRITE CASE - STANDARD QI/O I/O STATUS BLOCK ADDRESS
02D4 637 ;             FOR ALL OTHER CASES - CONTAINS GARBAGE WHICH IS IGNORED
02D4 638 ;
02D4 639 ; OUTPUTS:

```

```

02D4 640 ;
02D4 641 ; R4,R5 ALTERED
02D4 642 ; -
02D4 643
02D4 644 .ENABL LSB
;LJK0172 02D4 .1
;LJK0172 02D4 .2 ; Note that the differentiation between READ and WRITE operations is
;LJK0172 02D4 .3 ; encoded in the setting of the IRP$M_FUNC bit.
;LJK0172 02D4 .4 ;
;LJK0172 02D4 .5 ; IRP$M_FUNC = 1 => IO$_READPBLK ; Read operation
;LJK0172 02D4 .6 ; IRP$M_FUNC = 0 => IO$_WRITEPBLK ; Write operation
;LJK0172 02D4 .7
;LJK0172 02D4 .8 EXE$BLDPKTGSR:: ;BUILD PACKET FOR SHMGSD READ
;LJK0172 00520000 8F DD 02D4 .9 PUSHL #<IRP$M_SWAPIO ! IRP$M_VIRTUAL ! IRP$M_FUNC>@16
;LJK0172 30 11 02DA .10 BRB 20$ ;TYPE/SIZE ALREADY SET IN PACKET
;LJK0172 02DC .11
;LJK0172 02DC .12 EXE$BLDPKTGSW:: ;BUILD PACKET FOR SHMGSD WRITE
;LJK0172 00500000 8F DD 02DC .13 PUSHL #<IRP$M_SWAPIO ! IRP$M_VIRTUAL>@16
;LJK0172 28 11 02E2 .14 BRB 20$ ;TYPE/SIZE ALREADY SET IN PACKET
;LJK0172 02E4 .15
;LJK0172 02E4 .16 EXE$BLDPKTSWPR:: ;BUILD SWAP READ PACKET
;LJK0172 00520000 8F DD 02E4 .17 PUSHL #<IRP$M_SWAPIO ! IRP$M_VIRTUAL ! IRP$M_FUNC>@16
;LJK0172 16 11 02EA .18 BRB 10$ ;
;LJK0172 02EC .19
;LJK0172 02EC .20 EXE$BLDPKTSWPW:: ;BUILD SWAP WRITE PACKET
;LJK0172 00500000 8F DD 02EC .21 PUSHL #<IRP$M_SWAPIO ! IRP$M_VIRTUAL>@16
;LJK0172 0E 11 02F2 .22 BRB 10$ ;
;LJK0172 02F4 .23
;LJK0172 02F4 .24 EXE$BUILDPKTW:: ;BUILD I/O PACKET FOR PAGE WRITE
;LJK0172 00140000 8F DD 02F4 .25 PUSHL #<IRP$M_PAGIO ! IRP$M_VIRTUAL>@16
;LJK0172 10 11 02FA .26 BRB 20$ ;
;LJK0172 02FC .27
;LJK0172 02FC .28 EXE$BUILDPKTR:: ;BUILD I/O PACKET FOR PAGE READ
;LJK0172 00160000 8F DD 02FC .29 PUSHL #<IRP$M_PAGIO ! IRP$M_VIRTUAL ! IRP$M_FUNC>@16
;LJK0172 0302 .30
;LJK0172 000A00C4 8F FO 0302 .31 10$: INSV #<DYN$C_IRP@16 ! IRP$C_LENGTH>,- ;SET SIZE
;LJK0172 18 00 0308 .32 #0,#24,IRP$W_SIZE(R5) ;AND TYPE OF PACKET
;LJK0172 08 A5 030A
;LJK0172 2C A5 53 DO 030C .33 20$: MOVL R3,IRP$L_SVAPTE(R5) ;SYSTEM VIRTUAL ADR OF PAGE TABLE ENTRY
;LJK0172 4C A5 53 DO 0310 .34 MOVL R3,IRP$L_DIAGBUF(R5) ;NEED COPY OF ORIGINAL FOR SEGMENTED XFERS
;LJK0172 53 55 DO 0314 .35 MOVL R5,R3 ;PACKET ADDRESS TO R3
;LJK0172 6E 01 11 EE 0317 .36 EXTV #<IRP$V_FUNC+16>,#1,(SP),R5
;LJK0172 55 031B
;LJK0172 031C .37
;LJK0172 031C .38 ; R5 = -1 for read
;LJK0172 031C .39 ; R5 = 0 for write
;LJK0172 031C .40
;LJK0172 031C .41 ASSUME <IO$_WRITEPBLK + 1> EQ IO$_READPBLK
;LJK0172 031C .42 ASSUME <WCB$L_WRITES - 4> EQ WCB$L_READS
;LJK0172 031C .43
;LJK0172 28 A245 D6 031C .44 INCL WCB$L_WRITES(R2)[R5] ;USE CODE TO BUMP READ OR WRITE COUNT
;LJK0172 0B 55 A3 0320 .45 SUBW3 R5,#IO$_WRITEPBLK,IRP$W_FUNC(R3) ;SET REAL FUNCTION CODE
;LJK0172 20 A3 0323
;LJK0172 100FFFF 8F CB 0325 .46 BICL3 #^XFFFF,(SP)+,R5 ;GET STATUS BITS AND CLEAR CHANNEL
;LJK0172 55 8E 032B
;LJK0172 032D .47
;LJK0172 032D .48 ASSUME IRP$W_STS EQ IRP$W_CHAN+2

```

;LJK0172  
-33

			032D	.49			
28	A3	55	D0	032D	678	MOVL	R5,IRP\$W_CHAN(R3) ;SET CHANNEL AND STATUS
55	10	A2	D0	0331	679	MOVL	WCB\$L_ORGUCB(R2),R5 ;GET UCB ADDRESS FROM WINDOW
1C	A3	55	D0	0335	680	MOVL	R5,IRP\$L_UCB(R3) ;SET UCB ADDRESS
0C	A3	60	A4	D0	0339	MOVL	PCB\$L_PID(R4),IRP\$L_PID(R3) ;PROCESS ID FROM PCB
48	A3	50	D0	033E	682	MOVL	R0,IRP\$L_SEGVBN(R3) ;STARTING VIRTUAL BLOCK NUMBER
18	A3	52	D0	0342	683	MOVL	R2,IRP\$L_WIND(R3) ;WINDOW ADDRESS
	008C	C4	D0	0346	684	MOVL	PCB\$L_ARB(R4),IRP\$L_ARB(R3) ;ACCESS RIGHTS BLOCK ADDRESS
	58	A3		034A			
				034C	685		
	30	A3	B4	034C	686	CLR	IRP\$W_BOFF(R3) ;ZERO BYTE OFFSET
32	A3	51	D0	034F	687	MOVL	R1,IRP\$L_BCNT(R3) ;SET BYTE COUNT
	40	A3	D4	0353	688	CLRL	IRP\$L_ABCNT(R3) ;ZERO ACCUMULATED BYTE COUNT
44	A3	51	D0	0356	689	MOVL	R1,IRP\$L_OBCNT(R3) ;SET ORIGINAL BYTE COUNT
				035A	690		
				035A	691	.IF	DF,CA\$_MEASURE_IOT
				035A	692		
	FCA3'	30		035A	693	BSBW	PMS\$START_RQ ;INSERT START OF I/O REQUEST MESSAGE
				035D	694		
				035D	695	.ENDC	
				035D	696		
	FCA0'	31		035D	697	BRW	IOC\$QNXTSEG1 ;QUEUE THE FIRST SEGMENT OF THE I/O REQUEST
				0360	698		;AND RETURN
				0360	699	.DSABL	LSB



```

0360 701          .SBTTL  COMPLETE I/O OPERATION
0360 702 ;+
0360 703 ; EXE$ABORTIO - ABORT I/O OPERATION
0360 704 ;
0360 705 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
0360 706 ; TO FINISH AN I/O OPERATION WITHOUT RETURNING THE FINAL I/O STATUS.
0360 707 ;
0360 708 ; EXE$FINISHIO - FINISH I/O OPERATION
0360 709 ;
0360 710 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
0360 711 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS.
0360 712 ;
0360 713 ; EXE$FINISHIOC - FINISH I/O OPERATION WITH SECOND I/O STATUS LONGWORD CLEARED
0360 714 ;
0360 715 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
0360 716 ; TO FINISH AN I/O OPERATION AND RETURN THE FINAL I/O STATUS WITH THE
0360 717 ; SECOND I/O STATUS LONGWORD CLEARED.
0360 718 ;
0360 719 ; INPUTS:
0360 720 ;
0360 721 ;      R0 = FIRST LONGWORD OF FINAL I/O STATUS.
0360 722 ;      R1 = SECOND LONGWORD OF FINAL I/O STATUS.
0360 723 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
0360 724 ;      R4 = CURRENT PROCESS PCB ADDRESS.
0360 725 ;      R5 = UCB ADDRESS OF DEVICE UNIT.
0360 726 ;
0360 727 ; OUTPUTS:
0360 728 ;
0360 729 ;      THE FINAL I/O STATUS IS STORED IN THE I/O PACKET AND THE PACKET IS
0360 730 ;      INSERTED IN THE I/O POST PROCESSING QUEUE. A SOFTWARE INTERRUPT
0360 731 ;      IS GENERATED TO INITIATE I/O POST PROCESSING AND THE FIRST WORD
0360 732 ;      OF THE FINAL I/O STATUS IS RETURNED AS THE SERVICE STATUS.
0360 733 ;-
0360 734
0360 735          .ENABL  LSB
0360 736 EXE$ABORTIO::          ;ABORT I/O OPERATION
24 A3 D4 0360 737          CLRL  IRP$L_IOSB(R3)          ;CLEAR ADDRESS OF I/O STATUS BLOCK
11 0B A3 06 E5 0363 738          BBCC  #ACB$V_QUOTA,IRP$B_RMOD(R3),10$ ;IF CLR, NO AST SPECIFIED
38 A4 B6 0368 739          INCW  PCB$W_ASTCNT(R4)        ;UPDATE AVAILABLE AST QUEUE ENTRIES
0C 11 036B 740          BRB    10$
0360 741 EXE$FINISHIOC::      ;FINISH I/O OPERATION CLEAR SECOND LONGWORD
51 D4 036D 742          CLRL  R1          ;CLEAR SECOND I/O STATUS LONGWORD
036F 743 EXE$FINISHIO::      ;FINISH I/O OPERATION
70 A5 D6 036F 744          INCL  UCB$L_OPCNT(R5)          ;INCREMENT OPERATIONS COMPLETED
38 A3 50 7D 0372 745          MOVQ  R0,IRP$L_MEDIA(R3)        ;STORE FINAL I/O STATUS
50 01 3C 0376 746          MOVZWL S^#SS$_NORMAL,R0          ;SET NORMAL COMPLETION STATUS
0000'DF 63 0E 0379 747 10$:  INSQUE (R3),@W^IIOC$GL_PSBLL ;INSERT I/O PACKET IN POST PROCESS QUEUE
037E 748          SOFTINT #IPL$ IOPOST          ;INITIATE SOFTWARE INTERRUPT
39 11 0381 749          BRB    QIORETURN
0383 750          .DSABL  LSB

```

```
0383 752 .SBTTL QUEUE I/O PACKET TO DRIVER
0383 753 ;+
0383 754 ; EXE$QIODRVPKT - QUEUE I/O PACKET TO DRIVER
0383 755 ;
0383 756 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
0383 757 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE DRIVER.
0383 758 ;
0383 759 ; INPUTS:
0383 760 ;
0383 761 ; R3 = ADDRESS OF I/O REQUEST PACKET.
0383 762 ; R4 = CURRENT PROCESS PCB ADDRESS.
0383 763 ; R5 = UCB ADDRESS OF DEVICE UNIT.
0383 764 ;
0383 765 ; OUTPUTS:
0383 766 ;
0383 767 ; THE I/O PACKET IS QUEUED BY PRIORITY IN THE APPROPRIATE DEVICE
0383 768 ; QUEUE AND A NORMAL COMPLETION STATUS IS RETURNED.
0383 769 ;-
0383 770
0383 771 EXE$QIODRVPKT:: ;QUEUE I/O PACKET
66 10 0383 772 BSBB EXE$INSIOQ ;INSERT I/O PACKET IN DEVICE QUEUE
32 11 0385 773 BRB EXE$QIORETURN ;
```

```

0387 775 .SBTTL EXE$ALTQUEPKT - Call driver ALTSTART entry point
0387 776
0387 777 ;
0387 778 ; EXE$ALTQUEPKT - activates a driver at its ALTSTART entry point
0387 779 ;
0387 780 ; Routine description:
0387 781 ;
0387 782 ; Locates and calls a driver entry point supplied as an alternate
0387 783 ; START I/O entry point. Does not test for unit busy before the
0387 784 ; call. Exits by returning to caller.
0387 785 ;
0387 786 ; The routine expects to gain control at or below driver fork
0387 787 ; level. The routine raises to driver fork IPL before the call,
0387 788 ; and restores the previous IPL before returning to its caller.
0387 789 ;
0387 790 ; Inputs:
0387 791 ;
0387 792 ; R3 - address of packet or buffer
0387 793 ; R5 - address of UCB
0387 794 ;
0387 795 ; Outputs:
0387 796 ;
0387 797 ; Control returns to the requesting process.
0387 798 ;
0387 799 ; The routine destroys R0-R1.
0387 800 ;
0387 801 ;--
0387 802
0387 803 EXE$ALTQUEPKT:: ; Start I/O in driver.
0387 804 DSBINT UCB$B_FIPL(R5) ; Raise to fork IPL.
50 0088 C5 D0 038E 805 MOVL UCB$L_DDT(R5),R0 ; Get address of unit's DDT.
1C B0 16 0393 806 JSB @DDT$L_ALTSTART(R0) ; Call alternate start I/O routine.
0396 807 ENBINT ; Reenable interrupts.
05 0399 808 RSB ; Return to caller.

```

```

039A 810 .SBTTL QUEUE I/O PACKET TO ACP
039A 811 ;+
039A 812 ; EXE$QIOACPPKT - QUEUE I/O PACKET TO ACP
039A 813 ;
039A 814 ; THIS ROUTINE IS JUMPED TO FROM A FUNCTION DECISION TABLE ACTION ROUTINE
;CDS0001 039A .1 ; TO QUEUE AN I/O PACKET TO THE APPROPRIATE ACP OR XQP.
-1 039A 816 ;
039A 817 ; INPUTS:
039A 818 ;
039A 819 ; R3 = ADDRESS OF I/O REQUEST PACKET.
039A 820 ; R4 = CURRENT PROCESS PCB ADDRESS.
039A 821 ; R5 = UCB ADDRESS OF DEVICE UNIT.
039A 822 ;
039A 823 ; CURRENT IPL MUST BE AT SYNCH OR HIGHER LEVEL.
039A 824 ;
039A 825 ; OUTPUTS:
039A 826 ;
039A 827 ; R4 ALTERED
;CDS0001 039A .1 ; THE I/O PACKET IS QUEUED AT THE END OF THE APPROPRIATE ACP OR XQP QUEUE
-1 039A 829 ; AND A NORMAL COMPLETION STATUS IS RETURNED.
039A 830 ;-
039A 831
039A 832 EXE$QIOACPPKT:: ;QUEUE I/O PACKET TO ACP
52 34 A5 D0 039A 833 MOVL UCB$_VCB(R5),R2 ;GET ADDRESS OF VCB
52 10 A2 D0 039E 834 MOVL VCB$_ACB(R2),R2 ;GET ADDRESS OF ACP ACB
;CDS0001 039A 835 TSTL AQB$_ACPPID(R2) ;GET ADDRESS OF AQB
;CDS0001 19 13 03A5 .2 BEQL XQP ;EQL IF IT'S FOR XQP
60 10 03A7 835 BSBB EXE$INSERTIRP ;INSERT I/O PACKET IN ACP QUEUE
0E 12 03A9 836 BNEQ EXE$QIORETURN ;IF NEQ NOT FIRST ENTRY IN QUEUE
51 0C A2 D0 03AB 837 MOVL AQB$_ACPPID(R2),R1 ;GET ACP PROCESS ID
FC4E' 30 03AF 838 BSBW SCH$WAKE ;WAKE UP ACP PROCESS
04 50 E8 03B2 839 BLBS R0,EXE$QIORETURN ;IF LBS ACP STILL PRESENT
03B5 840 BUG_CHECK NONEXSTACP ;NONEXISTENT ACP PROCESS
03B9 841 EXE$QIORETURN:: ;QUEUE I/O REQUEST COMPLETION STATUS RETURN
50 01 3C 03B9 842 MOVZWL #SS$_NORMAL,R0 ;SET NORMAL COMPLETION STATUS
03BC 843 QIORETURN: ;RETURN SPECIFIED STATUS
03BC 844 SETIPL #0 ;ALLOW ALL INTERRUPTS
04 03BF 845 RET ;
;CDS0005 03C0 .1 XQP:
;CDS0005 55 60 A3 9E 03C0 .2 MOVAB IRP$_FQFL(R3), R5 ;USE CDRP PART OF IRP AS ACB
;CDS0005 03C4 .3 SETIPL #IPL$_ASTDEL ;ALLOW PAGEFAULTS
;CDS0005 F2 AF 9F 03C7 .4 PUSHAB QIORETURN ;RETURN ADDRESS FROM EXE$QXQPPKT
;CDS0005 03CA .5 ;
;CDS0005 03CA .6 ; FALL THROUGH TO XQP QUEUEING ROUTINE IMMEDIATELY FOLLOWING.
;CDS0005 03CA .7 ; RSB FROM THIS ROUTINE RETURNS TO EXIT ABOVE.
;CDS0005 03CA .8 ;
;CDS0005 03CA .9 ;
;CDS0005 03CA .10 .SBTTL EXE$QXQPPKT - QUEUE I/O PACKET TO XQP
;CDS0005 03CA .11 ;+
;CDS0005 03CA .12 ; EXE$QXQPPKT - INSERT I/O PACKET IN XQP QUEUE
;CDS0005 03CA .13 ;
;CDS0005 03CA .14 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN THE XQP QUEUE
;CDS0005 03CA .15 ; AND START THE THREAD OF EXECUTION IF IT IS THE ONLY REQUEST.
;CDS0005 03CA .16 ;
;CDS0005 03CA .17 ; CALLING SEQUENCE:
;CDS0005 03CA .18 ; BSB/JSB EXE$QXQPPKT - THIS IS EITHER CALLED FROM QIO OR
;CDS0005 03CA .19 ; AS A SPECIAL KERNEL AST INVOKED BY IOPOST.

```

```

;CDS0005      03CA      .20 ;
;CDS0005      03CA      .21 ; INPUTS:
;CDS0005      03CA      .22 ;
;CDS0005      03CA      .23 ;      R4 = CURRENT PROCESS PCB ADDRESS.
;CDS0005      03CA      .24 ;      R5 = ADDRESS OF TEMP ACB PART OF IRP.
;CDS0005      03CA      .25 ;
;CDS0005      03CA      .26 ; OUTPUTS:
;CDS0005      03CA      .27 ;
;CDS0005      03CA      .28 ;      R0 = status from SCH$QAST.
;CDS0005      03CA      .29 ;
;CDS0005      03CA      .30 ;      IF SUCCESS:
;CDS0005      03CA      .31 ;      A KERNEL AST IS QUEUED TO THE DISPATCH ROUTINE OF THE XQP
;CDS0005      03CA      .32 ;      IF NO PACKETS WERE ALREADY ON THE REQUEST QUEUE OF THE XQP.
;CDS0005      03CA      .33 ;
;CDS0005      03CA      .34 ;      THIS ROUTINE MUST BE CALLED AT IPL ASTDEL SO THAT THE
;CDS0005      03CA      .35 ;      IRP CANNOT BE LOST (BECAUSE OF PROCESS DELETION) UNTIL IT
;CDS0005      03CA      .36 ;      IS PLACED ON THE XQP REQUEST QUEUE.
;CDS0005      03CA      .37 ;
;CDS0005      03CA      .38 ; -
;CDS0005      03CA      .39
;CDS0005      03CA      .40 EXE$QXQPPKT:
;CDS0005      03CA      .41      MOVL      GACTL$GL_F11BXQP, R0      ;ADDR OF XQP QUEUE HEAD
;CDS0005      03D0
;CDS0005      14 A5      A0 A5      9E 03D1      .42      MOVAB      CDRP$L_IOQFL(R5), -      ;ADDRESS OF IRP
;CDS0005      03D6      .43      ACB$L_ASTPRM(R5)      ;IS AST PARAMETER.
;CDS0005      20 90 03D6      .44      MOVVB      #PSL$C_KERNEL!ACB$M_NODELETE,-;KERNEL MODE, DON'T DELETE IRP
;CDS0005      0B A5      03D8      .45      ACB$B_RMOD(R5)
;CDS0005      0C A5      60 A4      D0 03DA      .46      MOVL      PCB$L_PID(R4), ACB$L_PID(R5) ;COPY PID.
;CDS0005      10 A5      08 A0      D0 03DF      .47      MOVL      F11B$L_DISPATCH(R0), ACB$L_AST(R5) ;XQP DISPATCHER ADDRESS.
;CDS0005      52 02      D0 03E4      .48      MOVL      #PRI$_RESAVL, R2      ;SAME AS AFTER WAITING FOR A LOCK.
;CDS0005      FC16'      30 03E7      .49      BSBW      SCH$QAST      ;QUEUE THE AST.
;CDS0005      05 03EA      .50      RSB      ;AND RETURN

```

;SSA0017

```

03EB 847          .SBTTL  INSERT I/O PACKET IN UNIT QUEUE
03EB 848 ;+
03EB 849 ; EXE$INSIOQ - INSERT I/O PACKET IN UNIT QUEUE
03EB 850 ;
03EB 851 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A UNIT QUEUE AND CALL
03EB 852 ; THE APPROPRIATE I/O DRIVER IF THE UNIT IS NOT BUSY.
03EB 853 ;
03EB 854 ; INPUTS:
03EB 855 ;
03EB 856 ;          R3 = ADDRESS OF I/O REQUEST PACKET.
03EB 857 ;          R5 = UCB ADDRESS OF DEVICE UNIT.
03EB 858 ; -
03EB 859
03EB 860 EXE$INSIOQ::          ;INSERT IN I/O QUEUE
03EB 861          DSBINT  UCB$B_FIPL(R5)          ;RAISE IPL TO FORK LEVEL
05 64 A5 6A A5 B6 03F2 .1          INCW  UCB$W_QLEN(R5)          ; Bump device queue length
06 08 E2 03F5 862          BBSS  #UCB$V_BSY,UCB$W_STS(R5),10$ ;IF SET, THEN DEVICE IS BUSY
FC03 30 03FA 863          BSBW  IOC$INITIATE          ;INITIATE I/O FUNCTION
06 11 03FD 864          BRB   20$          ;
52 4C A5 DE 03FF 865 10$:        MOVAL  UCB$L_IOQFL(R5),R2          ;GET ADDRESS OF I/O QUEUE LISTHEAD
04 10 0403 866          BSBB  EXE$INSERTIRP          ;INSERT I/O PACKET IN DEVICE QUEUE
0405 867 20$:        ENBINT          ;ENABLE INTERRUPTS
05 0408 868          RSB   ;

```

```

0409 870 .SBTTL INSERT I/O PACKET IN QUEUE BY PRIORITY
0409 871 ;+
0409 872 ; EXE$INSERTIRP - INSERT I/O PACKET IN QUEUE BY PRIORITY
0409 873 ;
0409 874 ; THIS ROUTINE IS CALLED TO INSERT AN I/O PACKET IN A SPECIFIED QUEUE BY
0409 875 ; PRIORITY.
0409 876 ;
0409 877 ; INPUTS:
0409 878 ;
0409 879 ; R2 = ADDRESS OF QUEUE LISTHEAD.
0409 880 ; R3 = ADDRESS OF I/O PACKET.
0409 881 ;
0409 882 ; CURRENT IPL MUST BE THE FORK LEVEL OF THE RESPECTIVE DRIVER PROCESS
0409 883 ; OR HIGHER.
0409 884 ;
0409 885 ; OUTPUTS:
0409 886 ;
0409 887 ; THE I/O PACKET IS INSERTED IN THE SPECIFIED QUEUE BY PRIORITY AND
0409 888 ; THE 'Z' CONDITION CODE IS RETURNED TO THE CALLER.
0409 889 ;
0409 890 ; 'Z' = 1 = ENTRY WAS FIRST ENTRY IN THE QUEUE.
0409 891 ;
0409 892 ; 'Z' = 0 = ENTRIES WERE ALREADY IN THE QUEUE.
0409 893 ;
0409 894 ; R2 AND R3 ARE PRESERVED ACROSS THE CALL.
0409 895 ;-
0409 896
0409 897 EXE$INSERTIRP:: ;INSERT I/O PACKET IN QUEUE BY PRIORITY
51 52 D0 0409 898 MOVL R2,R1 ;COPY LISTHEAD ADDRESS
51 04 A1 D0 040C 899 10$: MOVL IRP$L_IOQBL(R1),R1 ;GET ADDRESS OF NEXT ENTRY
51 52 D1 0410 900 CMPL R2,R1 ;END OF QUEUE?
23 A1 23 A3 91 0413 901 BEQL 20$ ;IF EQL YES
61 63 0E 0415 902 CMPB IRP$B_PRI(R3),IRP$B_PRI(R1) ;NEW ENTRY PRIORITY GREATER?
05 041F 903 BLSSU 10$ ;IF LSS YES
0420 904 20$: INSQUE IRP$L_IOQFL(R3),IRP$L_IOQFL(R1) ;INSERT PACKET IN I/O QUEUE
0420 905 RSB ;
0420 906
0420 907 .END

```

ACB\$B_RMOD	=	0000000B			EXE\$CHKLOGACCES	*****	X	01
ACB\$L_AST	=	00000010			EXE\$CHKPHYACCES	*****	X	01
ACB\$L_ASTPRM	=	00000014			EXE\$CHKRDACCES	*****	X	01
ACB\$L_PID	=	0000000C			EXE\$CHKWRTACCES	*****	X	01
ACB\$M_NODELETE	=	00000020			EXE\$FINISHIO	0000036F	RG	01
ACB\$M_QUOTA	=	00000040			EXE\$FINISHIOC	0000036D	RG	01
ACB\$V_QUOTA	=	00000006			EXE\$INSERTIRP	00000409	RG	01
ACCVIO		00000124	R	01	EXE\$INSIOQ	000003EB	RG	01
ALLOC		0000013C	R	01	EXE\$QIO	0000006B	RG	01
AQB\$L_ACPPID	=	0000000C			EXE\$QIOACPPKT	0000039A	RG	01
ASTADR	=	00000014			EXE\$QIODRVPKT	00000383	RG	01
ASTPRM	=	00000018			EXE\$QIORETURN	000003B9	RG	01
BUG\$_NONEXSTACP		*****	X	01	EXE\$QXQPPKT	000003CA	RG	01
CCB\$B_AMOD	=	00000009			EXE\$SNGLEQUOTA	*****	X	01
CCB\$B_STS	=	00000008			F11B\$L_DISPATCH	=	00000008	
CCB\$C_LENGTH	=	00000010			FDTACT	=	00000010	
CCB\$L_UCB	=	00000000			FUNC	=	0000000C	
CCB\$L_WIND	=	00000004			GTPKT	00000170	R	01
CCB\$V_LOGCHKDON	=	00000003			ILLIO	0000010C	R	01
CCB\$V_PHYCHKDON	=	00000004			IO\$M_FCODE	=	0000003F	
CCB\$V_PROCHKDON	=	00000002			IO\$_ACPCONTROL	=	00000038	
CCB\$W_IOC	=	0000000A			IO\$_DEACCESS	=	00000034	
CDRP\$L_IOQFL	=	FFFFFFFA0			IO\$_LOGICAL	=	0000002F	
CHAN	=	00000008			IO\$_PHYSICAL	=	0000001F	
CHKDON		000000C8	R	01	IO\$_READHEAD	=	0000000E	
CLREF		00000036	R	01	IO\$_READLBLK	=	00000021	
CTL\$GL_CCBBASE		*****	X	01	IO\$_READPBLK	=	0000000C	
CTL\$GL_F11BXQP		*****	X	01	IO\$_READPROMPT	=	00000037	
CTL\$GL_PHD		*****	X	01	IO\$_READTRACKD	=	00000010	
CTL\$GW_CHINDX		*****	X	01	IO\$_READVBLK	=	00000031	
DACSPND		00000048	R	01	IO\$_REREADN	=	00000016	
DDT\$L_ALTSTART	=	0000001C			IO\$_REREADP	=	00000017	
DDT\$L_FDT	=	00000008			IO\$_TTYREADALL	=	0000003A	
DDT\$W_DIAGBUF	=	00000014			IO\$_TTYREADPALL	=	0000003B	
DEV\$M_FOD	=	00004000			IO\$_VIRTUAL	=	0000003F	
DEV\$M_SHR	=	00010000			IO\$_WRITECHECK	=	0000000A	
DEV\$M_SPL	=	00000040			IO\$_WRITECHECKH	=	00000018	
DEV\$V_FOD	=	0000000E			IO\$_WRITEHEAD	=	0000000D	
DEV\$V_FOR	=	00000018			IO\$_WRITELBLK	=	00000020	
DEV\$V_MNT	=	00000013			IO\$_WRITEPBLK	=	0000000B	
DEV\$V_SHR	=	00000010			IO\$_WRITERET	=	00000018	
DEV\$V_SPL	=	00000006			IO\$_WRITETRACKD	=	0000000F	
DIRECT		0000016B	R	01	IO\$_WRITEVBLK	=	00000030	
DYN\$C_IRP	=	0000000A			IOC\$GL_IRPFL	*****	X	01
EFN	=	00000004			IOC\$GL_PSBL	*****	X	01
ERROR		00000127	R	01	IOC\$INITIATE	*****	X	01
ERRORB		00000045	R	01	IOC\$QNXSTSEG1	*****	X	01
EXE\$ABORTIO		00000360	RG	01	IOSB	=	00000010	
EXE\$ALLOCBUF		*****	X	01	IOTYPE	=	00000008	
EXE\$ALLOCI RP		*****	X	01	IPL\$ASTDEL	=	00000002	
EXE\$ALTQUEPKT		00000387	RG	01	IPL\$_IOPOST	=	00000004	
EXE\$BLDPKTGSR		000002D4	RG	01	IPL\$_SYNCH	=	00000008	
EXE\$BLDPKTGSW		000002DC	RG	01	IRP\$B_EFN	=	00000022	
EXE\$BLDPKTSWPR		000002E4	RG	01	IRP\$B_PRI	=	00000023	
EXE\$BLDPKTSWPP		000002EC	RG	01	IRP\$B_RMOD	=	0000000B	
EXE\$BUILDPKTR		000002FC	RG	01	IRP\$B_TYPE	=	0000000A	
EXE\$BUILDPKTW		000002F4	RG	01	IRP\$C_LENGTH	=	000000C4	



IRP\$L_ABCNT	=	00000040		PCB\$W_ASTCNT	=	00000038	
IRP\$L_ARB	=	00000058		PCB\$W_BIOCNT	=	0000003A	
IRP\$L_AST	=	00000010		PCB\$W_DIOCNT	=	0000003E	
IRP\$L_ASTPRM	=	00000014		PHD\$L_PSTBASOFF	=	00000020	
IRP\$L_BCNT	=	00000032		PMS\$GL_IOPFMPDB	*****	X	01
IRP\$L_DIAGBUF	=	0000004C		PMS\$GL_IOPFMSEQ	*****	X	01
IRP\$L_FQFL	=	00000060		PMS\$START_RQ	*****	X	01
IRP\$L_IOQBL	=	00000004		PR\$IPL	=	00000012	
IRP\$L_IOQFL	=	00000000		PR\$I_SIRR	=	00000014	
IRP\$L_IOSB	=	00000024		PRI\$I_RESAVL	=	00000002	
IRP\$L_MEDIA	=	00000038		PRIOSB	000000DA	R	01
IRP\$L_OBCNT	=	00000044		PRIVERR	00000042	R	01
IRP\$L_PID	=	0000000C		PRV\$I_DIAGNOSE	=	00000006	
IRP\$L_SEGVBN	=	00000048		PRV\$I_LOG_IO	=	00000007	
IRP\$L_SEQNUM	=	00000050		PRV\$I_PHY_IO	=	00000016	
IRP\$L_SVAPE	=	0000002C		PSL\$C_KERNEL	=	00000000	
IRP\$L_UCB	=	0000001C		PSL\$I_PRVMOD	=	00000002	
IRP\$L_WIND	=	00000018		PSL\$I_PRVMOD	=	00000016	
IRP\$M_BUFIO	=	00000001		QIO	0000006D	R	01
IRP\$M_DIAGBUF	=	00000080		QIORETURN	000003BC	R	01
IRP\$M_FUNC	=	00000002		READ_ACCESS	00000000	R	01
IRP\$M_PAGIO	=	00000004		RSN\$I_ASTWAIT	=	00000001	
IRP\$M_PHYSIO	=	00000100		SCH\$CLREF	*****	X	01
IRP\$M_SWAPIO	=	00000040		SCH\$GL_RESMASK	*****	X	01
IRP\$M_VIRTUAL	=	00000010		SCH\$GQ_MWAIT	*****	X	01
IRP\$V_FUNC	=	00000001		SCH\$POSTEF	*****	X	01
IRP\$W_BOFF	=	00000030		SCH\$QAST	*****	X	01
IRP\$W_CHAN	=	00000028		SCH\$WAIT	*****	X	01
IRP\$W_FUNC	=	00000020		SCH\$WAKE	*****	X	01
IRP\$W_SIZE	=	00000008		SEC\$L_WINDOW	=	0000000C	
IRP\$W_STS	=	0000002A		SECTION	0000014F	R	01
IVCHAN	0000003B	R	01	SPOOL	0000005E	R	01
LEGAL	=	00000000		SS\$I_ACCVIO	=	0000000C	
MASKH	=	00010001		SS\$I_DEVOFFLINE	=	00000084	
MASKL	=	0100AC00		SS\$I_EXQUOTA	=	0000001C	
NALLOC	0000014B	R	01	SS\$I_ILLIOFUNC	=	000000F4	
NOCNT	000000FA	R	01	SS\$I_IVCHAN	=	0000013C	
NODCNT	00000166	R	01	SS\$I_NOPRIV	=	00000024	
NOIOSB	000000E8	R	01	SS\$I_NORMAL	=	00000001	
NOSECT	000001B8	R	01	SUCCESS	00000177	R	01
NOT_FILE_DEV	00000010	R	01	UCB\$B_FIPL	=	0000000B	
NOT_FILE_DEVB	00000069	R	01	UCB\$L_AMB	=	00000060	
NSPOOL	000000C3	R	01	UCB\$L_DDT	=	00000088	
OFFLINE	00000113	R	01	UCB\$L_DEVCHAR	=	00000038	
OK	00000170	R	01	UCB\$L_IOQFL	=	0000004C	
P1	=	0000001C		UCB\$L_OPCNT	=	00000070	
P2	=	00000020		UCB\$L_VCB	=	00000034	
P3	=	00000024		UCB\$I_BSY	=	00000008	
P4	=	00000028		UCB\$I_ONLINE	=	00000004	
P5	=	0000002C		UCB\$I_QLEN	=	0000006A	
P6	=	00000030		UCB\$I_STS	=	00000064	
PCB\$B_PRIV	=	0000002F		VCB\$L_AQB	=	00000010	
PCB\$L_ARB	=	0000008C		VCHAN	0000007B	R	01
PCB\$L_EFCS	=	00000050		WCB\$L_ORGUCB	=	00000010	
PCB\$L_EFWM	=	0000004C		WCB\$L_READS	=	00000024	
PCB\$L_PID	=	00000060		WCB\$L_WRITES	=	00000028	
PCB\$Q_PRIV	=	00000084		WRITE_ACCESS	00000008	R	01

XQP 000003C0 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 .	00000420 ( 1056.)	01 ( 1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$ABS\$	00000000 ( 0.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.09	00:00:00.54
Command processing	77	00:00:00.67	00:00:06.51
Pass 1	608	00:00:26.37	00:02:38.99
Symbol table sort	0	00:00:03.89	00:00:18.56
Pass 2	250	00:00:06.58	00:00:34.76
Symbol table output	29	00:00:00.21	00:00:00.37
Psect synopsis output	6	00:00:00.03	00:00:00.14
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	983	00:00:37.87	00:03:39.88

The working set limit was 1650 pages.  
139110 bytes (272 pages) of virtual memory were used to buffer the intermediate code.  
There were 140 pages of symbol table space allocated to hold 2591 non-local and 27 local symbols.  
1039 source lines were read in Pass 1, producing 18 object records in Pass 2.  
42 pages of virtual memory were used to define 41 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	26
_\$255\$DUA28:[SYS.LIB]STARLET.MLB;2	11
TOTALS (all libraries)	37

2723 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SYSQIOREQ/OBJ=OBJ\$:SYSQIOREQ MSRC\$:SYSQIOREQ/UPDATE=(ENH\$:SYSQIOREQ)+EXECML\$/LIB

**SYSQIOFDT**

(1)	97	PLACE PROCESS IN I/O RESOURCE WAIT
(1)	148	ONE PARAMETER FUNCTION PROCESSING
(1)	181	ZERO PARAMETER FUNCTION PROCESSING
(1)	213.1	LOCAL DISK VALID FUNCTION PROCESSING
(1)	214	READ AND WRITE FUNCTION PROCESSING
(1)	267	READ AND WRITE FUNCTION BUFFER CHECK AND LOCK ROUTINES
(1)	310	READ AND WRITE BUFFER CHECK AND LOCK AND RETURN ROUTINES
(1)	400	BACKOUT A QIO
(2)	438	CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
(2)	474	CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
(2)	513	CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
(2)	587	CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
(2)	652	SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
(2)	692	SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS
(2)	735	SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
(2)	773	CARRIAGE CONTROL INTERPRETATION

```
0000 1 .TITLE SYSQIOFDT - SYSTEM SERVICE QUEUE I/O FDT SUBROUTINES
;WMC0001 0000 .1 .IDENT 'V03-009'
-1 0000 3
0000 4 ;
0000 5 ;*****
0000 6 ;*
0000 7 ;* COPYRIGHT (c) 1978, 1980, 1982 BY *
0000 8 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 9 ;* ALL RIGHTS RESERVED. *
0000 10 ;*
0000 11 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 12 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 13 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 14 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 15 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 16 ;* TRANSFERRED. *
0000 17 ;*
0000 18 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 19 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 20 ;* CORPORATION. *
0000 21 ;*
0000 22 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 23 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 24 ;*
0000 25 ;*
0000 26 ;*****
0000 27 ;
0000 28 ; D. N. CUTLER 15-SEP-76
0000 29 ;
0000 30 ; MODIFIED BY:
;WMC0001 0000 .1 ; V03-009 WMC0001 Wayne Cardoza 23-Apr-1984
;WMC0001 0000 .2 ; Add a comment warning about general use of EXE$IORSNWAIT.
;WMC0001 0000 .3 ;
;ROW0259 0000 .4 ; V03-008 ROW0259 Ralph O. Weber 20-NOV-1983
;ROW0259 0000 .5 ; For IO$_PACKACK operations passing through EXE$LCLDSKVALID,
;ROW0259 0000 .6 ; always allow the PACKACK request to go to the driver when
;ROW0259 0000 .7 ; UCB$_VALID in UCB$_STS is clear, regardless of any other
;ROW0259 0000 .8 ; conditions. However, UCB$_LCL_VALID and UCB$_ONLCNT must
;ROW0259 0000 .9 ; still be correctly adjusted. This is believed to allow
;ROW0259 0000 .10 ; PACKACKs to fail and be retried.
;ROW0259 0000 .11 ;
;SSA00002 0000 .12 ; V03-007 SSA00002 Stan Amway 30-SEP-1983
;SSA00002 0000 .13 ; Modified BACKOUT_QIO to call new routine PMS$ABORT_RQ
;SSA00002 0000 .14 ; to insure complete traces of I/O activity.
;SSA00002 0000 .15 ;
;ROW0224 0000 .16 ; V03-006 ROW0224 Ralph O. Weber 15-SEP-1983
;ROW0224 0000 .17 ; Change EXE$LCLDSKVALID to alter UCB$_ONLCNT either up or down
;ROW0224 0000 .18 ; only if the local processor has not already performed such an
;ROW0224 0000 .19 ; alteration. Use UCB$_LCL_VALID in UCB$_STS to determine
;ROW0224 0000 .20 ; state of device with respect to the local processor.
;ROW0224 0000 .21 ;
;PRD0030 0000 .22 ; V03-005 PRD0030 Paul R. DeStefano 09-Sep-1983
;PRD0030 0000 .23 ; Added EXE$LCLDSKVALID routine to track disk online count
;PRD0030 0000 .24 ; and local valid status.
;PRD0030 0000 .25 ;
;ROW0192 0000 .26 ; V03-004 ROW0192 Ralph O. Weber 20-AUG-1983
```

```
;ROW0192      0000 .27 ;           Fix EXE$WRITE and EXE$READ to allow longword byte counts.
;ROW0192      0000 .28 ;           This should allow virtual disk transfers to exceed 65K bytes.
;ROW0192      0000 .29 ;           (This will be distributed in V3.5 as SYS ECO 65.)
;ROW0192      0000 .30 ;
;ROW49973     0000 .31 ;           V03-003 ROW49973      Ralph O. Weber      29-OCT-1982
;ROW49973     0000 .32 ;           Change calling requirements for EXE$IORSNWAIT from an entry
;ROW49973     0000 .33 ;           IPL of IPL$ SYNCH to an entry IPL of IPL$_ASTDEL. Have the
;ROW49973     0000 .34 ;           call to BACKOUT_QIO made at IPL$_ASTDEL. Then, raise to
;ROW49973     0000 .35 ;           IPL$ SYNCH to perform scheduler operations. This eliminates
;ROW49973     0000 .36 ;           undesirable page faults at an IPL above IPL$_ASTDEL when
;ROW49973     0000 .37 ;           BACKOUT_QIO references a channel control block.
;ROW49973     0000 .38 ;
;PRD0030      0000 .39 ;           V03-002 ROW49577      Ralph O. Weber      27-SEP-1982
;PRD0030      0000 .40 ;           ECO 25 Change EXE$SETCHAR and EXE$SETMODE to return SS$_ILLIOFUNC if
;PRD0030      0000 .41 ;           UCB$_DEVCLASS equals DC$_DISK. This is to prohibit SETMODE
;PRD0030      0000 .42 ;           (set mode) and SETCHAR (set characteristics) functions on disk
;PRD0030      0000 .43 ;           devices. On disk devices, those functions overwrite the disk
;PRD0030      0000 .44 ;           geometry information which results in abbarant system
;PRD0030      0000 .45 ;           behavior.
-35           0000 .67 ;
           0000 .68 ; SYSTEM SERVICE QUEUE I/O FUNCTION DECISION TABLE SUBROUTINES
           0000 .69 ;
           0000 .70 ; MACRO LIBRARY CALLS
           0000 .71 ;
           0000 .72 ;
           0000 .73 ;           $ACBDEF           ;DEFINE ACB OFFSETS
;ROW49973     0000 .74 ;           $CCBDEF           ;DEFINE CCB OFFSETS
;ROW49973     0000 .1  ;           $DCDEF           ;DEFINE DEVICE CLASSES
;ROW49973     0000 .2  ;           $DEVDEF          ;DEFINE DEVICE CHARACTERISTICS
;ROW49973     0000 .3  ;           $IODEF           ;DEFINE I/O FUNCTION CODES
;ROW49973     0000 .4  ;           $IPLDEF          ;DEFINE SYSTEM IPLS
;ROW49973     0000 .5  ;           $IRPDEF          ;DEFINE IRP OFFSETS
;ROW49973     0000 .6  ;           $PCBDEF          ;DEFINE PCB VALUES
;ROW49973     0000 .7  ;           $PRDEF           ;DEFINE PROCESSOR REGISTERS
;ROW49973     0000 .79 ;           $$SSDEF          ;DEFINE SYSTEM STATUS VALUES
;ROW49973     0000 .80 ;           $UCBDEF          ;DEFINE UCB OFFSETS
;ROW49973     0000 .81 ;           $VADEF           ;DEFINE VIRTUAL ADDRESS FIELDS
;ROW49973     0000 .82 ;           $$SFDEF          ;DEFINE CALL FRAME
           0000 .83 ;
           0000 .84 ;
           0000 .85 ; LOCAL SYMBOLS
           0000 .86 ;
           0000 .87 ; ARGUMENT LIST OFFSET DEFINITIONS
           0000 .88 ;
           0000 .89 ;
00000000     0000 .90 P1=0           ;FIRST FUNCTION DEPENDENT PARAMETER
00000004     0000 .91 P2=4           ;SECOND FUNCTION DEPENDENT PARAMETER
00000008     0000 .92 P3=8           ;THIRD FUNCTION DEPENDENT PARAMETER
0000000C     0000 .93 P4=12          ;FOURTH FUNCTION DEPENDENT PARAMETER
00000010     0000 .94 P5=16          ;FIFTH FUNCTION DEPENDENT PARAMETER
00000014     0000 .95 P6=20          ;SIXTH FUNCTION DEPENDENT PARAMETER
```

```

0000 97 .SBTTL PLACE PROCESS IN I/O RESOURCE WAIT
0000 98 ;+
0000 99 ; EXE$IORSNWAIT - PLACE PROCESS IN I/O RESOURCE WAIT
0000 100 ;
0000 101 ; FUNCTIONAL DESCRIPTION:
0000 102 ;
0000 103 ; THIS ROUTINE IS USED BY FDT PROCEEDURES TO RE-START A QIO REQUEST
0000 104 ; AFTER A RESOURCE WAIT. THE CURRENT I/O IS CLEANED UP AND THE PRE-QIO
0000 105 ; STACK IS SET UP. THEN THE PROCESS IS PLACED IN THE WAIT STATE.
0000 106 ; IF THE PROCESS DOES NOT HAVE RESOURCE WIAT ENABLED, THE I/O IS ABORTED
0000 107 ; WITH A STATUS SPECIFIED BY THE CALLER.
0000 108 ;
0000 .1 ; **CAUTION** THIS ROUTINE IS NOT CALLED AT SYNCH SO THE RESOURCE MAY ALREADY
;WMC0001 0000 .2 ; HAVE BEEN DECLARED AVAILABLE. THIS ROUTINE SHOULD ONLY BE USED FOR RESOURCES
;WMC0001 0000 .3 ; WHICH ARE GUARANTEED TO BE PERIODICALLY (TIMESCHDL) DECLARED AVAILABLE.
;WMC0001 0000 .4 ;
;WMC0001 0000 .5 ;
;WMC0001 0000 .6 ; IMPLICIT INPUTS:
;WMC0001 0000 .7 ;
;ROW49973 0000 .8 ; CALLER MUST BE AT IPL=IPL$_ASTDEL
-4 0000 113 ;
0000 114 ; INPUTS:
0000 115 ;
0000 116 ; R0 = STATUS TO RETURN IF NO WAIT REQUESTED
0000 117 ; R1 = RESOURCE NUMBER TO WAIT FOR
0000 118 ; R3 = ADDRESS OF CURRENT PACKET
0000 119 ; R4 = ADDRESS OF THE CURRENT PCB
0000 120 ; R6 = ADDRESS OF CHANNEL CONTROL BLOCK
0000 121 ;
0000 122 ; OUTPUTS:
0000 123 ;
0000 124 ; R0,R1,R2,R3 ARE USED.
0000 125 ;
0000 126 ; CONTROL IS TRANSFERED TO EXE$ABORTIO IF NO RESOURCE WAIT
0000 127 ; HAS BEEN REQUESTED.
0000 128 ;
0000 129 ; OR TO SCH$WAIT IF RESOURCE WAIT IS REQUESTED.
0000 130 ;
0000 131 ; -
0000 132
0000 133 EXE$IORSNWAIT:: ;PLACE PROCESS IN RESOURCE WAIT
21 24 A4 0A E0 0000 134 BBS #PCB$_SSRWAIT,PCB$_L_STS(R4),50$ ;BR IF NO WAIT REQUEST
;ROW49973 51 DD 0005 .1 PUSHL R1 ;REMEMBER RESOURCE NUMBER
;ROW49973 00F5 30 0007 .2 BSBW BACKOUT_QIO ;CLEANUP QIO
;ROW49973 000A .3 SETIPL #IPL$_SYNCH ;SYNCHRONIZE WITH SCHEDULER DATABASE
;ROW49973 4C A4 6E 9A 000D .4 MOVZBL (SP),PCB$_EFWM(R4) ;SET UP WAIT MARKER
-3 0000'CF 8E E2 0011 138 BBS (SP)+,W^SCH$GGL_RESMASK,30$ ;INDICATE PROCESS IS WAITING
00 0016
5C 08 AD D0 0017 139 30$: MOVL SF$_SAVE_AP(FP),AP ;RESTORE PRE-QIO ARGUMENT LIST POINTER
5E 5D D0 001B 140 MOVL FP,SP ;CLEAN STACK BACK TO CALL FRAME
52 0000'CF 7E 001E 141 MOVAQ W^SCH$GQ_MWAIT,R2 ;ADDRESS WAIT LIST
FFDA' 31 0023 142 BRW SCH$WAIT ;PLACE PROCESS IN WAIT STATE
0026 143 ;
0026 144 ; NO RESOURCE WAIT REQUESTED - ABORT THE I/O
0026 145 ;
FFD7' 31 0026 146 50$: BRW EXE$ABORTIO ;

```

```

0029 148 .SBTTL ONE PARAMETER FUNCTION PROCESSING
0029 149 ;+
0029 150 ; EXE$ONEPARM - ONE PARAMETER FUNCTION PROCESSING
0029 151 ;
0029 152 ; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
0029 153 ; PROCESS A ONE PARAMETER FUNCTION THAT REQUIRES NO SPECIAL CHECKING.
0029 154 ;
0029 155 ; INPUTS:
0029 156 ;
0029 157 ; R0 = SCRATCH.
0029 158 ; R1 = SCRATCH.
0029 159 ; R2 = SCRATCH.
0029 160 ; R3 = ADDRESS OF I/O REQUEST PACKET.
0029 161 ; R4 = CURRENT PROCESS PCB ADDRESS.
0029 162 ; R5 = ASSIGNED DEVICE UCB ADDRESS.
0029 163 ; R6 = ADDRESS OF CCB.
0029 164 ; R7 = I/O FUNCTION CODE BIT NUMBER.
0029 165 ; R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
0029 166 ; R9 = SCRATCH.
0029 167 ; R10 = SCRATCH.
0029 168 ; R11 = SCRATCH.
0029 169 ; AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
0029 170 ;
0029 171 ; OUTPUTS:
0029 172 ;
0029 173 ; ***TBS***
0029 174 ;-
0029 175
0029 176 .ENABL LSB
0029 177 EXE$ONEPARM:: ;ONE PARAMETER FUNCTION PROCESSING
38 A3 6C D0 0029 178 MOVL P1(AP),IRP$L_MEDIA(R3) ;STORE PARAMETER IN MEDIA ADDRESS
03 11 002D 179 BRB 10$ ;

```



```
002F 181 .SBTTL ZERO PARAMETER FUNCTION PROCESSING
002F 182 ;+
002F 183 ; EXE$ZEROPARAM - ZERO PARAMETER FUNCTION PROCESSING
002F 184 ;
002F 185 ; THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
002F 186 ; PROCESS A ZERO PARAMETER FUNCTION THAT REQUIRES NO ADDITION CHECKING.
002F 187 ;
002F 188 ; INPUTS:
002F 189 ;
002F 190 ; R0 = SCRATCH.
002F 191 ; R1 = SCRATCH.
002F 192 ; R2 = SCRATCH.
002F 193 ; R3 = ADDRESS OF I/O REQUEST PACKET.
002F 194 ; R4 = CURRENT PROCESS PCB ADDRESS.
002F 195 ; R5 = ASSIGNED DEVICE UCB ADDRESS.
002F 196 ; R6 = ADDRESS OF CCB.
002F 197 ; R7 = I/O FUNCTION CODE BIT NUMBER.
002F 198 ; R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
002F 199 ; R9 = SCRATCH.
002F 200 ; R10 = SCRATCH.
002F 201 ; R11 = SCRATCH.
002F 202 ; AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
002F 203 ;
002F 204 ; OUTPUTS:
002F 205 ;
002F 206 ; ***TBS***
002F 207 ;-
002F 208
002F 209 EXE$ZEROPARM:: ;ZERO PARAMETER FUNCTION PROCESSING
38 A3 D4 002F 210 CLRL IRP$L_MEDIA(R3) ;CLEAR PARAMETER
FFCB' 31 0032 211 10$: BRW EXE$QIODRVPKT ;QUEUE I/O PACKET TO DRIVER
0035 212 .DSABL LSB
```

```

;ROW0224      0035 .1      .SBTTL LOCAL DISK VALID FUNCTION PROCESSING
;ROW0224      0035 .2 ;+
;ROW0224      0035 .3 ; EXE$LCLDSKVALID - LOCAL DISK VALID FUNCTION PROCESSING
;ROW0224      0035 .4 ;
;ROW0224      0035 .5 ;      This routine is called from the function decision table dispatcher to
;ROW0224      0035 .6 ;      process functions which affect the online count and local valid status
;ROW0224      0035 .7 ;      of a disk.
;ROW0224      0035 .8 ;
;ROW0224      0035 .9 ;      If the function is the first local pack acknowledge function
;ROW0224      0035 .10 ;      (UCB$V_LCL_VALID is clear), the online count, UCB$B_ONLCNT, is
;ROW0224      0035 .11 ;      incremented and UCB$V_LCL_VALID is set. If the online count was
;ROW0224      0035 .12 ;      previously zero, the I/O packet is queued to the driver for further
;ROW0259      0035 .13 ;      PACKACK processing. If the online count was not previously zero but
;ROW0259      0035 .14 ;      the UCB$V_VALID bit is clear, the I/O packet is also queued to the
;ROW0259      0035 .15 ;      driver for further processing.
;ROW0224      0035 .16 ;
;ROW0224      0035 .17 ;      If the function is the first local available or unload function
;ROW0224      0035 .18 ;      (UCB$V_LCL_VALID is set), the online count, UCB$B_ONLCNT, is
;ROW0224      0035 .19 ;      decremented and UCB$V_LCL_VALID is cleared. If the decremented online
;ROW0224      0035 .20 ;      count is zero, the I/O packet is queued to the driver for further
;ROW0224      0035 .21 ;      AVAILABLE or UNLOAD processing.
;ROW0224      0035 .22 ;
;ROW0224      0035 .23 ; INPUTS:
;ROW0224      0035 .24 ;
;ROW0224      0035 .25 ;      R0 = SCRATCH.
;ROW0224      0035 .26 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
;ROW0224      0035 .27 ;      R5 = ASSIGNED DEVICE UCB ADDRESS.
;ROW0224      0035 .28 ;      R7 = I/O FUNCTION CODE BIT NUMBER.
;ROW0224      0035 .29 ;
;ROW0224      0035 .30 ; OUTPUTS:
;ROW0224      0035 .31 ;
;ROW0224      0035 .32 ;      UCB$B_ONLCNT is altered to reflect the number of hosts which have set
;ROW0224      0035 .33 ;      the drive online (i.e. issued PACKACK functions to the drive).
;ROW0224      0035 .34 ;
;ROW0224      0035 .35 ;      UCB$V_LCL_VALID in UCB$L_STS is set for PACKACK functions and cleared
;ROW0224      0035 .36 ;      for AVAILABLE or UNLOAD functions.
;ROW0224      0035 .37 ; -
;ROW0224      0035 .38
;ROW0224      0035 .39 EXE$LCLDSKVALID:: ; LOCAL DISK VALID FUNCTION PROCESSING.
;ROW0224      0035 .40
;ROW0224      08 57 91 0035 .41 CMPB R7, #IO$_PACKACK ; Pack acknowledge function?
;ROW0224      0038 .42 BNEQ 50$ ; Branch if not a PACKACK.
;ROW0224      OE 64 A5 11 E2 003A .43 BBSS #UCB$V_LCL_VALID, - ; Is this the first local PACKACK?
;ROW0259      003F .44 UCB$L_STS(R5), 20$ ; Branch if not first local PACKACK.
;ROW0224      003F .45 SETIPL #IPL$_SCS ; Synchronize with the MSCP server.
;ROW0224      00AE C5 96 0042 .46 INCB UCB$B_ONLCNT(R5) ; Increment online count.
;ROW0224      01 00AE C5 91 0046 .47 CMPB UCB$B_ONLCNT(R5), #1 ; Is this the first cluster PACKACK?
;ROW0259      0048 .48 BEQL 30$ ; Branch if first cluster PACKACK.
;ROW0259      11 64 A5 0B E0 004D .49 20$: BBS #UCB$V_VALID, - ; Is the volume already valid?
;ROW0259      0052 .50 UCB$L_STS(R5), 80$ ; Branch if volume is already valid.
;ROW0259      0052 .51
;ROW0259      0052 .52
;ROW0259      FFAB' 31 0052 .53 30$: BRW EXE$QIODRVPKT ; For first cluster PACKACK, last
;ROW0259      0055 .54 ; cluster UNLOAD or AVAILABLE, or
;ROW0259      0055 .55 ; truly invalid volume, ask driver
;ROW0259      0055 .56 ; to really perform the function.
;ROW0224      0055 .57

```

```

;ROW0224          0055 .58 50$:
;ROW0224      09 64 A5 11 E5 0055 .59      BBCC      #UCB$V_LCL_VALID, -      ; UNLOAD and AVAILABLE come here.
;ROW0224          005A .60      UCB$L_STS(R5), 80$      ; First local UNLOAD or AVAILABLE?
;ROW0224          005A .61      SETIPL #IPL$_SCS      ; Branch if not first.
;ROW0224      00AE C5 97 005D .62      DECB      UCB$_ONLCNT(R5)      ; Synchronize with MSCP server.
;ROW0224          EF 13 0061 .63      BEQL      30$      ; Decrement online count.
;ROW0224          0063 .64      ; Branch if the online count is zero.
;ROW0224          0063 .65 80$:
;ROW0224          0063 .66      ; For requests which are not being
;ROW0224          50 01 3C 0063 .67      MOVZWL #SS$_NORMAL, R0      ; passed on to the driver.
;ROW0224          FF97 31 0066 .68      BRW      EXE$FINISHIOC      ; Set normal completion status.
;ROW0224          ; Finish I/O operation.

```

```

0069 214 .SBTTL READ AND WRITE FUNCTION PROCESSING
0069 215 ;+
0069 216 ; EXE$READ - READ FUNCTION PROCESSING
0069 217 ; EXE$WRITE - WRITE FUNCTION PROCESSING
0069 218 ; EXE$MODIFY - MODIFY FUNCTION PROCESSING
0069 219 ;
0069 220 ; THESE ROUTINES ARE CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO
0069 221 ; PROCESS A READ OR WRITE PHYSICAL OR LOGICAL FUNCTION.
0069 222 ; EXE$MODIFY IS USED FOR FUNCTIONS THAT READ AND WRITE MEMORY.
0069 223 ;
0069 224 ; INPUTS:
0069 225 ;
0069 226 ; R0 = SCRATCH.
0069 227 ; R1 = SCRATCH.
0069 228 ; R2 = SCRATCH.
0069 229 ; R3 = ADDRESS OF I/O REQUEST PACKET.
0069 230 ; R4 = CURRENT PROCESS PCB ADDRESS.
0069 231 ; R5 = ASSIGNED DEVICE UCB ADDRESS.
0069 232 ; R6 = ADDRESS OF CCB.
0069 233 ; R7 = I/O FUNCTION CODE BIT NUMBER.
0069 234 ; R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
0069 235 ; R9 = SCRATCH.
0069 236 ; R10 = SCRATCH.
0069 237 ; R11 = SCRATCH.
0069 238 ; AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
0069 239 ;
0069 240 ; OUTPUTS:
0069 241 ;
0069 242 ; ***TBS***
0069 243 ;-
0069 244
0069 245 .ENABL LSB
0069 246 EXE$MODIFY:: ;MODIFY FUNCTION PROCESSING
52 A1'AF DE 0069 247 MOVAL B^EXE$MODIFYLOCK,R2 ;SET ADDRESS OF BUFFER CHECK ROUTINE
04 04 11 006D 248 BRB 5$
006F 249 EXE$READ:: ;READ FUNCTION PROCESSING
52 9B'AF DE 006F 250 MOVAL B^EXE$READLOCK,R2 ;SET ADDRESS OF BUFFER CHECK ROUTINE
04 2A A3 01 E3 0073 251 5$: BBBC #IRP$V_FUNC,IRP$W_STS(R3),10$ ;SET READ FUNCTION STATUS
0078 252 EXE$WRITE:: ;WRITE FUNCTION PROCESSING
52 9E'AF DE 0078 253 MOVAL B^EXE$WRITELOCK,R2 ;SET ADDRESS OF BUFFER CHECK ROUTINE
3C A3 0C AC D0 007C 254 10$: MOVL P4(AP),IRP$B_CARCON(R3) ;INSERT CARRIAGE CONTROL BYTE
06 00 ED 0081 255 CMPZV #IRP$V_FCODE,#IRP$S_FCODE,- ;PHYSICAL I/O FUNCTION?
1F 20 A3 0084 256 IRP$W_FUNC(R3),#IO$_PHYSICAL ;
04 15 0087 257 BLEQ 20$ ;IF LEQ YES
15 A2 0089 258 SUBW #IO$_READLBLK-IO$_READPBLK,- ;CONVERT TO PHYSICAL FUNCTION
20 A3 008B 259 IRP$W_FUNC(R3) ;
51 04 AC D0 008D .1 20$: MOVL P2(AP),R1 ;GET NUMBER OF BYTES TO TRANSFER
05 13 0091 261 BEQL 30$ ;IF EQL NONE
50 6C D0 0093 262 MOVL P1(AP),R0 ;GET STARTING VIRTUAL ADDRESS OF TRANSFER
62 16 0096 263 JSB (R2) ;CHECK BUFFER AND LOCK IN MEMORY
FF65' 31 0098 264 30$: BRW EXE$QIODRVPKT ;QUEUE I/O PACKET TO DRIVER
009B 265 .DSABL LSB

```

;ROW0192  
-1



```

00A4 310      .SBTTL  READ AND WRITE BUFFER CHECK AND LOCK AND RETURN ROUTINES
00A4 311 ;+
00A4 312 ; EXE$READLOCKR - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK AND RETURN
00A4 313 ;          ON ERROR
00A4 314 ; EXE$WRITELOCKR - CHECK BUFFER FOR WRITE ACCESSIBILITY AND LOCK AND RETURN
00A4 315 ;          ON ERROR
00A4 316 ; EXE$MODIFYLOCKR - CHECK BUFFER FOR READ ACCESSIBILITY AND LOCK AND RETURN
00A4 317 ;          ON ERROR
00A4 318 ;
00A4 319 ; THESE ROUTINES ARE CALLED TO CHECK THE ACCESSIBILITY OF AN I/O BUFFER
00A4 320 ; AND TO LOCK THE BUFFER IN MEMORY FOR A DIRECT MEMORY TRANSFER.  IN
00A4 321 ; ADDITION, THESE ROUTINES PERFORM A COROUTINE CALL IF THERE IS AN ERROR
00A4 322 ; OR ANY PAGES HAVE TO BE FAULTED IN.  THE PURPOSE OF THE COROUTINE
00A4 323 ; CALL IS TO ALLOW THE CALLER TO PERFORM ANY NECESSARY CLEANUP BEFORE
00A4 324 ; THE QIO IS BACKED UP OR ABORTED.  THESE ROUTINES ARE TYPICALLY CALLED
00A4 325 ; BY DRIVERS THAT MUST LOCK MULTIPLE AREAS INTO MEMORY.  SINCE THESE
00A4 326 ; ROUTINES CANNOT UNLOCK AREAS PREVIOUSLY LOCKED, THE COROUTINE CALL ALLOWS
00A4 327 ; THE CALLER (THE DRIVER) TO UNLOCK PREVIOUSLY LOCKED AREAS (AND PERFORM
00A4 328 ; ANY OTHER CLEANUP) AND THEN RETURN HERE TO BACK UP OR ABORT THE I/O.
00A4 329 ;
00A4 330 ; EXE$MODIFYLOCKR IS USED WHEN THE BUFFER WILL BE READ AND WRITTEN BY THE
00A4 331 ; I/O DEVICE.  IT DISABLES AN OPTIMIZATION IN MMG$IOLOCK WHICH IS USED
00A4 332 ; WHEN THE BUFFER IS ONLY WRITTEN.
00A4 333 ;
00A4 334 ; INPUTS:
00A4 335 ;
00A4 336 ;          R0 = STARTING ADDRESS OF I/O BUFFER.
00A4 337 ;          R1 = LENGTH OF BUFFER IN BYTES.
00A4 338 ;          R4 = CURRENT PROCESS PCB ADDRESS.
00A4 339 ;          R6 = ADDRESS OF CCB.
00A4 340 ;
00A4 341 ; OUTPUTS:
00A4 342 ;
00A4 343 ;          THE I/O BUFFER IS CHECKED FOR THE PROPER ACCESSIBILITY.  IF THE
00A4 344 ; CHECK SUCCEEDS, THEN THE BUFFER IS LOCKED IN MEMORY AND THE STARTING
00A4 345 ; ADDRESS OF THE PAGE TABLE ENTRIES THAT MAP THE TRANSFER IS STORED
00A4 346 ; IN THE I/O PACKET.
00A4 347 ;
00A4 348 ;          R0 = RETURN CODE
00A4 349 ;
00A4 350 ;          NOTE THAT IF THERE ARE NO ERRORS AND NO PAGES HAVE TO BE FAULTED
00A4 351 ; IN, THEN THESE ROUTINES RETURN NORMALLY.  HOWEVER, IF THERE IS AN
00A4 352 ; ERROR OR A PAGE HAS TO BE FAULTED IN, THEN THE CALLER IS CALLED
00A4 353 ; BY A COROUTINE CALL.  THE CALLER'S RSB THEN RETURNS HERE WHERE
00A4 354 ; THE QIO IS EITHER BACKED UP OR ABORTED.  NOTE THAT IN THIS CASE
00A4 355 ; THE CALLER'S ERROR HANDLING CODE MUST PRESERVE ALL REGISTERS,
00A4 356 ; INCLUDING R0 AND R1.
00A4 357 ;-
00A4 358      .ENABL  LSB
00A4 359 EXE$MODIFYLOCKR::          ;CHECK BUFFER FOR MODIFY FUNCTION AND LOCK
50 DD 00A4 360      PUSHL  R0          ;SAVE STARTING ADDRESS OF BUFFER
009D 30 00A6 361      BSBW   EXE$READCHKR          ;CHECK BUFFER FOR READ FUNCTION
52 04 C8 00A9 362      BISL   #4,R2          ;DISABLE OPTIMIZATION IN MMG$IOLOCK
0C 11 00AC 363      BRB    10$
00AE 364
00AE 365 EXE$READLOCKR::          ;CHECK BUFFER FOR READ FUNCTION AND LOCK
50 DD 00AE 366      PUSHL  R0          ;SAVE STARTING ADDRESS OF BUFFER

```

```

0093 30 00B0 367          BSBW  EXE$READCHKR          ;CHECK BUFFER FOR READ FUNCTION
05   11 00B3 368          BRB    10$
      00B5 369
      00B5 370 EXE$WRITELOCKR: ;CHECK BUFFER FOR WRITE FUNCTION AND LOCK
50   DD 00B5 371          PUSHL  R0                ;SAVE STARTING ADDRESS OF BUFFER
00EA 30 00B7 372          BSBW  EXE$WRITECHKR          ;CHECK BUFFER FOR WRITE FUNCTION
1A 50 E9 00BA 373 10$:   BLBC  R0,15$             ;BRANCH IF ERROR
50   50 8ED0 00BD 374          POPL  R0                ;RESTORE STARTING ADDRESS OF BUFFER
50 FE00 8F AB 00C0 375          BICW3 #^C<VASM_BYTE>,R0,IRP$W_BOFF(R3) ;SET BYTE OFFSET IN PAGE
30 A3 00C5
53   DD 00C7 376          PUSHL  R3                ;SAVE ADDRESS OF I/O PACKET
FF34' 30 00C9 377          BSBW  MMG$IOLOCK          ;LOCK PAGES FOR I/O
53   8E D0 00CC 378          MOVL  (SP)+,R3          ;RETRIEVE ADDRESS OF I/O PACKET
08 50 E9 00CF 379          BLBC  R0,20$             ;IF LBC LOCK FAILURE
2C A3 51 D0 00D2 380          MOVL  R1,IRP$L_SVAPTE(R3) ;INSERT ADDRESS OF FIRST PTE IN PACKET
      05 00D6 381          RSB
5E   04 C0 00D7 382 15$:   ADDL  #4,SP                ;THROW AWAY OLD R0
      9E 16 00DA 383 20$:   JSB   @(SP)+            ;COROUTINE CALL TO CLEANUP
50   D5 00DC 384          TSTL  R0                ;ERRORS ENCOUNTERED?
1C   12 00DE 385          BNEQ  50$             ;IF NEQ YES
51   DD 00E0 386          PUSHL  R1                ;SAVE VIRTUAL ADDRESS OF PAGE TO FAULT
1B   10 00E2 387          BSBB  BACKOUT_QIO      ;CLEANUP QIO
02   BA 00E4 388          POPR  #^M<R1>         ;RETRIEVE VIRTUAL ADDRESS OF PAGE TO FAULT
5E   5D D0 00E6 389          MOVL  FP,SP            ;TRIM STACK BACK TO CHANGE MODE FRAME
5C 08 AD 7D 00E9 390          MOVQ  8(FP),AP         ;RESTORE USER ARGUMENT AND FRAME POINTERS
5E   00' C0 00ED 391          ADDL  S^#EXE$C_CMSTKSZ,SP ;REMOVE CHANGE MODE CALL FRAME FROM STACK
50 8E 04 C3 00F0 392          SUBL3 #4,(SP)+,R0     ;CALCULATE RESTART ADDRESS
      F8'AF 9F 00F4 393          PUSHAB B^40$         ;SET NEW RETURN ADDRESS
      02 00F7 394          REI
61   95 00F8 395 40$:   TSTB  (R1)                ;FAULT USER BUFFER AGAIN
60   17 00FA 396          JMP   (R0)            ;REPEAT SYSTEM SERVICE
FF01' 31 00FC 397 50$:   BRW   EXE$ABORTIO          ;ABORT I/O REQUEST
      00FF 398          .DSABL  LSB

```

```

00FF 400          .SBTTL  BACKOUT A QIO
00FF 401 ;+
00FF 402 ; BACKOUT_QIO - BACKOUT A QIO
00FF 403 ;
00FF 404 ; THIS ROUTINE IS CALLED TO BACKOUT A QIO.  IT DECREASES THE CHANNEL I/O
00FF 405 ; COUNT, INCREMENTS THE DIRECT OR BUFFERED I/O COUNT, DEALLOCATES THE
00FF 406 ; DIAGNOSTIC BUFFER (IF PRESENT), OPTIONALLY INCREMENTS THE AST COUNT, AND
00FF 407 ; FINALLY DEALLOCATES THE IRP.
00FF 408 ;
00FF 409 ; INPUTS:
00FF 410 ;
00FF 411 ;      R3 = ADDRESS OF I/O REQUEST PACKET
00FF 412 ;      R4 = CURRENT PROCESS PCB ADDRESS
00FF 413 ;      R6 = ADDRESS OF CCB
00FF 414 ;
00FF 415 ; OUTPUTS:
00FF 416 ;
00FF 417 ;      R0 - R3 = CLOBBERED
00FF 418 ;
00FF 419 ;-
00FF 420
00FF 421 BACKOUT_QIO:                                ;BACKOUT A QIO
;SSA00002      FEFE' 30 00FF .1 BSBW PMS$ABORT_RQ ;RECORD ABORT IF I/O MONITORING ENABLED
              0A A6 B7 0102 422 DECW CCB$W_IOC(R6) ;DECREMENT CHANNEL I/O COUNT
05 2A A3 00 E1 0105 423 BBC #IRP$V_BUFIO,IRP$W_STS(R3),10$ ;BR IF NOT BUFFERED I/O
              3A A4 B6 010A 424 INCW PCB$W_BIOCNT(R4) ;ADJUST COUNT OF BUFFERED I/O
              03 11 010D 425 BRB 20$ ;CONTINUE
              3E A4 B6 010F 426 10$: INCW PCB$W_DIOCNT(R4) ;ADJUST DIRECT I/O COUNT
0C 2A A3 07 E1 0112 427 20$: BBC #IRP$V_DIAGBUF,IRP$W_STS(R3),30$ ;BR. IF NO DIAGNOSTIC BUFFER
50 4C A3 00 D0 0117 428 MOVL IRP$L_DIAGBUF(R3),R0 ;GET ADDRESS OF DIAGNOSTIC BUFFER
              53 DD 011B 429 PUSHL R3 ;SAVE R3
              FEEO' 30 011D 430 BSBW EXE$DEANONPAGED ;DEALLOCATE DIAGNOSTIC BUFFER
              53 BED0 0120 431 POPL R3 ;RESTORE R3
03 0B A3 06 E1 0123 432 30$: BBC #ACB$V_QUOTA,IRP$B_RMOD(R3),40$ ;BR IF AST NOT REQUESTED
              38 A4 B6 0128 433 INCW PCB$W_ASTCNT(R4) ;ADJUST AST COUNT
50 53 D0 012B 434 40$: MOVL R3,R0 ;DEALLOCATE PACKET
              FECE' 30 012E 435 BSBW EXE$DEANONPAGED ;
              05 0131 436 RSB

```



```

0132 438          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
0132 439 ;+
0132 440 ; EXE$READCHK - CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION
0132 441 ;
0132 442 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A READ I/O
0132 443 ; FUNCTION.
0132 444 ;
0132 445 ; INPUTS:
0132 446 ;
0132 447 ;         R0 = ADDRESS OF BUFFER.
0132 448 ;         R1 = SIZE OF TRANSFER IN BYTES.
0132 449 ;         R3 = ADDRESS OF I/O REQUEST PACKET.
0132 450 ;
0132 451 ; OUTPUTS:
0132 452 ;
0132 453 ;         IF BUFFER IS NOT WRITE ACCESSIBLE, THEN THE I/O REQUEST IS TERM-
0132 454 ;         INATED VIA EXE$IOFINISH WITH A STATUS OF SS$_ACCVIO.
0132 455 ;
0132 456 ;         IF BUFFER IS WRITE ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
0132 457 ;         TURNED:
0132 458 ;
0132 459 ;         R0 = ADDRESS OF BUFFER.
0132 460 ;         R1 = SIZE OF TRANSFER IN BYTES.
0132 461 ;         R2 = READ FUNCTION INDICATOR (1).
0132 462 ;         R3 = ADDRESS OF I/O REQUEST PACKET.
0132 463 ;
0132 464 ;         IRP$_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
0132 465 ;         IRP$_FUNC(R3) = READ.
0132 466 ;-
0132 467
0132 468          .ENABL  LSB
0132 469 EXE$READCHK:;
50 DD 0132 470          PUSHL  R0          ;CHECK BUFFER FOR READ FUNCTION
10 10 0134 471          BSBB   EXE$READCHKR ;SAVE ADDRESS OF BUFFER
04 11 0136 472          BRB    10$         ;CHECK BUFFER

```

```

0138 474          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
0138 475 ;+
0138 476 ; EXE$WRITECHK - CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION
0138 477 ;
0138 478 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A WRITE I/O
0138 479 ; FUNCTION.
0138 480 ;
0138 481 ; INPUTS:
0138 482 ;
0138 483 ;         R0 = ADDRESS OF BUFFER.
0138 484 ;         R1 = SIZE OF TRANSFER IN BYTES.
0138 485 ;         R3 = ADDRESS OF I/O REQUEST PACKET.
0138 486 ;
0138 487 ; OUTPUTS:
0138 488 ;
0138 489 ;         IF BUFFER IS NOT READ ACCESSIBLE, THEN THE I/O REQUEST IS TERM-
0138 490 ;         INATED VIA EXE$IOFINISH WITH A STATUS OF SS$_ACCVIO.
0138 491 ;
0138 492 ;         IF BUFFER IS READ ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
0138 493 ;         TURNED:
0138 494 ;
0138 495 ;         R0 = ADDRESS OF BUFFER.
0138 496 ;         R1 = SIZE OF TRANSFER IN BYTES.
0138 497 ;         R2 = WRITE FUNCTION INDICATOR (0).
0138 498 ;         R3 = ADDRESS OF I/O REQUEST PACKET.
0138 499 ;
0138 500 ;         IRP$_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
0138 501 ;         IRP$_FUNC(R3) = WRITE.
0138 502 ;-
0138 503
0138 504 EXE$WRITECHK::          ;CHECK BUFFER FOR WRITE FUNCTION
50 DD 0138 505          PUSHL  R0          ;SAVE ADDRESS OF BUFFER
68 10 013A 506          BSBB   EXE$WRITECHKR ;CHECK BUFFER
03 50 E8 013C 507 10$:  BLBS   R0,20$      ;BRANCH IF SUCCESS
FEBE' 31 013F 508          BRW   EXE$ABORTIO ;ABORT I/O
50 8ED0 0142 509 20$:  POPL  R0          ;RESTORE ADDRESS OF BUFFER
05 0145 510          RSB
0146 511          .DSABL  LSB

```

```

0146 513          .SBTTL  CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
0146 514 ;+
0146 515 ; EXE$READCHKR - CHECK BUFFER ACCESSIBILITY FOR READ FUNCTION AND RETURN
0146 516 ;
0146 517 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A READ I/O
0146 518 ; FUNCTION.  STATUS IS RETURNED IN R0.
0146 519 ;
0146 520 ; INPUTS:
0146 521 ;
0146 522 ;      R0 = ADDRESS OF BUFFER.
0146 523 ;      R1 = SIZE OF TRANSFER IN BYTES.
0146 524 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
0146 525 ;
0146 526 ; OUTPUTS:
0146 527 ;
0146 528 ;      IF THE BUFFER IS NOT WRITE ACCESSIBLE, THEN THE FOLLOWING
0146 529 ;      VALUE IS RETURNED:
0146 530 ;
0146 531 ;      R0 = SS$_ACCVIO
0146 532 ;
0146 533 ;      IF BUFFER IS WRITE ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
0146 534 ;      TURNED:
0146 535 ;
0146 536 ;      R0 = SS$_NORMAL
0146 537 ;      R1 = SIZE OF TRANSFER IN BYTES.
0146 538 ;      R2 = READ FUNCTION INDICATOR (1).
0146 539 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
0146 540 ;
0146 541 ;      IRP$_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
0146 542 ;      IRP$_FUNC(R3) = READ.
0146 543 ;-
0146 544
0146 545          .ENABL  LSB
0146 546 EXE$READCHKR: ; CHECK BUFFER FOR READ FUNCTION
32 A3 51 D0 0146 547      MOVL  R1,IRP$_BCNT(R3) ; SAVE R1
0146 548      BSBB  10$ ; CHECK ACCESS
51 32 A3 D0 014C 549      MOVL  IRP$_BCNT(R3),R1 ; RESTORE R1
0146 550      BLBC  R0,5$ ; IF LBC, NO ACCESS
2A A3 02 A8 0153 551      BISW  #IRP$_FUNC,IRP$_STS(R3) ; SET READ FUNCTION
0146 552      MOVL  #1,R2 ; SET READ FUNCTION INDICATOR
52 01 D0 0157 552
0146 553 5$: RSB
0146 554
0146 555 10$: ADDL  R0,R1 ; ENDING ADDRESS OF BUFFER
50 01FF 8F AA 015E 556      BICW  #VASM_BYTE,R0 ; TRUNCATE TO START OF PAGE
0146 557      SUBL  R0,R1 ; CALCULATE LENGTH OF BUFFER TO PROBE
52 FE00 8F 32 0166 558      CVTLW #-^X200,R2 ; SET ADDRESS ADJUSTMENT CONSTANT
0146 559 15$: CVTLW R1,R1 ; GREATER THAN 32K?
0146 560      BVS  30$ ; IF VS, YES; CHECK BY CHUNKS
0146 561
0146 562 20$: IFNOWRT R1,(R0),ACCVIO ; CAN ENDS OF USER'S BUFFER BE WRITTEN?
0146 563      SUBL  R2,R0 ; CALCULATE VA OF NEXT PAGE
51 6142 3E 0179 564      MOVAW (R1)[R2],R1 ; CALCULATE NEW LENGTH
0146 565      BGTR  20$ ; IF GTR THEN MORE TO TEST
50 01 3C 017F 566      MOVZWL #SS$_NORMAL,R0 ; INDICATE SUCCESS
0146 567      RSB ; AND RETURN
0146 568
7E 50 7D 0183 569 30$: MOVQ  R0,-(SP) ; SAVE CURRENT VALUES ON STACK

```



```

01A4 587 .SBTTL CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
01A4 588 ;+
01A4 589 ; EXE$WRITECHKR - CHECK BUFFER ACCESSIBILITY FOR WRITE FUNCTION AND RETURN
01A4 590 ;
01A4 591 ; THIS ROUTINE IS CALLED TO CHECK BUFFER ACCESSIBILITY FOR A WRITE I/O
01A4 592 ; FUNCTION. STATUS IS RETURNED IN R0
01A4 593 ;
01A4 594 ; INPUTS:
01A4 595 ;
01A4 596 ; R0 = ADDRESS OF BUFFER.
01A4 597 ; R1 = SIZE OF TRANSFER IN BYTES.
01A4 598 ; R3 = ADDRESS OF I/O REQUEST PACKET.
01A4 599 ;
01A4 600 ; OUTPUTS:
01A4 601 ;
01A4 602 ; IF BUFFER IS NOT READ ACCESSIBLE, THEN THE FOLLOWING VALUE IS
01A4 603 ; RETURNED:
01A4 604 ;
01A4 605 ; R0 = SS$_ACCVIO
01A4 606 ;
01A4 607 ; IF BUFFER IS READ ACCESSIBLE, THEN THE FOLLOWING VALUES ARE RE-
01A4 608 ; TURNED:
01A4 609 ;
01A4 610 ; R0 = SS$_NORMAL
01A4 611 ; R1 = SIZE OF TRANSFER IN BYTES.
01A4 612 ; R2 = WRITE FUNCTION INDICATOR (0).
01A4 613 ; R3 = ADDRESS OF I/O REQUEST PACKET.
01A4 614 ;
01A4 615 ; IRP$L_BCNT(R3) = SIZE OF TRANSFER IN BYTES.
01A4 616 ; IRP$W_FUNC(R3) = WRITE.
01A4 617 ;-
01A4 618
01A4 619 EXE$WRITECHKR:: ;CHECK BUFFER FOR WRITE FUNCTION
32 A3 51 D0 01A4 620 MOVL R1,IRP$L_BCNT(R3) ; SAVE R1
0A 10 01A8 621 BSBB 40$ ; CHECK ACCESS
51 32 A3 D0 01AA 622 MOVL IRP$L_BCNT(R3),R1 ; RESTORE R1
02 50 E9 01AE 623 BLBC R0,35$ ; IF LBC, NO ACCESS
52 D4 01B1 624 CLRL R2 ; SET WRITE FUNCTION INDICATOR
05 01B3 625 35$: RSB
01B4 626
51 51 50 C0 01B4 627 40$: ADDL R0,R1 ; ENDING ADDRESS OF BUFFER
50 01FF 8F AA 01B7 628 BICW #VA$M_BYTE,R0 ; TRUNCATE TO START OF PAGE
51 50 C2 01BC 629 SUBL R0,R1 ; CALCULATE LENGTH OF BUFFER TO PROBE
52 FE00 8F 32 01BF 630 CVTWL #-^X200,R2 ; SET ADDRESS ADJUSTMENT CONSTANT
51 51 F7 01C4 631 45$: CVTLW R1,R1 ; GREATER THAN 32k?
13 1D 01C7 632 BVS 60$ ; IF VS, YES; CHECK BY CHUNKS
01C9 633
01C9 634 50$: IFNORD R1,(R0),ACCVIO ; CAN ENDS OF USER'S BUFFER BE READ?
50 52 C2 01CF 635 SUBL R2,R0 ; CALCULATE VA OF NEXT PAGE
51 6142 3E 01D2 636 MOVAV (R1)[R2],R1 ; CALCULATE NEW LENGTH
F1 14 01D6 637 BGTR 50$ ; IF GTR THEN MORE TO TEST
50 01 3C 01D8 638 MOVZWL #SS$_NORMAL,R0 ; INDICATE SUCCESS
05 01DB 639 RSB ; AND RETURN
01DC 640
7E 50 7D 01DC 641 60$: MOVQ R0,-(SP) ; SAVE CURRENT VALUES ON STACK
51 7E00 8F 3C 01DF 642 MOVZWL #^X7E00,R1 ; SIZE OF CHUNK USED STEPPING THRU BUF.
01E4 643 ; (32K - 1 PAGE)

```

6E	51	C0	01E4	644	ADDL	R1,(SP)	; ADVANCE ADDRESS BY THIS AMOUNT
04 AE	51	C2	01E7	645	SUBL	R1,4(SP)	; DECREASE COUNT
		DC	10	01EB	646	BSBB	50\$
							; PROBE CHUNK
	AC	50	E9	01ED	647	BLBC	R0,ACCVI01
							; IF LBC, NO ACCESS
50	8E	7D	01F0	648	MOVQ	(SP)+,R0	; POP PRE-ADJUSTED VALUES OFF STACK
		CF	11	01F3	649	BRB	45\$
							; SEE IF LENGTH NOW LT 32K
			01F5	650	.DSABL	LSB	

```

01F5 652          .SBTTL SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
01F5 653 ;+
01F5 654 ; EXE$SETCHAR - SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS (AT FDT LEVEL)
01F5 655 ;
01F5 656 ; THIS ROUTINE PLACES THE NEW CHARACTERISTICS SPECIFIED BY THE QUADWORD POINTED
01F5 657 ; TO BY P1 INTO THE SECOND AND THIRD LONGWORDS OF THE DEVICE UCB.
01F5 658 ;
01F5 659 ; INPUTS:
01F5 660 ;
01F5 661 ;          R0 = SCRATCH.
01F5 662 ;          R1 = SCRATCH.
01F5 663 ;          R2 = SCRATCH.
01F5 664 ;          R3 = ADDRESS OF I/O REQUEST PACKET.
01F5 665 ;          R4 = CURRENT PROCESS PCB ADDRESS.
01F5 666 ;          R5 = ASSIGNED DEVICE UCB ADDRESS.
01F5 667 ;          R6 = ADDRESS OF CCB.
01F5 668 ;          R7 = I/O FUNCTION CODE BIT NUMBER.
01F5 669 ;          R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
01F5 670 ;          R9 = SCRATCH.
01F5 671 ;          R10 = SCRATCH.
01F5 672 ;          R11 = SCRATCH.
01F5 673 ;          AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
01F5 674 ;
01F5 675 ; OUTPUTS:
01F5 676 ;
01F5 677 ;          THE CHARACTERISTICS SPECIFIED BY THE QUADWORD POINTER TO BY P1 ARE STORED
01F5 678 ;          IN THE SECOND AND THIRD LONGWORDS OF THE DEVICE UCB.
01F5 .1 ;
01F5 .2 ; COMPLETION CODES:
01F5 .3 ;
01F5 .4 ;          SS$_NORMAL - SUCCESSFUL
01F5 .5 ;          SS$_ACCVIO - BUFFER ACCESS VIOLATION
01F5 .6 ;          SS$_ILLIOFUNC - FUNCTION IS ILLEGAL ON DISK DEVICES
01F5 .7 ;-
01F5 .8 ;
01F5 .9          .ENABL LSB
01F5 .10 EXE$SETCHAR: ;
01F5 .11          BSBB CHECK_SET ;SET DEVICE MODE AND CHARACTERISTICS
01F5 685          CMPL #IO$_SETMODE,R7 ;IS THIS SET FUNCTION VAILD?
01F5 686          BEQL 10$ ;SET MODE FUNCTION?
01F5 687          MOVW (R1),UCB$_DEVCLASS(R5) ;IF EQL YES
01F5 688 10$: MOVW 2(R1),UCB$_DEVBUFSIZ(R5) ;SET DEVICE TYPE AND CLASS
01F5 689          MOVL 4(R1),UCB$_DEVDEPEND(R5) ;SET DEFAULT BUFFER SIZE
01F5 690          BRB 20$ ;SET DEVICE CHARACTERISTICS
;

```

```

;ROW49577
;ROW49577
;ROW49577
;ROW49577
;ROW49577
;ROW49577
;PRD0030
;PRD0030
;PRD0030
;PRD0030
;ROW49577
-6

```

```

          1E 10
57 23 D1
          04 13
40 A5 61 B0
42 A5 02 A1 B0
44 A5 04 A1 D0
          2A 11 020A

```

```

020C 692 .SBTTL SET DEVICE MODE AND CHARACTERISTICS FUNCTIONS
020C 693 ;+
020C 694 ; EXE$SETMODE - SET DEVICE CHARACTERISTICS AND MODE
020C 695 ;
020C 696 ; FUNCTIONAL DESCRIPTION:
020C 697 ;
020C 698 ; THIS ROUTINE PLACES THE NEW CHARACTERISTICS SPECIFIED BY P1 INTO
020C 699 ; THE I/O PACKET FOR INSERTION INTO THE UCB WHEN THE UNIT IS IDLE.
020C 700 ; THE INPUT DATA IS IN THE FORM RETURNED BY $GTCHAN. THE SPECIFIED BUFFER
020C 701 ; IS ASSUMED TO BE 12 BYTES IN LENGTH. THE P2 LENGTH SPECIFIER IS IGNORED.
020C 702 ;
020C 703 ; THE NEW CHARACTERISTICS ARE PLACED IN IRP$L_MEDIA/MEDIA+4 AND THE
020C 704 ; PACKET IS QUEUED VIA EXE$QIODRVPKT.
020C 705 ;
020C 706 ; INPUTS:
020C 707 ;
020C 708 ; R3 = I/O PACKET ADDRESS
020C 709 ; R4 = CURRENT PCB
020C 710 ; R5 = ACB ADDRESS
020C 711 ; R6 = ASSIGNED CCB ADDRESS
020C 712 ; AP = ADDRESS OF THE QIO ARGUMENT P1
020C 713 ;
020C 714 ; OUTPUTS:
020C 715 ;
020C 716 ; R0 = STATUS OF THE OPERATION
020C 717 ; R3+ ARE PRESERVED.
020C 718 ;
020C 719 ; COMPLETION CODES:
020C 720 ;
020C 721 ; $$$_NORMAL - SUCCESSFUL
020C 722 ; $$$_ACCVIO - BUFFER ACCESS VIOLATION
;ROW49577 020C .1 ; $$$_ILLIOFUNC - FUNCTION IS ILLEGAL ON DISK DEVICES
;PRD0030 020C .2 ; -
;PRD0030 020C .3
;PRD0030 020C .4 EXE$SETMODE:: ;SET DEVICE MODE AND CHARACTERISTICS
;ROW49577 07 10 020C .5 BSBB CHECK_SET ;IS THIS SET FUNCTION VAILD?
;PRD0030 38 A3 61 7D 020E .6 MOVQ (R1),IRP$L_MEDIA(R3) ;INSERT CHARACTERISTICS IN I/O PACKET
;PRD0030 FDEB' 31 0212 .7 BRW EXE$QIODRVPKT ;QUEUE THE PACKET
;PRD0030 0215 .8
;ROW49577 0215 .9 CHECK_SET:
;ROW49577 40 A5 01 91 0215 .10 CMPB #DC$_DISK,UCB$B_DEVCLASS(R5) ; Is this a disk device?
;ROW49577 0A 13 0219 .11 BEQL 91$ ; Branch if disk; they can't be set.
;ROW49577 51 6C D0 021B .12 MOVL P1(AP), R1 ; Get buffer address.
;ROW49577 021E .13 IFNORD #8, (R1), 93$ ; Branch if no read access to buffer.
;ROW49577 05 0224 .14 RSB ; Else, all is ok; return to caller.
;ROW49577 0225 .15
;ROW49577 50 00F4 8F 3C 0225 .16 91$: MOVZWL #$$$_ILLIOFUNC, R0 ; Setup illegal I/O function status.
;ROW49577 03 11 022A .17 BRB 99$ ; or
;ROW49577 50 0C 3C 022C .18 93$: MOVZWL #$$$_ACCVIO, R0 ; Setup access violation status.
;ROW49577 FDCE' 31 022F .19 99$: BRW EXE$ABORTIO ; Then blow the I/O request away.

```



```

0232 735      .SBTTL  SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
0232 736 ;+
0232 737 ; EXE$SENSEMODE - SENSE DEVICE MODE AND CHARACTERISTICS FUNCTIONS
0232 738 ;
0232 739 ; THIS ROUTINE OBTAINS THE CURRENT DEVICE MODE/CHARACTERISTICS FROM THE DEVICE
0232 740 ; DEPENDENT CHARACTERISTICS LONGWORD IN THE UCB AND IMMEDIATELY COMPLETES THE
0232 741 ; I/O OPERATION WITH THE SECOND LONGWORD OF THE FINAL I/O STATUS EQUAL TO THE
0232 742 ; DEVICE DEPENDENT CHARACTERISTICS.
0232 743 ;
0232 744 ; INPUTS:
0232 745 ;
0232 746 ;      R0 = SCRATCH.
0232 747 ;      R1 = SCRATCH.
0232 748 ;      R2 = SCRATCH.
0232 749 ;      R3 = ADDRESS OF I/O REQUEST PACKET.
0232 750 ;      R4 = CURRENT PROCESS PCB ADDRESS.
0232 751 ;      R5 = ASSIGNED DEVICE UCB ADDRESS.
0232 752 ;      R6 = ADDRESS OF CCB.
0232 753 ;      R7 = I/O FUNCTION CODE BIT NUMBER.
0232 754 ;      R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
0232 755 ;      R9 = SCRATCH.
0232 756 ;      R10 = SCRATCH.
0232 757 ;      R11 = SCRATCH.
0232 758 ;      AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
0232 759 ;
0232 760 ; OUTPUTS:
0232 761 ;
0232 762 ;      THE DEVICE DEPENDENT CHARACTERISTICS ARE OBTAINED FROM THE UCB AND
0232 763 ;      THE I/O IS COMPLETED WITH THE SECOND I/O STATUS LONGWORD EQUAL TO THE
0232 764 ;      DEVICE CHARACTERISTICS.
0232 765 ;-
0232 766
0232 767 EXE$SENSEMODE::
51 44 A5 D0 0232 768      MOVL  UCB$L_DEVDEPEND(R5),R1      ;SENSE DEVICE MODE/CHARACTERISTICS
50 01 3C 0236 769 20$:  MOVZWL #SS$_NORMAL,R0      ;GET DEVICE DEPENDENT CHARACTERISTICS
      FDC4' 31 0239 770      BRW  EXE$FINISHIO      ;SET NORMAL COMPLETION STATUS
023C 771      .DSABL  LSB      ;FINISH I/O OPERATION

```

```

023C 773 .SBTTL CARRIAGE CONTROL INTERPRETATION
023C 774 ;+
023C 775 ; EXE$CARRIAGE - INTERPRET CARRIAGE CONTROL SPECIFIER
023C 776 ;
023C 777 ; FUNCTIONAL DESCRIPTION:
023C 778 ;
023C 779 ; THIS ROUTINE IS USED BY THE LINE PRINTER DRIVER AND THE TERMINAL
023C 780 ; DRIVER TO INTERPRET THE CARRIAGE CONTROL SPECIFIER IN IRP$B_CARCON .
023C 781 ; NOTE THAT IRP$B_CARCON IS USED AS A LONGWORD!
023C 782 ;
023C 783 ; THE SPECIFIER IS AS FOLLOWS:
023C 784 ;
023C 785 ; .BYTE 1 -- FORTRAN CARRIAGE CONTROL CHARACTER IF NOT 0
023C 786 ; .BYTE 2 -- ***** IGNORED *****
023C 787 ; .BYTE 3 -- PREFIX CARRIAGE CONTROL
023C 788 ; .BYTE 4 -- SUFFIX CARRIAGE CONTROL
023C 789 ;
023C 790 ; THE PRE/SUF FIELDS ARE AS FOLLOWS
023C 791 ;
023C 792 ; IF BIT 7=0 THEN BITS 6-0 ARE THE NUMBER OF NEWLINES TO INSERT.
023C 793 ; IF BIT 7=1 AND BIT 6=0 THEN BITS 4-0 ARE THE ASCII CHARACTER TO
023C 794 ; OUTPUT. ASCII SET C0 OR C1 IS SPECIFIED BY BIT 5.
023C 795 ; IF BIT 7=1 AND BIT 6=1 THEN BITS 5-0 ARE THE PRINTER CHANNEL NUMBER
023C 796 ;
023C 797 ; ASCII SET C0 IS ASSUMED AND BIT 6 IS IGNORED IF BIT 7=0.
023C 798 ;
023C 799 ; INPUTS:
023C 800 ;
023C 801 ; R3 = ADDRESS OF THE I/O PACKET
023C 802 ; R5 = ADDRESS OF THE UCB
023C 803 ;
023C 804 ; OUTPUTS:
023C 805 ;
023C 806 ; IRP$B_CARCON IS SET UP TO REFLECT THE PRE/SUF CHARACTERS TO SEND.
023C 807 ;
023C 808 ; BYTE 0 = NUMBER OF CHARACTERS TO SEND
023C 809 ; BYTE 1 = CHARACTER, IF 0 THEN NEWLINE
023C 810 ;
023C 811 ; IRP$B_CARCON+2 HAS THE SUFFIX CONTROL.
023C 812 ;
023C 813 ; R0,R1 ARE USED.
023C 814 ;
023C 815 ;-
023C 816 ;
023C 817 ; LOCAL DATA TABLE
023C 818 ;
023C 819 CCTABLE:
OD 01 00 01 023C 820 .BYTE 1,0,1,13 ; CARRIAGE CONTROL TO FORTRAN MATCH TABLE
20 0240 821 .ASCII / / ; SPACE => 1 NL, 1 CR
OD 01 00 02 0241 822 .BYTE 2,0,1,13 ; "0" => 2 NL, 1 CR
30 0245 823 .ASCII /0/
OD 01 0C 01 0246 824 .BYTE 1,12,1,13 ; "1" => 1 FF, 1 CR
31 024A 825 .ASCII /1/
OD 01 00 00 024B 826 .BYTE 0,0,1,13 ; "+" => NOTHING, 1 CR
2B 024F 827 .ASCII /+ /
00 00 00 01 0250 828 .BYTE 1,0,0,0 ; "$" => 1 NL, NOTHING
24 0254 829 .ASCII /$/

```

```

OD 01 00 01 0255 830 .BYTE 1,0,1,13 ; DEFAULT => 1 NL, 1 CR
      00 0259 831 .BYTE 0 ; TABLE END
      025A 832 ;
      025A 833 ;
      025A 834 ;
      025A 835 EXE$CARRIAGE:: ; INTERPRET CARRIAGE CONTROL
51 3C A3 9A 025A 836 MOVZBL IRP$B_CARCON(R3),R1 ; GET FORTRAN SPECIFIER
      12 13 025E 837 BEQL 20$ ; IF EQL THEN TRY PRE/SUF
50 D9 AF 9E 0260 838 MOVAB BACCTABLE,R0 ; ADDRESS MATCH TABLE
3C A3 80 D0 0264 839 10$: MOVL (R0)+,IRP$B_CARCON(R3) ; ASSUME MATCH
      60 95 0268 840 TSTB (R0) ; END OF TABLE?
      05 13 026A 841 BEQL 15$ ; IF EQL THEN YES
51 80 91 026C 842 CMPB (R0)+,R1 ; MATCH?
      F3 12 026F 843 BNEQ 10$ ; NO THEN SEARCH
      05 0271 844 15$: RSB ; ELSE RETURN
      0272 845 ;
      0272 846 ; PRE/SUF CARRIAGE CONTROL
      0272 847 ;
51 3E A3 9A 0272 848 20$: MOVZBL IRP$B_CARCON+2(R3),R1 ; GET PREFIX SPECIFIER
      02 13 0276 849 BEQL 30$ ; IF EQL THEN NONE
      19 10 0278 850 BSBB 100$ ; INTERPRET THE SPECIFIER
3C A3 51 90 027A 851 30$: MOVAB R1,IRP$B_CARCON(R3) ; INSERT NUMBER
3D A3 50 90 027E 852 MOVAB R0,IRP$B_CARCON+1(R3) ; INSERT CHARACTER
51 3F A3 9A 0282 853 MOVZBL IRP$B_CARCON+3(R3),R1 ; GET SUFFIX SPECIFIER
      02 13 0286 854 BEQL 40$ ; IF EQL THEN NONE
      09 10 0288 855 BSBB 100$ ; CONVERT THE SPECIFIER
3E A3 51 90 028A 856 40$: MOVAB R1,IRP$B_CARCON+2(R3) ; INSERT NUMBER
3F A3 50 90 028E 857 MOVAB R0,IRP$B_CARCON+3(R3) ; INSERT CHARACTER
      05 0292 858 RSB ; RETURN
      0293 859 ;
      0293 860 ; SUBROUTINE TO INTERPRET PRE/SUF SPECIFIER
      0293 861 ;
      50 D4 0293 862 100$: CLRL R0 ; ASSUME NEWLINE
08 51 07 E1 0295 863 BBC #7,R1,110$ ; IF BIT 7 CLEAR THEN DONE
51 E0 8F 8B 0299 864 BICB3 #AX0E0,R1,R0 ; REMOVE OTHER BITS
      50 029D
51 01 9A 029E 865 MOVZBL #1,R1 ; SET ONE CHARACTER
      05 02A1 866 110$: RSB ; RETURN
      02A2 867
      02A2 868 .END

```

ACB\$V_QUOTA	= 00000006			MMG\$IOLOCK	*****	X	01
ACCVIO	000001A0	R	01	P1	= 00000000		
ACCVIO1	0000019C	R	01	P2	= 00000004		
BACKOUT_QIO	000000FF	R	01	P3	= 00000008		
CCB\$W_IOC	= 0000000A			P4	= 0000000C		
CCTABLE	0000023C	R	01	P5	= 00000010		
CHECK SET	00000215	R	01	P6	= 00000014		
DC\$_DISK	= 00000001			PCB\$L_EFWM	= 0000004C		
EXE\$ABORTIO	*****	X	01	PCB\$L_STS	= 00000024		
EXE\$CARRIAGE	0000025A	RG	01	PCB\$V_SSRWAIT	= 0000000A		
EXE\$C_CMSTKSZ	*****	X	01	PCB\$W_ASTCNT	= 00000038		
EXE\$DEANONPAGED	*****	X	01	PCB\$W_BIOCNT	= 0000003A		
EXE\$FINISHIO	*****	X	01	PCB\$W_DIOCNT	= 0000003E		
EXE\$FINISHIOC	*****	X	01	PMS\$ABORT_RQ	*****	X	01
EXE\$IORSNWAIT	00000000	RG	01	PR\$ IPL	= 00000012		
EXE\$LCLDSKVALID	00000035	RG	01	SCH\$GL_RESMASK	*****	X	01
EXE\$MODIFY	00000069	RG	01	SCH\$GQ_MWAIT	*****	X	01
EXE\$MODIFYLOCK	000000A1	RG	01	SCH\$WAIT	*****	X	01
EXE\$MODIFYLOCKR	000000A4	RG	01	SF\$L_SAVE_AP	= 00000008		
EXE\$ONEPARM	00000029	RG	01	SS\$ ACCVIO	= 0000000C		
EXE\$QIDRVPKT	*****	X	01	SS\$ _ILLIOFUNC	= 000000F4		
EXE\$READ	0000006F	RG	01	SS\$ _NORMAL	= 00000001		
EXE\$READCHK	00000132	RG	01	UCB\$B_DEVCLASS	= 00000040		
EXE\$READCHKR	00000146	RG	01	UCB\$B_ONLCNT	= 000000AE		
EXE\$READLOCK	0000009B	RG	01	UCB\$L_DEVDEPEND	= 00000044		
EXE\$READLOCKR	000000AE	RG	01	UCB\$L_STS	= 00000064		
EXE\$SENSEMODE	00000232	RG	01	UCB\$V_LCL_VALID	= 00000011		
EXE\$SETCHAR	000001F5	RG	01	UCB\$V_VALID	= 0000000B		
EXE\$SETMODE	0000020C	RG	01	UCB\$W_DEVBUFSIZ	= 00000042		
EXE\$WRITE	00000078	RG	01	VA\$M_BYTE	= 000001FF		
EXE\$WRITECHK	00000138	RG	01				
EXE\$WRITECHKR	000001A4	RG	01				
EXE\$WRITELOCK	0000009E	RG	01				
EXE\$WRITELOCKR	000000B5	RG	01				
EXE\$ZEROPARM	0000002F	RG	01				
IO\$ _PACKACK	= 00000008						
IO\$ _PHYSICAL	= 0000001F						
IO\$ _READLBLK	= 00000021						
IO\$ _READPBLK	= 0000000C						
IO\$ _SETMODE	= 00000023						
IPL\$ _SCS	= 00000008						
IPL\$ _SYNCH	= 00000008						
IRP\$B_CARCON	= 0000003C						
IRP\$B_RMOD	= 0000000B						
IRP\$L_BCNT	= 00000032						
IRP\$L_DIAGBUF	= 0000004C						
IRP\$L_MEDIA	= 00000038						
IRP\$L_SVAPTE	= 0000002C						
IRP\$M_FUNC	= 00000002						
IRP\$\$_FCODE	= 00000006						
IRP\$V_BUFIO	= 00000000						
IRP\$V_DIAGBUF	= 00000007						
IRP\$V_FCIDE	= 00000000						
IRP\$V_FUNC	= 00000001						
IRP\$W_BOFF	= 00000030						
IRP\$W_FUNC	= 00000020						
IRP\$W_STS	= 0000002A						

+-----+  
! 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 .	000002A2 ( 674.)	01 ( 1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$ABSS\$	00000000 ( 0.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	15	00:00:00.07	00:00:02.88
Command processing	91	00:00:00.73	00:00:09.00
Pass 1	454	00:00:18.08	00:01:39.32
Symbol table sort	0	00:00:02.73	00:00:14.75
Pass 2	168	00:00:04.66	00:00:30.16
Symbol table output	10	00:00:00.11	00:00:00.36
Psect synopsis output	2	00:00:00.03	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	743	00:00:26.42	00:02:36.51

The working set limit was 1650 pages.  
98202 bytes (192 pages) of virtual memory were used to buffer the intermediate code.  
There were 100 pages of symbol table space allocated to hold 1840 non-local and 44 local symbols.  
969 source lines were read in Pass 1, producing 15 object records in Pass 2.  
23 pages of virtual memory were used to define 22 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	10
_\$255\$DUA28:[SYS.LIB]STARLET.MLB;2	9
TOTALS (all libraries)	19

1922 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SYSQIOFDT/OBJ=OBJ\$:SYSQIOFDT MSRC\$:SYSQIOFDT/UPDATE=(ENH\$:SYSQIOFDT)+EXECML\$/LIB



**IOSUBNPAG**

(3)	172	CANCEL I/O ON CHANNEL
(4)	204.2	Handle Last Channel Deassign
(5)	276	FILL DIAGNOSTIC BUFFER
(6)	307	RELEASE I/O CHANNEL
(7)	362	REQUEST I/O CHANNEL
(8)	422	I/O Request Completion Processing for Class Drivers
(9)	470	I/O REQUEST COMPLETION PROCESSING
(10)	560	MOUNT VERIFICATION HELPER
(11)	593	INITIATE I/O FUNCTION ON DEVICE
(12)	631	Allocate Buffered Data Path
(14)	718	Release Buffered Data Path
(15)	778	REQUEST AND ALLOCATE UNIBUS MAP REGISTERS FOR CLASS DRIVER
(16)	817.2	REQUEST UNIBUS MAP REGISTERS
(17)	862	ALLOCATE UNIBUS MAP REGISTERS
(18)	967	Allocate a specific set of UNIBUS Map Registers
(19)	1074	Permanently Allocate UNIBUS Map Registers
(21)	1197	Release UNIBUS Map Registers
(23)	1397	RETURN TO CALLER
(24)	1416	WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
(25)	1450	WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
(26)	1486	ALLOCATE SYSTEM PAGE TABLE
(27)	1521	CONVERT DEVICE NAME AND UNIT
(28)	1623	BROADCAST TO A TERMINAL
(29)	1734.2	SCAN THE I/O DATA BASE
(30)	1734.62	SCAN THE I/O DATA BASE BOTH PRIMARY & SECONDARY PATHS
(31)	1734.133	IOC\$CTRLINIT - Call driver controller init. routine
(32)	1734.198	IOC\$UNITINIT - Call driver unit init. routine
(33)	1734.277	Parse Device Name String
(34)	1734.458	Search I/O Database for Device
(35)	1734.622	Continue I/O Database Search
(36)	1734.672	Check UCB Against Search Rules



```
0000 1 .TITLE IOSUBNPAG - NONPAGED I/O RELATED SUBROUTINES
;WMC0002 0000 .1 .IDENT 'V03-033'
;WMC0002 0000 .2
-1 0000 3 ;*****
0000 4 ;*
0000 5 ;* COPYRIGHT (c) 1978, 1980, 1982 BY *
0000 6 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 7 ;* ALL RIGHTS RESERVED. *
0000 8 ;*
0000 9 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 10 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 11 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 12 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 13 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 14 ;* TRANSFERRED. *
0000 15 ;*
0000 16 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 17 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 18 ;* CORPORATION. *
0000 19 ;*
0000 20 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 21 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 22 ;*
0000 23 ;*
0000 24 ;*****
0000 25 ;
0000 26 ; D. N. CUTLER 13-JUN-76
0000 27 ;
0000 28 ;
0000 29 ; NONPAGED I/O RELATED SUBROUTINES
0000 30 ;
;ACG0399 0000 .1 ; MODIFIED BY:
;ACG0399 0000 .2 ;
;WMC0002 0000 .3 ; V03-033 WMC0002 Wayne Cardoza 03-May-1984
;WMC0002 0000 .4 ; Add support for MNTVERPND bit.
;WMC0002 0000 .5 ;
;RAS0300 0000 .6 ; V03-032 RAS0300 Ron Schaefer 2-May-1984
;RAS0300 0000 .7 ; Change IOC$CVT_DEVNAM to only prefix cluster node names if
;RAS0300 0000 .8 ; the DEV$V_NNM device characteristic is set in UCB$_DEVCHAR2.
;RAS0300 0000 .9 ; Add additional itemcode (4) to IOC$CVT_DEVNAM to provide
;RAS0300 0000 .10 ; the device name string sans unit number.
;RAS0300 0000 .11 ;
;TMK0001 0000 .12 ; V03-031 TMK0001 Todd M. Katz 23-Apr-1984
;TMK0001 0000 .13 ; Remove the $LOGDEF data definitions.
;TMK0001 0000 .14 ;
;RLRPDTADP 0000 .15 ; V03-030 RLRPDTADP Robert L. Rappaport 9-Apr-1984
;RLRPDTADP 0000 .16 ; Modify entrypoints used for allocating and deallocating
;RLRPDTADP 0000 .17 ; Buffered Data Paths and UNIBUS Map Registers for UQPORTS (UDA),
;RLRPDTADP 0000 .18 ; to pickup pointer for ADP from PDT$_ADP(R4).
;RLRPDTADP 0000 .19 ;
;ACG0414 0000 .20 ; V03-029 ACG0414 Andrew C. Goldstein, 30-Mar-1984 15:49
;ACG0414 0000 .21 ; Minor parse and searching fixes in IOC$SEARCH...
;ACG0414 0000 .22 ; add IOC$V_ALLOC to force allocation
;ACG0414 0000 .23 ;
;ACG0406 0000 .24 ; V03-028 ACG0406 Andrew C. Goldstein, 16-Mar-1984 15:42
;ACG0406 0000 .25 ; Fix bugs in searching for allocation class
;ACG0406 0000 .26 ;
```

;ACG0399	0000	.27 ;	V03-027	ACG0399	Andrew C. Goldstein,	24-Feb-1984	22:28
;ACG0399	0000	.28 ;			Add IOC\$LAST_CHAN subroutine, and move in internal I/O		
;ACG0399	0000	.29 ;			database parse and search routines, so they can be called		
;ACG0399	0000	.30 ;			by IPC.		
;ACG0399	0000	.31 ;					
;RLRMAPSP	0000	.32 ;	V03-026	RLRMAPSP	Robert L. Rappaport	15-Feb-1984	
;RLRMAPSP	0000	.33 ;			Correct bug in BEQL destination in IOC\$ALOUBAMAPSP that is		
;RLRMAPSP	0000	.34 ;			only triggered if the range specified, coincides with the		
;RLRMAPSP	0000	.35 ;			exact end of an extent of map registers.		
;RLRMAPSP	0000	.36 ;					
;ROW0292	0000	.37 ;	V03-025	ROW0292	Ralph O. Weber	4-FEB-1984	
;ROW0292	0000	.38 ;			Fix branch displacements broken by movement of EXE\$MOUNTVER to		
;ROW0292	0000	.39 ;			SYSLOAxxx.		
;ROW0292	0000	.40 ;					
;KPL0001	0000	.41 ;	V03-024	KPL0001	Peter Lieberwirth	7-Nov-1983	
;KPL0001	0000	.42 ;			Add paths for new processors to CPUDISP invocation.		
;KPL0001	0000	.43 ;					
;ROW0244	0000	.44 ;	V03-023	ROW0244	Ralph O. Weber	17-OCT-1983	
;ROW0244	0000	.45 ;			Change the IOC\$CVT_DEVNAM name string formation rules to		
;ROW0244	0000	.46 ;			eliminate _\$1TTA0_ and other allocation class based names		
;ROW0244	0000	.47 ;			for devices which can never be dual pathed. See routine		
;ROW0244	0000	.48 ;			comments for details of current operation mode.		
;ROW0244	0000	.49 ;					
;ROW0239	0000	.50 ;	V03-022	ROW0239	Ralph O. Weber	11-OCT-1983	
;ROW0239	0000	.51 ;			Fix IOC\$CVT_DEVNAM to not insert node name or trailing dollar		
;ROW0239	0000	.52 ;			sign when node name is null. Also correct comments describing		
;ROW0239	0000	.53 ;			the R4 argument to IOC\$CVT_DEVNAM.		
;ROW0239	0000	.54 ;					
;ROW0234	0000	.55 ;	V03-021	ROW0234	Ralph O. Weber	5-OCT-1983	
;ROW0234	0000	.56 ;			Change IOC\$CVT_DEVNAM to produce \$allocation-class\$device		
;ROW0234	0000	.57 ;			strings completely in ASCII, when allocation class output is		
;ROW0234	0000	.58 ;			requested. In the process rip up the whole thing because that		
;ROW0234	0000	.59 ;			was the only way to get something that worked and didn't		
;ROW0234	0000	.60 ;			occupy all non-page memory		
;ROW0234	0000	.61 ;					
;TCM0005	0000	.62 ;	V03-020	TCM0005	Trudy C. Matthews	5-OCT-1983	
;TCM0005	0000	.63 ;			Add IOC\$SCAN_IODB_2P which is functionally the same as		
;TCM0005	0000	.64 ;			IOC\$SCAN_IOCB except that both primary and secondary paths to		
;TCM0005	0000	.65 ;			a device are scanned.		
;TCM0005	0000	.66 ;					
;KDM0084	0000	.67 ;	V03-019	KDM0084	Kathleen D. Morse	26-Sep-1983	
;KDM0084	0000	.68 ;			Added MicroVAX I support to CPUDISP macros.		
;KDM0084	0000	.69 ;					
;ROW0221	0000	.70 ;	V03-018	ROW0221	Ralph O. Weber	8-SEP-1983	
;ROW0221	0000	.71 ;			Change IOC\$UNITINIT to look for a unit initialization routine		
;ROW0221	0000	.72 ;			in the DDT before looking in the CRB. See the note in the		
;ROW0221	0000	.73 ;			routine's header for details.		
;ROW0221	0000	.74 ;					
;ROW0203	0000	.75 ;	V03-017	ROW0203	Ralph O. Weber	5-AUG-1983	
;ROW0203	0000	.76 ;			Add two new routines IOC\$CTRLINIT and IOC\$UNITINIT. These are		
;ROW0203	0000	.77 ;			the proscribed mechanism for calling device driver controller.		
;ROW0203	0000	.78 ;			and unit initialization routines. These routines correctly		
;ROW0203	0000	.79 ;			setup for, locate, and call the appropriate driver routines.		
;ROW0203	0000	.80 ;					
;TCM0004	0000	.81 ;	V03-016	TCM0004	Trudy C. Matthews	26-Jul-1983	
;TCM0004	0000	.82 ;			Change IOC\$CVT_DEVNAM to return the <allocation_class>+		
;TCM0004	0000	.83 ;			<devnam> form of device name if R4 > 0.		

;TCM0004	0000	.84 ;			
;RLRBYTEOFF	0000	.85 ;	V03-015	RLRBYTEOFF	Robert L. Rappaport 27-Jun-1983
;RLRBYTEOFF	0000	.86 ;			Correct error in IOC\$REQDATAPUDA. Error is that this
;RLRBYTEOFF	0000	.87 ;			routine has operated in a NOWAIT mode, that is, if no
;RLRBYTEOFF	0000	.88 ;			Buffered Datapath was available, we just used the
;RLRBYTEOFF	0000	.89 ;			Direct Datapath. Unfortunately, this doesn't work on
;RLRBYTEOFF	0000	.90 ;			780's and 790's if the user buffer is located at an
;RLRBYTEOFF	0000	.91 ;			odd byte address since Byte Offset doesn't work on the
;RLRBYTEOFF	0000	.92 ;			Direct Datapath for the UNIBUS Adapters on these
;RLRBYTEOFF	0000	.93 ;			processors.
;RLRBYTEOFF	0000	.94 ;			
;LMPBUILD	0000	.95 ;	V03-014	LMPBUILD	L. Mark Pilant, 26-Jun-1983 23:11
;LMPBUILD	0000	.96 ;			Change references from TTY\$K_WB_HDRLEN to TTY\$K_WB_LENGTH.
;LMPBUILD	0000	.97 ;			
;TCM0003	0000	.98 ;	V03-013	TCM0003	Trudy C. Matthews 17-Jun-1983
;TCM0003	0000	.99 ;			Change the way cluster-style device names are conditionally
;TCM0003	0000	.100 ;			returned, such that cluster-style names are returned for
;TCM0003	0000	.101 ;			local disk devices if the system is participating in a
;TCM0003	0000	.102 ;			cluster (routine IOC\$CVT_DEVNAM).
;TCM0003	0000	.103 ;			
;TCM0002	0000	.104 ;	V03-012	TCM0002	Trudy C. Matthews 09-Jun-1983
;TCM0002	0000	.105 ;			Fix bug in TCM0001.
;TCM0002	0000	.106 ;			
;TCM0001	0000	.107 ;	V03-011	TCM0001	Trudy C. Matthews 21-Apr-1983
;TCM0001	0000	.108 ;			Add new parameter to IOC\$CVT_DEVNAM that allows caller
;TCM0001	0000	.109 ;			to specify whether he wants the node name returned for
;TCM0001	0000	.110 ;			local devices or not.
;TCM0001	0000	.111 ;			
;ROW0188	0000	.112 ;	V03-010	ROW0188	Ralph O. Weber 30-APR-1983
;ROW0188	0000	.113 ;			Fix broken branches to PMS\$ routines.
;ROW0188	0000	.114 ;			
;KTA3022	0000	.115 ;	V03-009	KTA3022	Kerbey T. Altmann 29-Dec-1982
;KTA3022	0000	.116 ;			Enhance KTA3018. Add new routine to scan the IO
;KTA3022	0000	.117 ;			data base and return the blocks.
;KTA3022	0000	.118 ;			
;ROW0140	0000	.119 ;	V03-008	ROW0140	Ralph O. Weber 18-NOV-1982
;ROW0140	0000	.120 ;			Cause IOC\$DALOCUBAMAP to give non-fatal INCONSTATE,
;ROW0140	0000	.121 ;			"Inconsistent UBA data base" bugcheck if number of map
;ROW0140	0000	.122 ;			registers to deallocate is zero.
;ROW0140	0000	.123 ;			
;MLJ0101	0000	.124 ;	V03-007	MLJ0101	Martin L. Jack 11-Nov-1982
;MLJ0101	0000	.125 ;			Add \$\$BDEF.
;MLJ0101	0000	.126 ;			
;KTA3018	0000	.127 ;	V03-006	KTA3018	Kerbey T. Altmann 01-Nov-1982
;KTA3018	0000	.128 ;			Modify CVT_DEVNAME for new IO database.
;KTA3018	0000	.129 ;			
;ROW0130	0000	.130 ;	V03-005	ROW0130	Ralph O. Weber 5-OCT-1982
;ROW0130	0000	.131 ;			Remove IOC\$DELMBX whose functionality is replaced by new
;ROW0130	0000	.132 ;			routines in module UCBCREDEL.
;ROW0130	0000	.133 ;			
;KDM0002	0000	.134 ;	V03-004	KDM0002	Kathleen D. Morse 28-Jun-1982
;KDM0002	0000	.135 ;			Added \$DCDEF.
;KDM0002	0000	.136 ;			
;RLR0003	0000	.137 ;	V03-003	RLR0003	Robert L. Rappaport 1-June-1982
;RLR0003	0000	.138 ;			Correct errors in UNIBUS map register allocation and
;RLR0003	0000	.139 ;			deallocation that occur when the number of active
;RLR0003	0000	.140 ;			descriptors is zero. Errors were in IOC\$ALOUBAMAPSP

```
;RLR0003      0000 .141 ;      (allocation error), IOC$ALOUBAPRM (allocation error),
;RLR0003      0000 .142 ;      and IOC$DALOCUBAMAP (deallocation error). The error
;RLR0003      0000 .143 ;      in IOC$DALOCUBAMAP is corrected in a patch to V3.1.
;RLR0003      0000 .144 ;
;RLR0002      0000 .145 ;      V03-002 RLR0002      Robert L. Rappaport      22-May-1982
;RLR0002      0000 .146 ;      Remove IOC$REQMAPREGN and all comments that reference it.
;RLR0002      0000 .147 ;
;RLR0001      0000 .148 ;      V03-001 RLR0001 Robert L. Rappaport      22-May-1982
;RLR0001      0000 .149 ;      Correct error in UNIBUS map register allocation that
;RLR0001      0000 .150 ;      doubly allocated registers when the number of active
;RLR0001      0000 .151 ;      descriptors was zero.
;RLR0001      0000 .152 ;      This bug corrected in patch to V3.1.
;RLR0001      0000 .153 ;
```

```
-114          0000 145 ;  
              0000 146 ;  
              0000 147 ; MACRO LIBRARY CALLS  
              0000 148 ;  
              0000 149 ;  
              0000 150      $ADPDEF          ;DEFINE ADP OFFSETS  
              0000 151      $CADEF          ;DEFINE CONDITIONAL ASSEMBLY PARAMETERS  
;ACG0399      0000 .1      $CANDEF          ;DEFINE CANCEL I/O REASON CODES  
;ACG0399      0000 .2      $CDRPDEF        ;DEFINE CLASS DRIVER I/O REQUEST PACKET  
;ACG0399      0000 .3      $CRBDEF         ;DEFINE CRB OFFSETS  
;KDM0002      0000 .4      $DCDEF          ;DEFINE DEVICE CLASSES  
-2           0000 154      $DDBDEF         ;DEFINE DDB OFFSETS  
              0000 155      $DDTDEF         ;DEFINE DDT OFFSETS  
              0000 156      $DEVDEF        ;DEFINE DEVICE CHARACTERISTICS FLAGS  
              0000 157      $DYNDEF        ;DEFINE DYNAMIC POOL BLOCK TYPES  
              0000 158      $EMBDEF        ;DEFINE EMB OFFSETS  
              0000 159      $IDBDEF        ;DEFINE IDB OFFSETS  
;TMK0001      0000 .1      $IOCDEF         ;DEFINE IOC$SEARCHxxx FLAGS  
;TMK0001      0000 .2      $IPLDEF        ;DEFINE INTERRUPT PRIORITY LEVELS  
;TMK0001      0000 .3      $IRPDEF        ;DEFINE IRP OFFSETS  
;TMK0001      0000 .4      $JIBDEF        ;DEFINE JIB OFFSETS  
;TMK0001      0000 .5      $LCKDEF        ;DEFINE LOCK MANAGER SYMBOLS  
;TMK0001      0000 .6      $MSCPDEF       ;DEFINE MSCP STRUCTURES  
;TMK0001      0000 .7      $PCBDEF        ;DEFINE PCB OFFSETS  
;TMK0001      0000 .8      $PDTDEF        ;Define PDT offsets  
;TMK0001      0000 .9      $PRDEF         ;DEFINE PROCESSOR REGISTERS  
;TMK0001      0000 .10     $PRVDEF        ;DEFINE PRIVILEGE BITS  
;MLJ0101      0000 .11     $SBDEF         ; Define system block offsets  
;ACG0399      0000 .12     $SSDEF         ;DEFINE SYSTEM STATUS CODES  
;ACG0399      0000 .13     $TTYDEF        ;DEFINE TERMINAL WRITE PACKET OFFSETS  
;ACG0399      0000 .14     $UBMDEF        ;Define UNIBUS Map Descriptor structure  
;ACG0399      0000 .15     $UCBDEF        ;DEFINE UCB OFFSETS  
;ACG0399      0000 .16     $VECDEF        ;DEFINE CRB VECTOR OFFSETS
```

```

-12      0000 172      .SBTTL  CANCEL I/O ON CHANNEL
        0000 173 ;+
        0000 174 ; IOC$CANCELIO - CANCEL I/O ON CHANNEL
        0000 175 ;
        0000 176 ; THIS ROUTINE IS A DEVICE INDEPENDENT CANCEL I/O ROUTINE THAT CONDITIONALLY
        0000 177 ; MARKS THE UCB SUCH THAT THE CURRENT I/O REQUEST WILL BE CANCELED IF CONDITIONS
        0000 178 ; WARRANT SUCH A ACTION.
        0000 179 ;
        0000 180 ; INPUTS:
        0000 181 ;
        0000 182 ;           R2 = NEGATIVE OF THE CHANNEL NUMBER.
        0000 183 ;           R3 = CURRENT IO PACKET.
        0000 184 ;           R4 = PCB ADDRESS.
        0000 185 ;           R5 = UCB ADDRESS.
        0000 186 ;
        0000 187 ; OUTPUTS:
        0000 188 ;
        0000 189 ;           IF THE DEVICE IS BUSY, THE REQUEST IS FOR THE CURRENT PROCESS, AND
        0000 190 ;           THE I/O WAS ISSUED FROM THE DESIGNATED CHANNEL, THEN THE CANCEL I/O
        0000 191 ;           BIT IS SET IN THE CORRESPONDING UCB.
        0000 192 ;
        0000 193 ;           R2, R3, R4, AND R5 ARE PRESERVED ACROSS CALL.
        0000 194 ;-
        0000 195
00000000 196      .PSECT  WIONONPAGED
        0000 197 IOC$CANCELIO:: ;CANCEL I/O ON CHANNEL
11 64 A5 08 E1 0000 198      BBC      #UCB$V_BSY,UCB$W_STS(R5),10$ ;IF CLR, DEVICE NOT BUSY
60 A4 0C A3 D1 0005 199      Cmpl    IRP$L_PID(R3),PCB$L_PID(R4) ;PROCESS ID MATCH?
        0000 200      BNEQ    10$      ;IF NEQ NO
28 A3 52 B1 000C 201      CMPW    R2,IRP$W_CHAN(R3) ;CHANNEL NUMBER MATCH
        0000 202      BNEQ    10$      ;IF NEQ NO
64 A5 08 A8 0012 203      BISW    #UCB$M_CANCEL,UCB$W_STS(R5) ;SET CANCEL PENDING
        0000 204 10$:      RSB

```

```

;ACG0399          0017 .2      .SBTTL Handle Last Channel Deassign
;ACG0399          0017 .3
;ACG0399          0017 .4 ;+
;ACG0399          0017 .5 ; IOC$LAST_CHAN - Last Channel Deassign Specific
;ACG0399          0017 .6 ; IOC$LAST_CHAN_AMBX - Last Assoc. MBX Channel Deassign Specific
;ACG0399          0017 .7 ;
;ACG0399          0017 .8 ; Functional Description:
;ACG0399          0017 .9 ;
;ACG0399          0017 .10 ;      Common functions done on last channel deassignment are handled. The
;ACG0399          0017 .11 ;      driver's cancel I/O routine is called with an appropriate reason code
;ACG0399          0017 .12 ;      (CAN$C_DASSGN for regular deassign, or CAN$C_AMBXDGN for associated
;ACG0399          0017 .13 ;      mailboxes). If after the cancel routine finished UCB$V_DELETEUCB is
;ACG0399          0017 .14 ;      set, the UCB is credited and deleted.
;ACG0399          0017 .15 ;
;ACG0399          0017 .16 ; Inputs:
;ACG0399          0017 .17 ;
;ACG0399          0017 .18 ;      R5      UCB address
;ACG0399          0017 .19 ;      R2      Channel index (LAST_CHAN only)
;ACG0399          0017 .20 ;
;ACG0399          0017 .21 ; Outputs:
;ACG0399          0017 .22 ;
;ACG0399          0017 .23 ;      R0 thru R3 destroyed.
;ACG0399          0017 .24 ;      If appropriate, UCB is deallocated.
;ACG0399          0017 .25 ;
;ACG0399          0017 .26 ; -
;ACG0399          0017 .27
;ACG0399          0017 .28      .ENABLE LSB
;ACG0399          0017 .29
;ACG0399          0017 .30 IOC$LAST_CHAN_AMBX::
;ACG0399          52 7C 0017 .31      CLRQ   R2                ; Clear unused cancel inputs.
;ACG0399          58 02 9A 0019 .32      MOVZBL #CAN$C_AMBXDGN, R8      ; Set cancel reason code.
;ACG0399          07 11 001C .33      BRB     10$
;ACG0399          001E .34
;ACG0399          001E .35 IOC$LAST_CHAN::
;ACG0399          53 58 A5 D0 001E .36      MOVL   UCB$L_IRP(R5), R3      ; Get active packet address.
;ACG0399          58 01 9A 0022 .37      MOVZBL #CAN$C_DASSGN, R8      ; Set cancel reason code.
;ACG0399          0025 .38
;ACG0399          50 0088 C5 D0 0025 .39 10$:   MOVL   UCB$L_DDT(R5), R0      ; Get DDT address.
;ACG0399          002A .40      SETIPL UCB$B_FIPL(R5)          ; Raise to fork IPL.
;ACG0399          0C B0 16 002E .41      JSB    @DDT$L_CANCEL(R0)      ; Call driver's cancel I/O routine.
;ACG0399          0031 .42      SETIPL #IPL$_ASTDEL          ; Lower IPL.
;ACG0399          1A 38 A5 17 E0 0034 .43      BBS    #DEV$V_ALL, -          ; Branch if still allocated
;ACG0399          0039 .44      UCB$L_DEVCHAR(R5), 30$
;ACG0399          00100004 8F D3 0039 .45      BITL   #DEV$M_TRM:DEV$M_MBX, - ; Is this a terminal, remote terminal
;ACG0399          38 A5 003F .46
;ACG0399          0041 .46      UCB$L_DEVCHAR(R5)          ; or mailbox?
;ACG0399          00 38 A5 05 13 0041 .47      BEQL   20$                  ; Branch if not.
;ACG0399          00 38 A5 07 E4 0043 .48      BBSC   #DEV$V_OPR, -          ; Else, clear OPR bit.
;ACG0399          0048 .49      UCB$L_DEVCHAR(R5), 20$      ; This is an implicit operator disable.
;ACG0399          10 E1 0048 .50 20$:   BBC    #UCB$V_DELETEUCB, -      ; Branch if UCB not to be deleted.
;ACG0399          06 64 A5 004A .51      UCB$L_STS(R5), 30$
;ACG0399          FF00 30 004D .52      BSBW   IOC$CREDIT_UCB        ; Else credit UCB quotas,
;ACG0399          FFAD 30 0050 .53      BSBW   IOC$DELETE_UCB        ; and delete the UCB.
;ACG0399          05 0053 .54 30$:   RSB
;ACG0399          0054 .55
;ACG0399          0054 .56      .DISABLE LSB

```

-71

```

0054 276          .SBTTL  FILL DIAGNOSTIC BUFFER
0054 277 ;+
0054 278 ; IOC$DIAGBUFILL - FILL DIAGNOSTIC BUFFER
0054 279 ;
0054 280 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION, BUT BEFORE RELEASING
0054 281 ; THE I/O CHANNEL, TO FILL THE FINAL DEVICE PARAMETERS INTO AN INTERNAL DIAG-
0054 282 ; NOSTIC BUFFER IF ONE IS SPECIFIED.
0054 283 ;
0054 284 ; INPUTS:
0054 285 ;
0054 286 ;          R4 = ADDRESS OF DEVICE CSR REGISTER.
0054 287 ;          R5 = DEVICE UNIT UCB ADDRESS.
0054 288 ;
0054 289 ; OUTPUTS:
0054 290 ;
0054 291 ;          IF A DIAGNOSTIC BUFFER WAS SPECIFIED IN THE ORIGINAL REQUEST, THEN
0054 292 ;          THE COMPLETION TIME, FINAL ERROR COUNTERS, AND DEVICE REGISTERS ARE
0054 293 ;          FILLED INTO THE DIAGNOSTIC BUFFER.
0054 294 ; -
0054 295
0054 296 IOC$DIAGBUFILL::          ; FILL DIAGNOSTIC BUFFER
      53 58 A5 D0 0054 297        MOVL   UCB$L_IRP(R5),R3          ; GET ADDRESS OF I/O PACKET
      1B 2A A3 07 E1 0058 298        BBC   #IRP$V_DIAGBUF,IRP$W_STS(R3),10$ ; IF CLR, NO DIAGNOSTIC BUFFER
      50 4C B3 D0 005D 299        MOVL   @IRP$L_DIAGBUF(R3),R0      ; GET ADDRESS OF INTERNAL BUFFER DATA AREA
      50 08 C0 0061 300        ADDL   #8,R0          ; POINT PAST START TIME
      00000000'EF 7D 0064 301        MOVQ   EXE$GQ_SYSTIME,(R0)+    ; INSERT COMPLETION TIME
      80 80 C5 006A 302
      80 0080 C5 3C 006B 302        MOVZWL UCB$B_ERTCNT(R5),(R0)+  ; INSERT FINAL ERROR COUNTERS
      52 0088 C5 D0 0070 303        MOVL   UCB$L_DDT(R5),R2      ; GET ADDRESS OF DDT
      10 B2 16 0075 304        JSB    @DDT$L_REGDUMP(R2)        ; CALL DEVICE SPECIFIC REGISTER DUMP ROUTINE
      05 0078 305 10$: RSB          ;

```



-1

```

0079 307          .SBTTL  RELEASE I/O CHANNEL
0079 308 ;+
0079 309 ; IOC$RELCHAN - RELEASE ALL I/O CHANNELS
0079 310 ; IOC$RELSCHAN - RELEASE SECONDARY I/O CHANNEL
0079 311 ;
0079 312 ; THIS ROUTINE IS CALLED AT THE END OF AN I/O OPERATION TO RELEASE ALL
0079 313 ; CHANNELS THE I/O WAS BEING PERFORMED ON.
0079 314 ;
0079 315 ; INPUTS:
0079 316 ;
0079 317 ;          R5 = UCB ADDRESS OF DEVICE UNIT.
0079 318 ;
0079 319 ; OUTPUTS:
0079 320 ;
0079 321 ;          THE CHANNELS ARE RELEASED AND AN ATTEMPT IS MADE TO REMOVE THE NEXT
0079 322 ;          WAITING DRIVER PROCESS FROM EACH CHANNEL QUEUE. IF A DRIVER PROCESS
0079 323 ;          IS WAITING, THEN THE CHANNEL IS ASSIGNED TO THAT DRIVER PROCESS AND
0079 324 ;          IT IS CALLED VIA A JSB TO ITS CHANNEL WAIT RETURN ADDRESS. WHEN THE
0079 325 ;          CALLED DRIVER PROCESS RETURNS, A RETURN IS MADE TO THE DRIVER PROCESS
0079 326 ;          THAT RELEASED THE CHANNEL. IF THERE IS NO DRIVER PROCESS WAITING FOR
0079 327 ;          THE CHANNEL, THEN THE CHANNEL STATUS IS SET TO IDLE.
0079 328 ;
0079 329 ;          R3 AND R4 ARE PRESERVED ACROSS CALL.
0079 330 ;-
0079 331
0079 332          .ENABL  LSB
0079 333 IOC$RELSCHAN::          ;RELEASE SECONDARY I/O CHANNEL
50 24 A5 D0 0079 334          MOVL  UCB$L_CRB(R5),R0          ;GET ADDRESS OF PRIMARY CRB
50 20 A0 D0 007D 335          MOVL  CRB$L_LINK(R0),R0          ;GET ADDRESS OF SECONDARY CRB
                    10 11 0081 336          BRB    20$          ;
                    0083 337 IOC$RELCHAN::          ;RELEASE I/O CHANNEL
50 24 A5 D0 0083 338          MOVL  UCB$L_CRB(R5),R0          ;GET ADDRESS OF PRIMARY CRB
50 20 A0 D0 0087 339          MOVL  CRB$L_LINK(R0),R0          ;GET ADDRESS OF SECONDARY CRB
                    02 13 008B 340          BEQL  10$          ;IF EQL NONE
                    04 10 008D 341          BSBB  20$          ;RELEASE SECONDARY CHANNEL
50 24 A5 D0 008F 342 10$:      MOVL  UCB$L_CRB(R5),R0          ;GET ADDRESS OF PRIMARY CRB
25 0E A0 00 E1 0093 343 20$:      BBC    *CRB$V_BSY,CRB$B_MASK(R0),30$ ;IF CLR, THEN CHANNEL NOT BUSY
51 2C A0 D0 0098 344          MOVL  CRB$L_INTD+VEC$L_IDB(R0),R1 ;GET ADDRESS OF IDB
04 A1 55 D1 009C 345          CMPL  R5,IDB$L_OWNER(R1)          ;DRIVER PROCESS OWN CHANNEL?
                    1B 12 00A0 346          BNEQ  30$          ;IF NEQ NO
52 00 B0 0F 00A2 347          REMQUE @CRB$L_WQFL(R0),R2          ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
                    16 1D 00A6 348          BVS   40$          ;IF VS NO DRIVER PROCESS WAITING
                    38 BB 00A8 349          PUSHR #^M<R3,R4,R5>          ;SAVE CONTEXT OF CURRENT DRIVER PROCESS
                    55 52 D0 00AA 350          MOVL  R2,R5          ;COPY ADDRESS OF DRIVER PROCESS FORK BLOCK
53 10 A5 D0 00AD 351          MOVL  UCB$L_FR3(R5),R3          ;LOAD WAITING DRIVER PROCESS CONTEXT
                    54 61 D0 00B1 352          MOVL  IDB$L_CSR(R1),R4          ;SET ASSIGNED CHANNEL CSR ADDRESS
04 A1 55 D0 00B4 353          MOVL  R5,IDB$L_OWNER(R1)          ;SET ADDRESS OF OWNER PROCESS UCB
                    0C B5 16 00B8 354          JSB   @UCB$L_FPC(R5)          ;CALL DRIVER AT CHANNEL WAIT RETURN ADDRESS
                    38 BA 00BB 355          POPR  #^M<R3,R4,R5>          ;RESTORE PREVIOUS DRIVER PROCESS CONTEXT
                    05 00BD 356 30$:      RSB          ;
                    04 A1 D4 00BE 357 40$:      CLRL  IDB$L_OWNER(R1)          ;CLEAR OWNER UNIT UCB ADDRESS
0E A0 01 8A 00C1 358          BICB  #CRB$M_BSY,CRB$B_MASK(R0) ;CLEAR CHANNEL BUSY
                    05 00C5 359          RSB          ;
                    00C6 360          .DSABL  LSB

```

-1

```

00C6 362          .SBTTL  REQUEST I/O CHANNEL
00C6 363 ;+
00C6 364 ; IOC$REQPCHANH - REQUEST PRIMARY I/O CHANNEL HIGH PRIORITY
00C6 365 ; IOC$REQSCHANH - REQUEST SECONDARY I/O CHANNEL HIGH PRIORITY
00C6 366 ; IOC$REQPCHANL - REQUEST PRIMARY I/O CHANNEL LOW PRIORITY
00C6 367 ; IOC$REQSCHANL - REQUEST SECONDARY I/O CHANNEL LOW PRIORITY
00C6 368 ;
00C6 369 ; THESE ROUTINES ARE CALLED TO REQUEST AN I/O CHANNEL TO PERFORM AN I/O
00C6 370 ; OPERATION ON.
00C6 371 ;
00C6 372 ; INPUTS:
00C6 373 ;
00C6 374 ;         R5 = UCB ADDRESS OF DEVICE UNIT.
00C6 375 ;         04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
00C6 376 ;
00C6 377 ; OUTPUTS:
00C6 378 ;
00C6 379 ;         IF THE SPECIFIED I/O CHANNEL IS IDLE, THEN IT IS IMMEDIATELY
00C6 380 ;         ASSIGNED TO THE CURRENT DRIVER PROCESS. ELSE THE DRIVER PROCESS
00C6 381 ;         CONTEXT IS SAVED IN ITS FORK BLOCK, THE FORK BLOCK IS INSERTED
00C6 382 ;         IN THE CHANNEL WAIT QUEUE, AND A RETURN TO THE DRIVER PROCESS'
00C6 383 ;         CALLER IS EXECUTED.
00C6 384 ;
00C6 385 ;         WHEN THE CHANNEL IS ASSIGNED, THE CSR ADDRESS OF THE ASSIGNED
00C6 386 ;         CONTROLLER IS RETURNED TO THE CALLER IN REGISTER R4.
00C6 387 ;
00C6 388 ;         R3 IS PRESERVED ACROSS CALL.
00C6 389 ; -
00C6 390
00C6 391          .ENABL  LSB
00C6 392 IOC$REQSCHANH:: ;REQUEST SECONDARY I/O CHANNEL HIGH PRIORITY
50 24 A5 D0 00C6 393        MOVL  UCB$_CRB(R5),R0 ;GET ADDRESS OF PRIMARY CRB
50 20 A0 D0 00CA 394        MOVL  CRB$_LINK(R0),R0 ;GET ADDRESS OF SECONDARY CRB
           OE 11 00CE 395        BRB   10$ ;
           00D0 396 IOC$REQSCHANL:: ;REQUEST SECONDARY I/O CHANNEL LOW PRIORITY
50 24 A5 D0 00D0 397        MOVL  UCB$_CRB(R5),R0 ;GET ADDRESS OF PRIMARY CRB
50 20 A0 D0 00D4 398        MOVL  CRB$_LINK(R0),R0 ;GET ADDRESS OF SECONDARY CRB
           OD 11 00D8 399        BRB   20$ ;
           00DA 400 IOC$REQPCHANH:: ;REQUEST PRIMARY I/O CHANNEL HIGH PRIORITY
50 24 A5 D0 00DA 401        MOVL  UCB$_CRB(R5),R0 ;GET ADDRESS OF PRIMARY CRB
           52 50 D0 00DE 402 10$: MOVL  R0,R2 ;SET ADDRESS OF WAIT QUEUE LISTHEAD
           08 11 00E1 403        BRB   30$ ;
           00E3 404 IOC$REQPCHANL:: ;REQUEST PRIMARY I/O CHANNEL LOW PRIORITY
50 24 A5 D0 00E3 405        MOVL  UCB$_CRB(R5),R0 ;GET ADDRESS OF PRIMARY CRB
52 04 A0 D0 00E7 406 20$: MOVL  CRB$_WQBL(R0),R2 ;GET ADDRESS OF LAST ENTRY IN QUEUE
51 2C A0 D0 00EB 407 30$: MOVL  CRB$_INTD+VEC$_IDB(R0),R1 ;GET ADDRESS OF IDB
08 0E A0 00 E2 00EF 408        BBSS  #CRB$_BSY,CRB$_MASK(R0),40$ ;IF SET, THEN CHANNEL BUSY
           54 61 D0 00F4 409        MOVL  IDB$_CSR(R1),R4 ;SET ASSIGNED CHANNEL CSR ADDRESS
04 A1 55 D0 00F7 410        MOVL  R5,IDB$_OWNER(R1) ;SET OWNER UCB ADDRESS
           05 00FB 411        RSB ;
10 A5 53 D0 00FC 412 40$: MOVL  R3,UCB$_FR3(R5) ;SAVE R3 IN FORK BLOCK
           OC A5 8ED0 0100 413        POPL  UCB$_FPC(R5) ;SAVE CHANNEL WAIT RETURN ADDRESS
           62 65 OE 0104 414        INSQUE UCB$_FQFL(R5),CRB$_WQFL(R2) ;INSERT DRIVER PROCESS IN CHANNEL WAIT
04 A1 55 D1 0107 415        CMPL  R5,IDB$_OWNER(R1) ;CURRENT DRIVER PROCESS OWNER?
           03 12 010B 416        BNEQ  50$ ;IF NEQ, BRANCH TO RETURN
           FF73 31 010D 417        BRW   IOC$RELCHAN ;IF EQL BRW TO RELEASE CHANNELS
           0110 418 50$:

```

IOSUBNPAG  
V03-033

- NONPAGED I/O RELATED SUBROUTINES  
REQUEST I/O CHANNEL

3-JUN-1984 11:33:30 VAX-11 Macro V03-01 Page 11  
12-MAR-1982 17:12:23 DISK\$VMSMASTER:[SYS.SRC]IOSUBNPAG.(7)

05 0110 419 RSB  
0111 420 .DSABL LSB ;

```

-1          0111 422          .SBTTL I/O Request Completion Processing for Class Drivers
          0111 423
          0111 424 ;+
          0111 425 ; IOC$ALTREQCOM - I/O Request Complete Alternate Entry.
          0111 426 ;
          0111 427 ; This routine is entered when an I/O operation is completed on one
          0111 428 ;         one of the devices using the disk or tape class drivers.
          0111 429 ;         The packet is inserted in the I/O finish queue for I/O post
          0111 430 ;         processing.
          0111 431 ;
          0111 432 ; INPUTS:
          0111 433 ;
          0111 434 ;         R0 = First longword of I/O status
          0111 435 ;         R1 = Second longword of I/O status
          0111 436 ;         R5 = CDRP address
          0111 437 ;
          0111 438 ; OUTPUTS:
          0111 439 ;
          0111 440 ;         The I/O packet is inserted in the I/O Post Processing Queue,
          0111 441 ;         a Software interrupt is requested to initiate I/O Post
          0111 442 ;         Processing.
          0111 443 ; -
          0111 444
          0111 445 IOC$ALTREQCOM::
53   A0 A5 9E 0111 446          MOVAB   CDRP$L_IOQFL(R5),R3          ; R3 => IRP section of CDRP. This is
          0115 447          ; for compatibility with rest of QIO
          0115 448          ; logic.
55   1C A3 D0 0115 449          MOVL   IRP$L_UCB(R3),R5          ; R5 => UCB.
       70 A5 D6 0119 450          INCL   UCB$L_OPCNT(R5)          ; Increment operations completed
          011C 451
          15 50 E9 011C 452          BLBC   R0,20$          ; LBC implies I/O error, so goto call
          011F 453          ; MOUNT VERIFICATION just in case.
          011F 454 10$:
38   A3 50 7D 011F 455          MOVQ   R0,IRP$L_MEDIA(R3)          ; Save final I/O status in IRP.
          0123 456
          0123 457          .IF DF CA$_MEASURE_IOT
          0123 458
;ROW0188 00000000'GF 16 0123 459          JSB   G^PMS$END_IO          ; Insert end of I/O transaction message
-1          0129 460          .ENDC
          0129 461
          0129 462
          63 0E 0129 463          INSQUE (R3),@L^IOC$GL_PSBL          ; Insert packet in POST process queue
00000000'FF          012B
          0130 464          SOFTINT #IPL$_IOPOST          ; Initiate SOFTWARE INTERRUPT
          05 0133 465          RSB
          0134 466 20$:
;ROW0292 00000000'GF 16 0134 467          JSB   G^EXE$MOUNTVER          ; If LBC, call MOUNT VERIFICATION.
;ACG0399          E3 11 013A 468          BRB   10$          ; Go back to normal flow.

```

-3

```

013C 470 .SBTTL I/O REQUEST COMPLETION PROCESSING
013C 471 ;+
013C 472 ; IOC$REQCOM - I/O REQUEST COMPLETE
013C 473 ;
013C 474 ; THIS ROUTINE IS ENTERED WHEN AN I/O OPERATION IS COMPLETED ON A
013C 475 ; DEVICE UNIT. THE FINAL I/O STATUS IS STORED IN THE ASSOCIATED I/O
013C 476 ; PACKET AND THE PACKET IS INSERTED IN THE I/O FINISH QUEUE FOR
013C 477 ; I/O POST PROCESSING. DEVICE UNIT BUSY IS CLEARED AND AN ATTEMPT
013C 478 ; IS MADE TO START ANOTHER I/O REQUEST ON THE DEVICE UNIT.
013C 479 ;
013C 480 ; IF THE I/O REQUEST COMPLETED WITH AN ERROR, AND THE DEVICE IS
013C 481 ; A DISK, THEN BRANCH TO THE MOUNT VERIFICATION CODE, WHICH WILL
013C 482 ; DETERMINE IF THE SITUATION REQUIRES MOUNT VERIFICATION.
013C 483 ;
013C 484 ; IF MOUNT VERIFICATION IS IN PROGRESS, NO FURTHER I/O REQUESTS WILL
013C 485 ; BE INITIATED. THIS HAS A SIDE EFFECT OF KEEPING THE 'BSY' BIT IN
013C 486 ; WHATEVER STATE IT IS CURRENTLY IN. FOR CONVENTIONAL DISK DRIVERS,
013C 487 ; THE BSY BIT WILL BE LEFT ON, WHICH WILL BLOCK $QIO FROM INITIATING
013C 488 ; ANY NEW I/O ON THE DEVICE. FOR THE DISK CLASS DRIVER, THE BUSY
013C 489 ; BIT WILL BE OFF, WHICH WILL ALLOW $QIO TO INITIATE NEW I/O.
013C 490 ;
013C 491 ; INPUTS:
013C 492 ;
013C 493 ; R0 = FIRST LONGWORD OF I/O STATUS.
013C 494 ; R1 = SECOND LONGWORD OF I/O STATUS.
013C 495 ; R5 = UCB ADDRESS OF DEVICE UNIT.
013C 496 ;
013C 497 ; OUTPUTS:
013C 498 ;
013C 499 ; THE I/O PACKET IS INSERTED IN THE I/O POST PROCESSING QUEUE
013C 500 ; AND DEVICE UNIT BUSY IS CLEARED. A SOFTWARE INTERRUPT IS
013C 501 ; REQUESTED TO INITIATE I/O POST PROCESSING.
013C 502 ;-
013C 503
013C 504 .ENABL LSB
013C 505 IOC$REQCOM: ; I/O DONE PROCESSING
1C 64 A5 02 E5 013C 506 BBCC #UCB$V_ERLOGIP,UCB$W_STS(R5),10$ ;IF CLR, ERROR LOG NOT IN PROGRESS
52 0094 C5 D0 0141 507 MOVL UCB$L_EMB(R5),R2 ;GET ADDRESS OF ERROR MESSAGE BUFFER
1A A2 64 A5 B0 0146 508 MOVW UCB$W_STS(R5),EMB$W_DV_STS(R2) ;INSERT FINAL DEVICE STATUS
0080 C5 B0 014B 509 MOVW UCB$B_ERTCNT(R5),EMB$B_DV_ERTCNT(R2) ;INSERT FINAL ERROR COUNTERS
10 A2 014F
12 A2 50 7D 0151 510 MOVQ R0,EMB$Q_DV_IOSB(R2) ;INSERT FINAL I/O STATUS
50 DD 0155 511 PUSHL R0 ;SAVE R0
FEA6' 30 0157 512 BSBW ERL$RELEASEMB ;RELEASE ERROR MESSAGE BUFFER
50 BED0 015A 513 POPL R0 ;RESTORE R0
53 58 A5 D0 015D 514 10$: MOVL UCB$L_IRP(R5),R3 ;GET ADDRESS OF I/O PACKET
70 A5 D6 0161 515 INCL UCB$L_OPCNT(R5) ;INCREMENT OPERATIONS COMPLETED
2A 50 E9 0164 516 BLBC R0,DISKCHK ;IF I/O ERROR, CHECK FOR DISK DEVICE
0167 517 ;
0167 518 ; DO NOT SAVE THE I/O STATUS IN THE IRP UNTIL IT HAS BEEN DECIDED THAT
0167 519 ; MOUNT VERIFICATION IS NOT NECESSARY. THIS IS TO AVOID OVERWRITING THE
0167 520 ; PHYSICAL DISK ADDRESS STORED IN THE IRP AT OFFSET IRP$L_MEDIA.
0167 521 ;
38 A3 50 7D 0167 522 20$: MOVQ R0,IRP$L_MEDIA(R3) ;STORE FINAL I/O STATUS
0168 523
0168 524 .IF DF CA$_MEASURE_IOT
0168 525

```

```

00000000'EF D5 016B 526 TSTL L^PMS$GL_IOPFMPDB ;DATA COLLECTION ENABLED?
          36 12 0171 527 BNEQ DO_PMS ;BRANCH IF YES
          0173 528
          0173 529 .ENDC
          0173 530
          63 0E 0173 531 PMSSEND: INSQUE (R3),@L^IOC$GL_PSBLL ;INSERT PACKET IN POST PROCESS QUEUE
00000000'FF 0175
          017A 532 SOFTINT #IPL$ IOPOST ;INITIATE SOFTWARE INTERRUPT
          0E E0 017D 533 BBS #UCB$V_MNTVERIP,- ;BRANCH IF MOUNT VERIFICATION IN PROGRESS
;WMC0002 2F 64 A5 017F .1 UCB$W_STS(R5),MNTVERPNDCHK ;(NOTE THIS LEAVES 'BSY' AS IS)
-1 53 4C B5 0F 0182 535 NXTIRP: REMQUE @UCB$L_IOQFL(R5),R3 ;REMOVE I/O PACKET FROM DEVICE UNIT QUEUE
          4C 1C 0186 536 BVC IOC$INITIATE ;IF VC INITIATE NEXT FUNCTION
          0100 8F AA 0188 537 BICW #UCB$M_BSY,UCB$W_STS(R5) ;CLEAR UNIT BUSY
          64 A5 018C
          018E 538 RELEASE: ;RELEASE ALL CHANNELS
          FEF2 31 018E 539 BRW IOC$RELCHAN ;
          0191 540 ;
          0191 541 ; IF THIS IS A DISK DEVICE, CALL THE MOUNT VERIFICATION ROUTINE
          0191 542 ; TO DETERMINE IF MOUNT VERIFICATION IS NECESSARY. IF NOT, CONTROL
          0191 543 ; WILL RETURN, AND THE REQUEST WILL BE COMPLETED IN THE NORMAL MANNER.
          0191 544 ;
          0191 545 DISKCHK:
          01 91 0191 546 CMPB #DC$_DISK,- ;IS THIS DEVICE A DISK?
          40 A5 0193 547 UCB$_DEVCLASS(R5) ;
          D0 12 0195 548 BNEQ 20$ ;BRANCH IF NOT
;WMC0002 13 E5 0197 .1 BBCC #UCB$V_MNTVERPND,- ;CHECK FOR MOUNT VERIFICATION PENDING
;WMC0002 05 64 A5 0199 .2 UCB$L_STS(R5),30$ ;IF NOT, JUST ENTER MOUNT VERIFICATION
;WMC0002 0E E5 019C .3 BBCC #UCB$V_MNTVERIP,- ;CLEAR IN-PROGRESS BIT BEFORE CALL
;WMC0002 00 64 A5 019E .4 UCB$L_STS(R5),30$ ; SO IT WILL REALLY START
;WMC0002 00000000'GF 16 01A1 .5 30$: JSB G^EXE$MOUNTVER ;START MOUNT VERIFICATION
;RLRMAPSP BE 11 01A7 .6 BRB 20$ ;COMPLETE I/O REQUEST
;RLRMAPSP 01A9 .7
;RLRMAPSP 01A9 .8 .IF DF CA$_MEASURE_IOT
;RLRMAPSP 01A9 .9
;ROW0188 00000000'GF 16 01A9 .10 DO_PMS: JSB G^PMS$END_IO ;INSERT END OF I/O TRANSACTION MESSAGE
;WMC0002 C2 11 01AF .11 BRB PMSSEND ;REJOIN COMMON CODE
;WMC0002 01B1 .12
;WMC0002 01B1 .13 .ENDC
;WMC0002 01B1 .14 ;
;WMC0002 01B1 .15 ; THE MOUNT-VERIFICATION-PENDING BIT IS USED TO INDICATE THAT A DISK SHOULD GO
;WMC0002 01B1 .16 ; INTO MOUNT VERIFICATION AS SOON AS THE CURRENT I/O IS DONE. THIS IS INTENDED
;WMC0002 01B1 .17 ; FOR USE IN A CLUSTER TO STALL I/O WHEN QUORUM IS LOST.
;WMC0002 01B1 .18 ;
;WMC0002 01B1 .19 MNTVERPNDCHK:
;WMC0002 13 E5 01B1 .20 BBCC #UCB$V_MNTVERPND,- ;CHECK FOR MOUNT VERIFICATION PENDING
;WMC0002 D8 64 A5 01B3 .21 UCB$L_STS(R5),RELEASE ;IF NOT, JUST CLEAN UP
;WMC0002 01 91 01B6 .22 CMPB #DC$_DISK,- ;IS THIS DEVICE A DISK?
;WMC0002 40 A5 01B8 .23 UCB$_DEVCLASS(R5) ;
;WMC0002 D2 12 01BA .24 BNEQ RELEASE ;BRANCH IF NOT
;WMC0002 0E E5 01BC .25 BBCC #UCB$V_MNTVERIP,- ;CLEAR IN-PROGRESS BIT BEFORE CALL
;WMC0002 00 64 A5 01BE .26 UCB$L_STS(R5),40$
;WMC0002 53 D4 01C1 .27 40$: CLRL R3 ;NO IRP PASSED TO MOUNT VERIFICATION
;WMC0002 00000000'GF 16 01C3 .28 JSB G^EXE$MOUNTVER ;TRY TO START MOUNT VERIFICATION
;WMC0002 B7 11 01C9 .29 BRB NXTIRP ;WASN'T NECESSARY
;WMC0002 01CB .30
;WMC0002 01CB .31 .DSABL LSB

```

-11

```

01CB 560          .SBTTL  MOUNT VERIFICATION HELPER
01CB 561 ;++
01CB 562 ; IOC$MNTVER  - Assist driver with mount verification.
01CB 563 ;
01CB 564 ; This routine is called by EXE$MOUNTVER to perform some driver-specific
01CB 565 ; actions necessary for mount verification. This routine is used by non-
01CB 566 ; CLASS drivers, and is called by default if EXE$MOUNTVER finds the address
01CB 567 ; of IOC$RETURN in DDT$_MNTVER.
01CB 568 ;
01CB 569 ; Inputs:
01CB 570 ;
01CB 571 ;         R3      = IRP address or 0
01CB 572 ;         R5      = UCB address
01CB 573 ;
01CB 574 ; Outputs:
01CB 575 ;
01CB 576 ;         None.
01CB 577 ;
01CB 578 ; Side effects:
01CB 579 ;
01CB 580 ;         If R3 contains an IRP address, the IRP will be queued to the
01CB 581 ;         head of the UCB's IRP work queue. If R3 contains is zero, then
01CB 582 ;         remove the IRP from the head of the UCB's work queue and attempt
01CB 583 ;         to initiate the I/O.
01CB 584 ;--
01CB 585
01CB 586 IOC$MNTVER::
53  D5 01CB 587      TSTL    R3              ;Driver-specific mount verification code
B3  13 01CD 588      BEQL   NXTIRP         ;Check IRP address
63  0E 01CF 589      INSQUE  IRP$_IOQFL(R3),- ;Branch if none
4C  A5 01D1 590      UCB$_IOQFL(R5)       ;Requeue the IRP
05  01D3 591      RSB                    ;
                                           ;Return

```

```

-1          01D4 593          .SBTTL  INITIATE I/O FUNCTION ON DEVICE
          01D4 594 ;+
          01D4 595 ; IOC$INITIATE - INITIATE NEXT FUNCTION ON DEVICE
          01D4 596 ;
          01D4 597 ; THIS ROUTINE IS CALLED TO INITIATE THE NEXT FUNCTION ON A DEVICE BY CLEARING
          01D4 598 ; STATUS BITS, SETTING THE OPERATION START TIME IF A DIAGNOSTIC BUFFER IS
          01D4 599 ; SPECIFIED, AND CALLING THE DRIVER AT ITS START I/O ENTRY POINT.
          01D4 600 ;
          01D4 601 ; INPUTS:
          01D4 602 ;
          01D4 603 ;           R3 = ADDRESS OF I/O REQUEST PACKET.
          01D4 604 ;           R5 = DEVICE UNIT UCB ADDRESS.
          01D4 605 ;
          01D4 606 ; OUTPUTS:
          01D4 607 ;
          01D4 608 ;           CANCEL I/O, POWERFAIL, AND TIME OUT STATUS BITS ARE CLEARED, THE
          01D4 609 ;           CURRENT SYSTEM TIME IS FILLED INTO THE INTERNAL DIAGNOSTIC BUFFER
          01D4 610 ;           IF ONE IS SPECIFIED, AND THE DRIVER IS CALLED AT ITS START I/O ENTRY
          01D4 611 ;           POINT.
          01D4 612 ; -
          01D4 613
          01D4 614 IOC$INITIATE::          ;INITIATE I/O FUNCTION
          58 A5 53 D0 01D4 615          MOVL   R3,UCB$_IRP(R5)          ;SAVE I/O PACKET ADDRESS
          01D8 616
          01D8 617          .IF DF  CA$_MEASURE_IOT
          01D8 618
          ;ROW0188          00000000'GF 16 01D8 .1          JSB    GAPMS$START_IO          ;INSERT START OF I/O TRANSACTION MESSAGE
          -1
          01DE 620
          01DE 621          .ENDC
          01DE 622
          78 A5 2C A3 7D 01DE 623          MOVQ   IRP$_SVAPTE(R3),UCB$_SVAPTE(R5) ;COPY TRANSFER PARAMETERS
          0048 8F AA 01E3 624          BICW   #UCB$_CANCEL!UCB$_TIMEOUT,UCB$_STS(R5) ;CLEAR CANCEL AND TIME OUT
          64 A5 01E7
          0B 2A A3 07 E1 01E9 625          BBC    #IRP$_DIAGBUF,IRP$_STS(R3),10$ ;IF CLR, NO DIAGNOSTIC BUFFER
          50 4C B3 D0 01EE 626          MOVL   @IRP$_DIAGBUF(R3),R0          ;GET ADDRESS OF DIAGNOSTIC BUFFER DATA AREA
          00000000'EF 7D 01F2 627          MOVQ   EXE$GQ_$SYSTEME,(R0)          ;INSERT I/O OPERATION START TIME
          60 01F8
          50 0088 C5 D0 01F9 628 10$: MOVL   UCB$_DDT(R5),R0          ;GET ADDRESS OF DRIVER DISPATCH TABLE
          00 B0 17 01FE 629          JMP    @DDT$_START(R0)          ;START I/O OPERATION

```



```

-1      0201 631      .SBTTL Allocate Buffered Data Path
        0201 632 ;+
        0201 633 ; ALLOCATE BUFFERED DATA PATH CODE -
        0201 634 ;
        0201 635 ; IOC$REQDATAP - Entrypoint (called from traditional drivers) where caller
        0201 636 ; wishes to be queued (using UCB fork block) if no buffered data path
        0201 637 ; is available at the time of the call.
        0201 638 ; INPUT:
        0201 639 ; R5 => UCB.
        0201 640 ;
        0201 641 ; IOC$REQDATAPNW - Entrypoint to call when caller does not want to wait for
        0201 642 ; unavailable data path.
        0201 643 ; INPUT:
        0201 644 ; R5 => UCB
        0201 645 ;
        0201 646 ; IOC$REQDATAPUDA - Entrypoint (called from UDA port driver) where CDRP
        0201 647 ; is used as the source of information about the request and where
        0201 648 ; the caller does not want to wait for unavailable datapath.
        0201 649 ;
        0201 650 ; INPUT:
        0201 .1 ; R4 => PDT
        0201 651 ; R5 => CDRP
        0201 652 ; -
        0201 653 ;
        0201 654 IOC$REQDATAP::
            10 10 0201 655 BSBB IOC$REQDATAPNW ; Try to alloc. and get control after.
            OC 50 E8 0203 656 BLBS R0,10$ ; LBS implies allocation success.
            0206 657
            10 A5 53 7D 0206 658 MOVQ R3,UCB$L_FR3(R5) ; Save driver context in UCB fork block.
            OC A5 8ED0 020A 659 POPL UCB$L_FPC(R5) ; Save caller's return point.
            65 0E 020E 660 INSQUE UCB$L_FQFL(R5),- ; Queue fork block to resource wait queue.
            18 B1 05 0210 661 @ADP$L_DPQBL(R1) ; Assumes IOC$ALODATAP saves R1=>ADP.
            0212 662 10$: RSB ; Return to caller or caller's caller.
            0213 663
            0213 664 IOC$REQDATAPNW::
            50 24 A5 D0 0213 665 MOVL UCB$L_CRB(R5),R0 ; R0=>CRB.
            51 38 A0 D0 0217 666 MOVL CRB$L_INTD+VEC$L_ADP(R0),R1 ; R1=>ADP (pass to IOC$ALODATAP)
            52 34 A0 9E 021B 667 MOVAB CRB$L_INTD+VEC$W_MAPREG(R0),R2 ; R2=>UBMD " " "
            021F 668
            40 11 021F 669 BRB IOC$ALODATAP ; NOWAIT, RSB from IOC$ALODATAP
            0221 670 ; returns to our caller.
            0221 671 IOC$REQDATAPUDA::
            51 00E0 C4 D0 0221 .1 MOVL PDT$L_ADP(R4),R1 ; R1=>ADP (pass to IOC$ALODATAP)
            52 3C A5 9E 0226 .2 MOVAB CDRP$L_UBARSRCE(R5),R2 ; R2=>UBMD " " "
            ;RLRPTADP 022A .3
            ;RLRPTADP 022A .4 BSBB IOC$ALODATAP ; Call to allocate a data path.
            ;RLRBYTEOFF 35 10 022A .4 ;
            ;RLRBYTEOFF 2D 50 E8 022C .5 BLBS R0,20$ ; LBS means we got one.
            ;RLRBYTEOFF 29 D0 A5 E9 022F .6 BLBC CDRP$W_BOFF(R5),20$ ; LBC means, user buffer is on an
            ;RLRBYTEOFF 0233 .7 ; even byte address so we can use
            ;RLRBYTEOFF 0233 .8 ; the Direct Data Path.
            ;RLRBYTEOFF 0233 .9
            ;RLRBYTEOFF 0233 .10 ; Here we have a transfer to a user buffer located at an odd byte address.
            ;RLRBYTEOFF 0233 .11 ; On those processors which support Byte Offset on the Direct Datapath, we
            ;RLRBYTEOFF 0233 .12 ; can continue processing. On other processors, we must wait for a buffered
            ;RLRBYTEOFF 0233 .13 ; datapath.
            ;RLRBYTEOFF 0233 .14
            ;KPL0001 0233 .15 CPUDISP <<780,10$>,- ; On 11-780 we wait.

```

```

;KPL0001          0233 .16          <750,20$>,-          ; On 11-750 we continue.
;KPL0001          0233 .17          <730,20$>,-          ; On 11-730 we continue.
;KPL0001          0233 .18          <790,10$>,-          ; On 11-790 we wait.
;KPL0001          0233 .19          <8SS,10$>,-          ; On SCORPIO we wait.
;KPL0001          0233 .20          <8NN,10$>,-          ; On NAUTILUS we wait.
;KPL0001          0233 .21          <UV1,30$>>          ; On MicroVAX we bugcheck.
;KPL0001          024D .22
;KPL0001          10 A5 53 7D 024D .23 10$: MOVQ R3,CDRP$L_FR3(R5)          ; Save driver context in CDRP fork block.
;RLRBYTEOFF       0C A5 8ED0 0251 .24      POPL CDRP$L_FPC(R5)          ; Save caller's return point.
;RLRBYTEOFF       28 B5 B6 0255 .25      INCW @CDRP$L_RWCPTR(R5)          ; Increment RWAITCNT.
;RLRBYTEOFF       65 0E 0258 .26      INSQUE CDRP$L_FQFL(R5),-          ; Queue fork block to resource wait queue.
;RLRBYTEOFF       18 B1 025A .27          @ADP$L_DPQBL(R1)          ; Assumes IOC$ALODATAP saves R1=>ADP.
;RLRBYTEOFF       025C .28 20$:
;RLRBYTEOFF       05 025C .29          RSB          ; Return to caller or caller's caller.
;RLRBYTEOFF       025D .30
;KDM0084          025D .31 30$: BUG_CHECK          IVBYTEALGN,FATAL

```

```

-9          0261 681 ; IOC$ALODATAP - Common subroutine called by above routines to allocate
          0261 682 ;           a UNIBUS buffered datapath.
          0261 683 ;
          0261 684 ; INPUTS:
          0261 685 ;           R1 => ADP wherein the datapath allocation bit map is stored.
          0261 686 ;           R2 => UBA mapping descriptor in user's data structure.
          0261 687 ;
          0261 688 ; OUTPUTS:
          0261 689 ;           R0 LBS - implies allocation success
          0261 690 ;           datapath field in R2 => UBA mapping descriptor is set to the
          0261 691 ;           number of the datapath allocated.
          0261 692 ;           appropriate bit in datapath allocation bit map is cleared.
          0261 693 ;           R0 LBC - implies allocation failure.
          0261 694 ;
          0261 695 ;
          0261 696 IOC$ALODATAP:
07      E0 0261 697         BBS      #VEC$V_PATHLOCK,-           ; If this user has a permanently allocated
17 03 A2 0263 698         UBMD$B_DATAPATH(R2),10$ ; datapath, branch around to success.
          0266 699
          0266 700         ASSUME  ADP$C_NUMDATAP EQ 16
00      EA 0266 701         FFS      #0,-                       ; Find first available datapath,
10      A1 0268 702                  #ADP$C_NUMDATAP,-         ; according to bit map. Note failure
60      A1 0269 703                  ADP$W_DPBITMAP(R1),-     ; leaves R0 with the value "16", an
50      12 026B 704                  R0                        ; even number with the low bit clear.
12      13 026C 705         BEQL    20$                        ; EQL implies failure.
          026E 706
          50      F0 026E 707         INSV   R0,-                ; Upon success, R0 has number of the
          0270 708                  #VEC$V_DATAPATH,-         ; available datapath to allocate.
05      00 0270 709                  #VEC$$_DATAPATH,-       ; So we update the user's datapath
03      A2 0272 710                  UBMD$B_DATAPATH(R2)      ; descriptor pointed at by R2.
          0274 711
04      60 A1 50      E4 0274 712         BBSC   R0,ADP$W_DPBITMAP(R1),10$ ; And we update the bit map.
          0279 713         BUG_CHECK INCONSTATE                ; We shouldn't be here obviously.
          027D 714
          50      01 027D 715 10$:      MOVL    S^#SS$_NORMAL,R0 ; Indicate allocation success.
          05      0280 716 20$:      RSB      ; And we return to our caller.

```

```

-1          0281 718          .SBTTL Release Buffered Data Path
          0281 719 ;+
          0281 720 ; RELEASE BUFFERED DATA PATH CODE -
          0281 721 ;
          0281 722 ; IOC$RELDATAPUDA - Entry point called from UDA port driver in response
          0281 723 ; to an UNMAP call. Here the data as to the buffered data path
          0281 724 ; is in the CDRP.
          0281 725 ;
          0281 726 ; INPUTS:
;RLRPDTADP 0281 .1 ; R4 => PDT
          0281 727 ; R5 => CDRP
          0281 728 ;
          0281 729 ; IOC$RELDATAP - Entry point called from traditional drivers to release
          0281 730 ; the buffered datapath described in CRB$L_INTD+VEC$B_DATAPATH.
          0281 731 ;
          0281 732 ; INPUTS:
          0281 733 ; R5 => UCB
          0281 734 ;
          0281 735 ; OUTPUTS:
          0281 736 ; Datapath re-allocated (if any waiters). R0, R1, and R2 modified.
          0281 737 ; NOTE: Since calls to IOC$REQDATAPUDA are NOWAIT, fork blocks dequeued
          0281 738 ; here from ADP$L_DPQFL are guaranteed to be UCB's.
          0281 739 ;
          0281 740
          0281 741 IOC$RELDATAPUDA::
;RLRPDTADP 51 00E0 C4 D0 0281 .1 MOVL PDT$L_ADP(R4),R1 ; R1 => ADP.
;RLRPDTADP 52 3C A5 9E 0286 .2 MOVAB CDRP$L_UBARSRCE(R5),R2 ; R2 => UBMD.
;RLRPDTADP 0C 11 028A .3 BRB RELDATAP_COMMON
;RLRPDTADP 028C .4 IOC$RELDATAP::
;RLRPDTADP 50 24 A5 D0 028C .5 MOVL UCB$L_CRB(R5),R0 ; R0 => CRB.
;RLRPDTADP 52 34 A0 9E 0290 .6 MOVAB CRB$L_INTD+VEC$W_MAPREG(R0),R2 ; R2 => UBMD.
;RLRPDTADP 51 38 A0 D0 0294 .7 MOVL CRB$L_INTD+VEC$L_ADP(R0),R1 ; R1 => ADP.
;RLRPDTADP 0298 .8 RELDATAP_COMMON:
-9 50 03 A2 98 0298 751 CVTBL UBMD$B_DATAPATH(R2),R0 ; Get datapath designator.
          029C 752 BLEQ 10$ ; If LSS permanent assignment.
          029E 753 ; If EQL we had NO datapath to
          029E 754 ; release.
          029E 755 INSV #0,- ; Zero datapath number.
          05 00 F0 029E 756 #VEC$V_DATAPATH,#VEC$S_DATAPATH,-
          03 00 02A0 756 UBMD$B_DATAPATH(R2)
;RLRBYTEOFF 00 EF 02A4 .1 EXTZV #VEC$V_DATAPATH,- ; Extract datapath number.
;RLRBYTEOFF 52 50 05 02A6 .2 #VEC$S_DATAPATH,R0,R2 ;
;RLRBYTEOFF 50 14 B1 0F 02A9 .3 REMQUE @ADP$L_DPQFL(R1),R0 ; R0 => next driver fork block
;RLRBYTEOFF 26 1D 02AD .4 BVS 20$ ; If VS no driver process waiting
;RLRBYTEOFF 02AF .5
;RLRBYTEOFF 7E 53 7D 02AF .6 MOVQ R3,-(SP) ; Save R3, R4, R5
;RLRBYTEOFF 55 DD 02B2 .7 PUSHL R5
;RLRBYTEOFF 55 50 D0 02B4 .8 MOVL R0,R5 ; R5 => driver fork block.
;RLRBYTEOFF 10 91 02B7 .9 CMPB #DYN$C_UCB,- ; See if we dequeued a UCB or a CDRP.
;RLRBYTEOFF 0A A5 02B9 .10 UCB$B_TYPE(R5)
;RLRBYTEOFF 22 12 02BB .11 BNEQ 30$ ; NEQ implies a CDRP.
;RLRBYTEOFF 02BD .12
;RLRBYTEOFF 02BD .13 ; Here we have R5 => UCB.
;RLRBYTEOFF 02BD .14
;RLRBYTEOFF 51 24 A5 D0 02BD .15 MOVL UCB$L_CRB(R5),R1 ; R1 => CRB.
;RLRBYTEOFF 02C1 .16
;RLRBYTEOFF 52 F0 02C1 .17 INSV R2,- ; Store assigned datapath #

```

```

;RLRBYTEOFF          02C3 .18          #VEC$V_DATAPATH,-      ; in CRB.
;RLRBYTEOFF          05 00 02C3 .19          #VEC$S_DATAPATH,-
;RLRBYTEOFF          37 A1 02C5 .20          CRB$L_INTD+VEC$B_DATAPATH(R1)
;RLRBYTEOFF          02C7 .21
;RLRBYTEOFF          53 10 A5 7D 02C7 .22      MOVQ   UCB$L_FR3(R5),R3      ; Restore driver context.
;RLRBYTEOFF          0C B5 16 02CB .23      JSB    @UCB$L_FPC(R5)      ; Call back waiting driver.
;RLRBYTEOFF          02CE .24 5$:
;RLRBYTEOFF          55 8ED0 02CE .25      POPL   R5                  ; Restore deallocator's R5,R4,R3
;RLRBYTEOFF          53 8E 7D 02D1 .26      MOVQ   (SP)+,R3            ;
;RLRBYTEOFF          05 02D4 .27 10$:      RSB                    ;
;RLRBYTEOFF          FA 60 A1 52 E3 02D5 .28 20$: BBS    R2,-                ; Return to deallocator.
;RLRBYTEOFF          02DA .29      ADP$W_DPBITMAP(R1),10$    ; Set datapath bit and exit
;RLRBYTEOFF          02DA .30      BUG_CHECK INCONSTATE    ; Inconsistent state.
;RLRBYTEOFF          05 02DE .31      RSB                    ;
;RLRBYTEOFF          02DF .32
;RLRBYTEOFF          02DF .33      ; Here we have R5 => CDRP.
;RLRBYTEOFF          02DF .34
;RLRBYTEOFF          02DF .35 30$:
;RLRBYTEOFF          52 F0 02DF .36      INSV   R2,-                ; Store assigned datapath #
;RLRBYTEOFF          02E1 .37      #VEC$V_DATAPATH,-      ; in CDRP field.
;RLRBYTEOFF          05 00 02E1 .38      #VEC$S_DATAPATH,-
;RLRBYTEOFF          3F A5 02E3 .39      CDRP$L_UBARSRCE+UBMD$B_DATAPATH(R5)
;RLRBYTEOFF          02E5 .40
;RLRBYTEOFF          00000000'EF 16 02E5 .41      JSB    SCSS$RESUMEWAITR    ; Resume waiting thread and any backed
;RLRBYTEOFF          02EB .42      ; up IRP's.
;RLRBYTEOFF          E1 11 02EB .43      BRB    5$                  ; Branch back to resume deallocator's
;RLRBYTEOFF          02ED .44      ; thread.

```

-20

```

02ED 778 .SBTTL REQUEST AND ALLOCATE UNIBUS MAP REGISTERS FOR CLASS DRIVER
02ED 779 ;+
02ED 780 ; IOC$REQMAPUDA - REQUEST AND ALLOCATE UNIBUS MAP REGISTERS FOR CLASS DRIVER
02ED 781 ;
02ED 782 ; THIS ROUTINE IS CALLED TO ALLOCATE UBA MAP REGISTERS AND TO MARK THE ALLOCATION
02ED 783 ; IN THE UBA MAP REGISTER ALLOCATION DATA STRUCTURES.
02ED 784 ;
02ED 785 ; INPUTS:
02ED 786 ;
02ED 787 ; R4 = ADDRESS OF PORT DESCRIPTOR TABLE.
02ED 788 ; R5 = ADDRESS OF CLASS DRIVER REQUEST PACKET (CDRP).
02ED 789 ;
02ED 790 ; OUTPUTS:
02ED 791 ;
02ED 792 ; IF MAP REGISTERS ARE ALLOCATED FOR THE CDRP, THE APPROPRIATE FIELDS
02ED 793 ; IN THE CDRP ARE MODIFIED TO INDICATE WHICH REGISTERS, AND THE NUMBER
02ED 794 ; OF REGISTERS THAT HAVE BEEN ALLOCATED. ALSO THE ALLOCATION DATA
02ED 795 ; STRUCTURE IN THE ADP IS MODIFIED.
02ED 796 ;
02ED 797 ; IF MAP REGISTERS CANNOT BE ALLOCATED AT THIS TIME, THE CDRP IS
02ED 798 ; QUEUED ONTO THE RESOURCE WAIT LIST AND THE UCB$W_RWAITCNT IS
02ED 799 ; INCREMENTED.
02ED 800 ;
02ED 801 ;-
02ED 802
02ED 803 IOC$REQMAPUDA:: ; Allocate UBA map registers for class drive
25 10 02ED 804 BSBB IOC$ALOMAPUDA ; Call to allocate map registers if availabl
02EF 805 ; Returns R2 => ADP.
02EF 806
02EF 807 ; If here, low bit of R0 tells us whether we were successful in the allocation
02EF 808 ; attempt.
02EF 809
0F 50 E8 02EF 810 BLBS R0,10$ ; Branch around if successful.
10 A5 53 7D 02F2 811 MOVQ R3,CDRP$L_FR3(R5) ; Save driver process context
28 B5 B6 02F6 812 INCW @CDRP$L_RWCPTR(R5) ; One more CDRP, on this UCB, awaiting
; resources.
0C A5 8ED0 02F9 814 POPL CDRP$L_FPC(R5) ; Save map register wait return address
65 0E 02FD 815 INSQUE CDRP$L_FQFL(R5),-
34 B2 02FF 816 @ADP$L_MRQBL(R2) ; Insert process in map register wait queue
05 0301 817 10$: RSB ;

```

```

;ACG0399      0302 .2      .SBTTL REQUEST UNIBUS MAP REGISTERS
;ACG0399      0302 .3 ;+
;ACG0399      0302 .4 ; IOC$REQMAPREG - REQUEST UNIBUS MAP REGISTERS
-5            0302 823 ;
            0302 824 ; THIS ROUTINE IS CALLED TO REQUEST UNIBUS MAP REGISTERS TO PERFORM AN
            0302 825 ; I/O TRANSFER.
            0302 826 ;
            0302 827 ; INPUTS:
            0302 828 ;
-1            0302 830 ;      R5 = UCB ADDRESS OF DEVICE UNIT.
            0302 831 ;      04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
            0302 832 ;
            0302 833 ;      IT IS ASSUMED THAT THE CALLER OWNS THE I/O CHANNEL ON WHICH THE
            0302 834 ;      TRANSFER IS TO OCCUR ON.
            0302 835 ;
            0302 836 ; OUTPUTS:
            0302 837 ;
            0302 838 ;      IF MAP REGISTERS HAVE BEEN PERMANENTLY ASSIGNED TO THE ASSOCIATED
            0302 839 ;      I/O CHANNEL, THEN CONTROL IS IMMEDIATELY RETURNED TO THE CALLER.
            0302 840 ;      ELSE AN ATTEMPT IS MADE TO ALLOCATE THE REQUESTED NUMBER OF MAP REG-
            0302 841 ;      ITERS. IF SUFFICIENT CONTIGUOUS MAP REGISTERS ARE FOUND, THEN THEY
            0302 842 ;      ARE ASSIGNED TO THE ASSOCIATED I/O CHANNEL AND CONTROL IS RETURNED
            0302 843 ;      TO THE CALLER. ELSE THE DRIVER PROCESS CONTEXT IS SAVED IN ITS FORK
            0302 844 ;      BLOCK, THE FORK BLOCK IS INSERTED IN THE MAP REGISTER WAIT QUEUE,
            0302 845 ;      AND A RETURN TO THE DRIVER PROCESS' CALLER IS EXECUTED.
            0302 846 ; -
            0302 847
;KTA3018      0302 .1 IOC$REQMAPREG:: ;REQUEST UNIBUS MAP REGISTERS
;KTA3018      0302 .2      BSBB      IOC$ALOUBAMAP      ; ALLOCATE UBA MAP REGISTER
;KTA3018      0304 .3      BLBS      R0,10$             ; IF LBS SUCCESSFUL ALLOCATION
;KTA3018      10 A5 53 7D 0307 .4      MOVQ      R3,UCB$L_FR3(R5) ;SAVE DRIVER PROCESS CONTEXT
;KTA3018      030B .5      POPL      UCB$L_FPC(R5)      ;SAVE MAP REGISTER WAIT RETURN ADDRESS
;KTA3018      34 B2 65 0E 030F .6      INSQUE   UCB$L_FQFL(R5),@ADP$L_MRQBL(R2) ;INSERT PROCESS IN MAP REGISTER WAIT
;KTA3018      05 0313 .7 10$:      RSB              ;

```

```

-14          0314 862          .SBTTL  ALLOCATE UNIBUS MAP REGISTERS
0314 863 ;+
0314 864 ; IOC$ALOUBAMAP - ALLOCATE UBA MAP REGISTERS (CRB DATABASE SPECIFIED)
0314 865 ; IOC$ALOUBAMAPN - ALLOCATE UBA MAP REGISTERS (ARGUMENT SPECIFIED)
0314 866 ; IOC$ALOMAPUDA - ALLOCATE UBA MAP REGISTERS (FOR CLASS DRIVER(S))
0314 867 ;
0314 868 ; This routine is called to allocate uba map registers and to mark the allocation
0314 869 ; in the map register allocation structure located in the ADP. The state
0314 870 ; of the UNIBUS map registers is maintained in a set of descriptors
0314 871 ; that describe contiguous extents of allocatable (i.e. free) map
0314 872 ; registers. A map register descriptor consists of the
0314 873 ; corresponding elements of two distinct arrays (of one word items)
0314 874 ; located in the ADP. These arrays, ADP$W_MRNREGARY and ADP$W_MRFREGARY,
0314 875 ; contain the number of map registers and the first map register in each
0314 876 ; contiguous extent of free map registers. These arrays are each
0314 877 ; preceded by a one word field containing all 1's (-1) so that compares
0314 878 ; made against the "previous" descriptor fail when the current descriptor
0314 879 ; is the one whose index is zero.
0314 880 ;
0314 881 ; ADP$L_MRACTMDRS maintains the number of active descriptors, i.e. the
0314 882 ; number of elements of each array which contain valid data.
0314 883 ;
0314 884 ; INPUTS: (FOR IOC$ALOUBAMAP AND ALOUBAMAPN)
0314 885 ; R3 = NUMBER OF MAP REGISTERS TO ALLOCATE (IOC$ALOUBAMAPN only).
0314 886 ; R5 = DEVICE UNIT UCB ADDRESS.
0314 887 ;
0314 888 ; INPUT: (FOR IOC$ALOMAPUDA)
;RLRPDTADP 0314 .1 ; R4 => PDT
0314 889 ; R5 => CDRP
0314 890 ;
0314 891 ; OUTPUTS:
0314 892 ; R0 = SUCCESS INDICATION.
0314 893 ; R2 => ADP
0314 894 ;-
0314 895 .enabl 1sb
0314 896 IOC$ALOMAPUDA:
7E 53 7D 0314 897 MOVQ R3,-(SP) ; Save R3,R4,R5
55 DD 0317 898 PUSHL R5 ;
;RLRPDTADP 52 00E0 C4 D0 0319 899 .1 MOVL PDT$_ADP(R4),R2 ; R2 => ADP before we modify R4.
;RLRPDTADP 031E .2
;RLRPDTADP 53 D2 A5 D0 031E .3 MOVL CDRP$_BCNT(R5),R3 ; Get transfer byte count
;RLRPDTADP 54 D0 A5 3C 0322 .4 MOVZWL CDRP$_BOFF(R5),R4 ; Get byte offset in page
;RLRPDTADP 53 03FF C344 9E 0326 .5 MOVAB ^X3FF(R3)[R4],R3 ; Calculate highest relative byte and round
;RLRPDTADP 53 F7 8F 78 032C .6 ASHL #-9,R3,R3 ; Calculate number of map registers required
53 53
-8          0331 908
51 3C A5 9E 0331 909 MOVAB CDRP$_UBARSRCE(R5),R1 ; R1 => UBMD.
2F 11 0335 910 BRB COMMON_ALOUBAMAP ; Branch to common code.
0337 911
0337 912 IOC$ALOUBAMAPN:: ;ALLOCATE UBA MAP REGISTERS ARGUMENT SPECIFI
7E 53 7D 0337 913 MOVQ R3,-(SP) ; Save R3,R4,R5
55 DD 033A 914 PUSHL R5 ;
18 11 033C 915 BRB 5$ ;
033E 916
033E 917 IOC$ALOUBAMAP:: ;ALLOCATE UBA MAP REGISTERS CRB SPECIFIED
7E 53 7D 033E 918 MOVQ R3,-(SP) ; Save R3,R4,R5

```



```

55 DD 0341 919 PUSH R5 ;
      0343 920
53 7E A5 3C 0343 921 MOVZWL UCB$W_BCNT(R5),R3 ;GET TRANSFER BYTE COUNT
54 7C A5 3C 0347 922 MOVZWL UCB$W_BOFF(R5),R4 ;GET BYTE OFFSET IN PAGE
53 03FF C344 9E 034B 923 MOVAB ^X3FF(R3)[R4],R3 ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
53 F7 8F 78 0351 924 ASHL #-9,R3,R3 ;CALCULATE NUMBER OF MAP REGISTERS REQUIRED
      53 0355
      0356 925 5$:
51 24 A5 D0 0356 926 MOVL UCB$L_CRB(R5),R1 ; R1 => CRB.
52 38 A1 D0 035A 927 MOVL CRB$L_INTD+VEC$L_ADP(R1),R2 ; R2 => ADP.
51 34 A1 9E 035E 928 MOVAB CRB$L_INTD+VEC$W_MAPREG(R1),R1 ; R1 => UBMD.
      0F E0 0362 929 BBS #VEC$V_MAPLOCK,- ; If SET, already permanently
      38 61 0364 930 UBMD$W_MAPREG(R1),40$ ; allocated, so branch around.
      0366 931
      0366 932 ; Here:
      0366 933 ; R1 => UBMD - caller's structure where we record registers allocated
      0366 934 ; R2 => ADP
      0366 935 ; R3 = number of map registers to allocate
      0366 936 ;
      0366 937
      0366 938 COMMON_ALOUBAMAP:
;RLR0001 5C A2 D5 0366 .1 TSTL ADP$L_MRACTMDRS(R2) ; Test for zero active descriptors.
;RLR0001 13 13 0369 .2 BEQL 15$ ; EQL implies no registers available.
      53 D6 036B 939 INCL R3 ; Round up request to next multiple
      01 8A 036D 940 BICB #1,R3 ; of 2.
      55 D4 0370 941 CLRL R5 ; Establish loop variable.
      0372 942 10$:
64 A245 53 B1 0372 943 CMPW R3,ADP$W_MRNREGARY(R2)[R5] ; See if enough regs described here.
      09 15 0377 944 BLEQ 20$ ; LEQ implies YES.
      0379 945
;RLR0001 F4 55 5C A2 F2 0379 946 AOBLS ADP$L_MRACTMDRS(R2),R5,10$ ; Else branch back and continue
      037E .1 15$:
      50 D4 037E 947 CLRL R0 ; If here, allocation failure.
      1F 11 0380 948 BRB 50$ ; Branch around to return.
      0382 949 20$:
61 015E C245 B0 0382 950 MOVW ADP$W_MRFREGARY(R2)[R5],- ; Allocate from low end of extent
      0388 951 UBMD$W_MAPREG(R1) ; by copying 1st map reg. #.
      02 A1 53 90 0388 952 MOVAB R3,UBMD$B_NUMREG(R1) ; Set # of map regs allocated.
64 A245 53 A2 038C 953 SUBW R3,ADP$W_MRNREGARY(R2)[R5] ; Subtract out # regs allocated.
      05 12 0391 954 BNEQ 30$ ; NEQ implies extent not empty,
      0393 955 ; branch around deallocate.
      0129 30 0393 956 BSBW DEALLOC_DESCRIP ; Call to deallocate descriptor.
      06 11 0396 957 BRB 40$ ; And branch back to return.
      0398 958 30$:
015E C245 53 A0 0398 959 ADDW R3,ADP$W_MRFREGARY(R2)[R5] ; Bump descriptor past
      039E 960 ; allocated registers.
      50 01 D0 039E 961 40$: MOVL S^#SS$_NORMAL,R0 ; Indicate success.
      55 8ED0 03A1 962 50$: POPL R5 ; Restore R5,R4,R3
      53 8E 7D 03A4 963 MOVQ (SP)+,R3 ;
      05 03A7 964 RSB
      03A8 965 .dsab1 1sb

```

```

-1      03A8 967      .SBTTL Allocate a specific set of UNIBUS Map Registers
      03A8 968 ;+
      03A8 969 ; IOC$ALOUBAMAPSP
      03A8 970 ;
      03A8 971 ; This routine is called to allocate a specific set of UNIBUS Map Registers.
      03A8 972 ;
      03A8 973 ; INPUTS:
      03A8 974 ;     R3 = # of map registers to allocate
      03A8 975 ;     R4 = # of first map register to allocate
      03A8 976 ;     R5 => UCB
      03A8 977 ;
      03A8 978 ; OUTPUTS:
      03A8 979 ;     R0 = Success or failure indication
      03A8 980 ;     Note R0, R1 and R2 modified.
      03A8 981 ; -
      03A8 982
      03A8 983 IOC$ALOUBAMAPSP:
      03A8 984      MOVQ R3,-(SP)          ; Save R3,R4,R5
      03A8 985      PUSHL R5
      03A8 986
      03AD 987      MOVL UCB$L_CRB(R5),R0      ; R0 => CRB.
      03B1 988      MOVL CRB$L_INTD+VEC$L_ADP(R0),R2      ; R2 => ADP.
      03B5 989      MOVAB CRB$L_INTD+VEC$W_MAPREG(R0),R1 ; R1 => UBA mapping descriptor.
      03B9 990
      03B9 991      TSTL ADP$L_MRACTMDRS(R2)      ; Test for zero active descriptors.
      03BC 992      BEQL 30$                    ; EQL implies no registers available.
      03BE 993      BLBC R4,10$                ; Prepare to round DOWN to even boundary.
      03C1 994      BICB #1,R4                 ; Clear low bit if set and
      03C4 995      INCL R3                     ; then increment # of registers to allocate
      03C6 996      10$:
      03C6 997      INCL R3                     ; Prepare to round UP to even # of registers
      03C8 998      BICB #1,R3
      03CB 999
      03CB 1000     CLRL R5                      ; R5 will be index register.
      03CD 1001     20$:
      03CD 1002     CMPW R4,ADP$W_MRFREGARY(R2)[R5] ; Are registers we want in
      03D3 1003     BLSS 30$                    ; current extent?
      03D5 1004     ; LSS means current extent is beyond the
      03D5 1005     BEQL 50$                    ; desired registers. Therefore they are
      03D7 1006     ; not available and we have failed.
      03D7 1007     ; EQL means they are at the beginning
      03D7 1008     ; of the current extent.
      03D7 1009     ; Here the registers we want are either within the middle of the current
      03D7 1010     ; extent or else beyond the current extent.
      03D7 1011     ADDW3 ADP$W_MRFREGARY(R2)[R5],- ; R0 = 1st register beyond
      03DC 1012     ; current extent.
      03E0 1013     CMPW R4,R0                  ; Are we in current extent?
      03E3 1014     BLSS 40$                    ; LSS means YES, in current.
      03E5 1015     AOBLS ADP$L_MRACTMDRS(R2),R5,20$ ; Loop thru all extents.
      03EA 1016     30$:
      03EA 1017     CLRL R0                      ; Set failure code.
      03EC 1018     BRB 80$                     ; And branch to return.
      03EE 1019     40$:
      03EE 1020

```

;RLR0003  
;RLR0003

```

      7E 53 7D 03A8 984
      55 DD 03AB 985
      50 24 A5 D0 03AD 987
      52 38 A0 D0 03B1 988
      51 34 A0 9E 03B5 989
      5C A2 D5 03B9 990
      2C 13 03BC 991
      05 54 E9 03BE 992
      54 01 8A 03C1 992
      53 D6 03C4 993
      53 D6 03C6 994
      53 01 8A 03C8 996
      55 D4 03CB 997
      55 D4 03CB 998
      015E C245 54 B1 03CD 1000
      15 19 03D3 1001
      03D3 1002
      03D5 1003
      03D5 1004
      46 13 03D5 1005
      03D7 1006
      03D7 1007
      03D7 1008
      03D7 1009
      015E C245 A1 03D7 1011
      50 64 A245 03DC 1012
      50 54 B1 03E0 1013
      09 19 03E3 1014
      E3 55 5C A2 F2 03E5 1015
      50 D4 03EA 1016
      52 11 03EC 1018
      03EE 1019
      03EE 1020

```

```

03EE 1021 ; Here the first register we want is greater than the first register of
03EE 1022 ; current extent (defined by R5 = index) and is less than or equal to
03EE 1023 ; the last register of the extent. R0 contains the # of the register just
03EE 1024 ; beyond the current extent. In other words,
03EE 1025 ;
03EE 1026 ; ADP$W_MRFREGARY(R2)[R5] < R4 < R0
03EE 1027
50 54 A2 03EE 1028 SUBW R4,R0 ; R0 = length of subextent based at R4.
53 50 B1 03F1 1029 CMPW R0,R3 ; Compare to # of registers needed.
F4 19 03F4 1030 BLSS 30$ ; LSS means failure.
03F6 1031
61 54 B0 03F6 1032 MOVW R4,UBMD$W_MAPREG(R1) ; Success. Fill in user's descriptor
02 A1 53 90 03F9 1033 MOVVB R3,UBMD$B_NUMREG(R1) ; with base register and # of registers.
03FD 1034
03FD 1035 ; SUBW3 ADP$W_MRFREGARY(R2)[R5],R4,- ; Distance from beginning of
03FD 1036 ; ADP$W_MRNREGARY(R2)[R5] ; extent to R4 is new length.
64 A245 50 A2 03FD 1037 SUBW R0,ADP$W_MRNREGARY(R2)[R5] ; Equivalent result.
0402 1038
50 53 A2 0402 1039 SUBW R3,R0 ; R0 = # regs. left in sub-extent.
36 13 0405 .1 BEQL 70$ ; EQL means we do not have to allocate
0407 1041 ; and fill a new extent descriptor.
7E 50 B0 0409 1043 INCL R5 ; R5 = index of new extent descriptor.
00C9 30 040C 1044 MOVW R0,-(SP) ; Save length of new extent.
040F 1045 BSBW ALLOC_DESCRIP ; Call to allocate a new descriptor.
53 54 A1 040F 1046 ADDW3 R4,R3,ADP$W_MRFREGARY(R2)[R5] ; Fill in new descriptor with
015E C245 0412
64 A245 8E B0 0416 1047 MOVW (SP)+,ADP$W_MRNREGARY(R2)[R5] ; 1st register and # registers.
20 11 041B 1048 BRB 70$ ; Branch around to success.
041D 1049 50$:
041D 1050
041D 1051 ; Here the first register we want is equal to the first register of the current
041D 1052 ; extent (defined by index register R5). In other words,
041D 1053 ;
041D 1054 ; R4 = ADP$W_MRFREGARY(R2)[R5]
041D 1055
64 A245 53 B1 041D 1056 CMPW R3,ADP$W_MRNREGARY(R2)[R5] ; See if we have enough registers.
C6 14 0422 1057 BGTR 30$ ; GTR implies failure.
0424 1058
61 54 B0 0424 1059 MOVW R4,UBMD$W_MAPREG(R1) ; Success. Fill in user's descriptor
02 A1 53 B0 0427 1060 MOVW R3,UBMD$B_NUMREG(R1) ; with 1st register and # allocated.
042B 1061
64 A245 53 A2 042B 1062 SUBW R3,ADP$W_MRNREGARY(R2)[R5] ; Update current descriptor.
08 13 0430 1063 BEQL 60$ ; EQL means current extent now
0432 1064 ; empty. Go to deallocate.
015E C245 53 A0 0432 1065 ADDW R3,ADP$W_MRFREGARY(R2)[R5] ; If not empty, update 1st register.
03 11 0438 1066 BRB 70$ ; Branch around deallocate.
043A 1067 60$:
0082 30 043A 1068 BSBW DEALLOC_DESCRIP ; Deallocate system descriptor.
50 01 D0 043D 1069 70$: MOVL S^#SS$_NORMAL,R0 ; Set success indicator.
55 BEO 0440 1070 80$: POPL R5 ; Restore R5,R4,R3
53 8E 7D 0443 1071 MOVQ (SP)+,R3 ;
05 0446 1072 RSB ; And return to caller.

```

;RLRMAPSP  
-1

```

-1          0447 1074      .SBTTL  Permanently Allocate UNIBUS Map Registers
          0447 1075 ;+
          0447 1076 ; IOC$ALOUBAMAPRM - Permanently Allocate UBA Map Registers (CRB Database Specified)
          0447 1077 ; IOC$ALOUBAMAPRMN - Permanently Allocate UBA Map Registers (Argument Specified)
          0447 1078 ;
          0447 1079 ; This routine is called to permanently allocate UNIBUS map registers.
          0447 1080 ;      Here we allocate the map registers from the highest numbered
          0447 1081 ;      available registers.
          0447 1082 ;
          0447 1083 ; INPUTS:
          0447 1084 ;      R3 = # Registers to allocate (IOC$ALOUBAMAPRMN only)
          0447 1085 ;      R5 => UCB
          0447 1086 ;
          0447 1087 ; OUTPUTS:
          0447 1088 ;      R0 = Success indication
          0447 1089 ;
          0447 1090 ;-
          0447 1091
          0447 1092      .enabl  LSB
          0447 1093 IOC$ALOUBMAPRMN: ;
          7E 53 7D 0447 1094 MOVQ  R3,-(SP) ;ALLOCATE UBA MAP REGISTERS ARGUMENT SPECIFI
          55 DD 044A 1095 PUSHL  R5 ; Save R3,R4,R5
          044C 1096 ;
          18 11 044C 1097 BRB    5$ ;
          044E 1098 IOC$ALOUBMAPRM: ;ALLOCATE UBA MAP REGISTERS CRB SPECIFIED
          7E 53 7D 044E 1099 MOVQ  R3,-(SP) ; Save R3,R4,R5
          55 DD 0451 1100 PUSHL  R5 ;
          0453 1101 ;
          53 7E A5 3C 0453 1102 MOVZWL UCB$W_BCNT(R5),R3 ;GET TRANSFER BYTE COUNT
          54 7C A5 3C 0457 1103 MOVZWL UCB$W_BOFF(R5),R4 ;GET BYTE OFFSET IN PAGE
          53 03FF C344 9E 045B 1104 MOVAB  ^X3FF(R3)[R4],R3 ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
          53 F7 8F 78 0461 1105 ASHL  #-9,R3,R3 ;CALCULATE NUMBER OF MAP REGISTERS REQUIRED
          0465 ;
          0466 1106 5$:
          51 24 A5 D0 0466 1107 MOVL  UCB$L_CRB(R5),R1 ; R1 => CRB
          52 38 A1 D0 046A 1108 MOVL  CRB$L_INTD+VEC$L_ADP(R1),R2 ; R2 => ADP
          51 34 A1 9E 046E 1109 MOVAB CRB$L_INTD+VEC$W_MAPREG(R1),R1 ; R1 => UBMD.
          38 61 E0 0472 1110 BBS   #VEC$V_MAPLOCK,- ; If SET, already permanently
          0474 1111 UBMD$W_MAPREG(R1),30$ ; allocated, so branch around.
          0476 1112 ;
          53 D6 0476 1113 INCL  R3 ; Round up request to next multiple
          53 01 8A 0478 1114 BICB  #1,R3 ; of 2.
          55 5C A2 D0 047B 1115 MOVL  ADP$L_MRACTMDRS(R2),R5 ; R5 = index beyond last MRD.
          0A 13 047F .1 BEQL  15$ ; EQL implies no registers available.
          0481 .2 10$:
          62 A245 53 B1 0481 .3 CMPW  R3,ADP$W_MRNREGARY-2(R2)[R5] ; See if enough regs described here.
          07 15 0486 .4 BLEQ  20$ ; LEQ implies YES.
          0488 .5 ;
          F6 55 F5 0488 .6 SOBGTR R5,10$ ; Else branch back and continue
          048B .7 15$:
          50 D4 048B 1121 CLRL  R0 ; If here, allocation failure.
          22 11 048D 1122 BRB   40$ ; Branch around to return.
          048F 1123 20$:
          015C C245 A1 048F 1124 ADDW3 ADP$W_MRFREGARY-2(R2)[R5],- ; Calculate register # beyond
          50 62 A245 0494 ;
          0498 1125 ADP$W_MRNREGARY-2(R2)[R5],R0 ; last extent.
          50 53 A2 0498 1126 SUBW  R3,R0 ; We allocate from high end.. R0

```



-1

```

04BF 1146 ;+
04BF 1147 ; DEALLOC_DESCRIP - Common internal subroutine called to deallocate
04BF 1148 ;     a UBA Map Register descriptor.
04BF 1149 ;
04BF 1150 ; INPUTS:
04BF 1151 ;     R2 => ADP
04BF 1152 ;     R5 = index of descriptor to deallocate.
04BF 1153 ; OUTPUTS:
04BF 1154 ;     The UBA Map Allocation structures are updated by contracting
04BF 1155 ;     descriptors over the deallocated one.
04BF 1156 ;     Register R5 is modified.
04BF 1157 ;-
04BF 1158
04BF 1159 DEALLOC_DESCRIP:
    5C A2  D7 04BF 1160     _DECL     ADP$L_MRACTMDRS(R2)      ; Decrement # active descriptors.
    66 A245 B0 04C2 1161 10$:     MOVW     ADP$W_MRNREGARY+2(R2)[R5],-      ; Move data towards lower index
    64 A245     04C6                                     ;
    0160 C245 B0 04C9 1163                                     ADP$W_MRNREGARY(R2)[R5]      ; to fill up hole.
    015E C245     04C9 1164     MOVW     ADP$W_MRFREGARY+2(R2)[R5],-      ;
    EB 55 5C A2 F2 04D2 1165                                     ADP$W_MRFREGARY(R2)[R5]      ;
    05      05 04D7 1166     AOBLS    ADP$L_MRACTMDRS(R2),R5,10$      ; Loop thru rest of active MDRS.
    04D8 1167     RSB
    04D8 1168
    04D8 1169 ;+
    04D8 1170 ; ALLOC_DESCRIP - Common internal subroutine to allocate a UBA map register
    04D8 1171 ;     descriptor in the middle of the range of descriptors.
    04D8 1172 ;
    04D8 1173 ; INPUTS:
    04D8 1174 ;     R2 => ADP
    04D8 1175 ;     R5 = index of where we must allocate descriptor
    04D8 1176 ; OUTPUTS:
    04D8 1177 ;     Allocation is accomplished by creating a hole in each of the arrays
    04D8 1178 ;     by moving descriptor items to the next higher element.
    04D8 1179 ;     Note R0 is modified.
    04D8 1180 ;-
    04D8 1181
    50 5C A2 D0 04D8 1182 ALLOC_DESCRIP:
    55 50 D1 04D8 1183     MOVL     ADP$L_MRACTMDRS(R2),R0      ; R0 = # active descriptors.
    13 15 04DC 1184 10$:     CMPL     R0,R5      ; Have we finished?
    62 A240 B0 04DF 1185     BLEQ     20$      ; LEQ implies YES.
    64 A240     04E1 1187     MOVW     ADP$W_MRNREGARY-2(R2)[R0],-      ; Starting from ends of arrays,
    04E5                                     ;
    04E8 1188     ADP$W_MRNREGARY(R2)[R0]      ; copy # register items.
    04E8 1189
    015C C240 B0 04E8 1190     MOVW     ADP$W_MRFREGARY-2(R2)[R0],-      ;
    015E C240     04ED                                     ;
    EB 50 F5 04F1 1191     ADP$W_MRFREGARY(R2)[R0]      ;
    04F1 1192     SOBGTR   R0,10$      ; And loop back until we reach
    04F4 1193                                     ; the hole we have created.
    5C A2 D6 04F4 1194 20$:     INCL     ADP$L_MRACTMDRS(R2)      ; Increment # active descriptors.
    05      05 04F7 1195     RSB      ; Return to caller
    
```

```

-1          04F8 1197          .SBTTL Release UNIBUS Map Registers
          04F8 1198 ;+
          04F8 1199 ; IOC$RELMAPUDA - RELEASE UNIBUS MAP REGISTERS (CALLED FROM UDA PORT DRIVER)
          04F8 1200 ; IOC$RELMAPREG - RELEASE UNIBUS MAP REGISTERS
          04F8 1201 ;
          04F8 1202 ; This routine is called to release UNIBUS map registers that were previously
          04F8 1203 ; assigned for an I/O transfer.
          04F8 1204 ;
          04F8 1205 ; INPUTS:
          04F8 1206 ; (For IOC$RELMAPUDA only)
          04F8 1207 ;
;RLRPDTADP          04F8 .1 ; R4 => PDT
          04F8 1208 ; R5 => CDRP
          04F8 1209 ;
          04F8 1210 ; (For IOC$RELMAPREG call only)
          04F8 1211 ;
          04F8 1212 ; R5 = UCB ADDRESS OF DEVICE UNIT.
          04F8 1213 ;
          04F8 1214 ; It is assumed that the caller still owns the I/O channel on which
          04F8 1215 ; the transfer took place.
          04F8 1216 ;
          04F8 1217 ; OUTPUTS:
          04F8 1218 ;
          04F8 1219 ; If the mapping registers have been permanently assigned to the asso-
          04F8 1220 ; ciated I/O channel (only possible for IOC$RELMAPREG), then control
          04F8 1221 ; is immediately returned to the caller. Else the mapping registers are
          04F8 1222 ; released (via a call to IOC$DALOCUBAMAP) and we then go into a loop
          04F8 1223 ; removing waiting driver processes from the Map Register Wait Queue
          04F8 1224 ; until either the Queue is completely drained or we run out of map
          04F8 1225 ; registers to satisfy the needs of a given waiting driver process.
          04F8 1226 ; Driver processes waiting here have their context stored in either
          04F8 1227 ; a UCB fork block or a CDRP fork block and the processing required to
          04F8 1228 ; resume each of these types of driver process is slightly different.
          04F8 1229 ; What is done for each is to allocate the required map registers
          04F8 1230 ; (via a call to IOC$ALOUBAMPA for UCB threads and via a call to
          04F8 1231 ; IOC$ALOUBAMAP for CDRP threads) and to resume the waiting driver
          04F8 1232 ; process. Resuming a UCB thread is done by restoring register
          04F8 1233 ; context and JSB'ing to the saved PC. Resuming a CDRP thread is
          04F8 1234 ; accomplished by calling SCS$RESUMEWAITR.
          04F8 1235 ;-
          04F8 1236          .enabl lsb
          04F8 1237 IOC$RELMAPUDA::
          04F8 1238          MOVQ R3,-(SP) ; Save R3-R6
          04F8 1239          MOVQ R5,-(SP) ;
          04FE 1240
;RLRPDTADP          52 00E0 C4 DO 04FE .1          MOVL PDT$_ADP(R4),R2 ; R2 => ADP.
;RLRPDTADP          56 52 DO 0503 .2          MOVL R2,R6 ; R6 => ADP also.
-4          0506 1245
          53 3C A5 9E 0506 1246          MOVAB CDRP$_UBARSRCE(R5),R3 ; R3 => UBMD.
          54 63 3C 050A 1247          MOVZWL UBMD$_MAPREG(R3),R4 ; R4 has 1st mapreg #.
          53 02 A3 9A 050D 1248          MOVZBL UBMD$_NUMREG(R3),R3 ; R3 has # of mapregs.
          1E 11 0511 1249          BRB 10$ ; Branch to common code.
          0513 1250
          0513 1251 IOC$RELMAPREG:: ; Release unibus map registers
          51 24 A5 DO 0513 1252          MOVL UCB$_CRB(R5),R1 ; R1 => CRB.
          OF EO 0517 1253          BBS #VEC$V_MAPLOCK,- ; If SET, permanent allocation so branch.
          3D 34 A1 0519 1254          CRB$_INTD+VEC$_MAPREG(R1),50$

```

```

7E 53 7D 051C 1255      MOVQ   R3,-(SP)          ; Save R3-R6
7E 55 7D 051F 1256      MOVQ   R5,-(SP)          ;
      0522 1257
52 38 A1 D0 0522 1258      MOVL   CRB$L_INTD+VEC$L_ADP(R1),R2 ;GET ADDRESS OF ADP
56 52 D0 0526 1259      MOVL   R2,R6             ;SAVE ADDRESS OF ADP
54 34 A1 3C 0529 1260      MOVZWL CRB$L_INTD+VEC$W_MAPREG(R1),R4 ;GET STARTING MAP REGISTER NUMBER
53 36 A1 9A 052D 1261      MOVZBL CRB$L_INTD+VEC$B_NUMREG(R1),R3 ;GET NUMBER OF REGISTERS TO DEALLOC
      0531 1262 10$:
      0034 30 0531 1263      BSBW   IOC$DALOCUBAMAP      ; Free up UBA map resources.
      0534 1264 20$:
55 30 B6 0F 0534 1265      REMQUE @ADP$L_MRQFL(R6),R5    ;GET ADDRESS OF NEXT DRIVER FORK BLOCK
      19 1D 0538 1266      BVS    40$                ;IF VS NO DRIVER PROCESS WAITING
      053A 1267
      10 91 053A 1268      CMPB   #DYN$C_UCB,-        ; See if we dequeued a UCB or a CDRP.
      0A A5 053C 1269      UCB$B_TYPE(R5)
      1A 12 053E 1270      BNEQ   REALLOC_CD_MAPREGS  ; NEQ implies a CDRP.
      0540 1271
      FDFB 30 0540 1272      BSBW   IOC$ALOUBAMAP      ;SEARCH MAP REGISTER BITMAP AND ALLOCATE
      09 50 E9 0543 1273      BLBC   R0,30$            ;IF LBC ALLOCATION FAILURE
53 10 A5 7D 0546 1274      MOVQ   UCB$L_FR3(R5),R3    ;RESTORE DRIVE PROCESS CONTEXT
      0C B5 16 054A 1275      JSB    @UCB$L_FPC(R5)     ;CALL DRIVER AT MAP REGISTER WAIT RETURN ADD
      E5 11 054D 1276      BRB   20$                ;
30 A6 65 0E 054F 1277 30$:  INSQUE  UCB$L_FQFL(R5),ADP$L_MRQFL(R6) ;REINSERT DRIVER PROCESS AT FRONT OF
55 8E 7D 0553 1278 40$:  MOVQ   (SP)+,R5           ; Restore R3-R6
53 8E 7D 0556 1279      MOVQ   (SP)+,R3           ;
      05 0559 1280 50$:  RSB    ;
      055A 1281
      055A 1282 REALLOC_CD_MAPREGS: ; Reallocate mapregs to a class driver
      055A 1283 ; process.
      FDB7 30 055A 1284      BSBW   IOC$ALOMAPUDA      ; Allocate map registers if we can.
      EF 50 E9 055D 1285      BLBC   R0,30$            ; LBC implies allocation failure, branch
      0560 1286
00000000'EF 16 0560 1287      JSB    SCSS$RESUMEWAITR   ; Resume waiting thread and any backed
      0566 1288 ; up IRP's.
      CC 11 0566 1289      BRB    20$                ; Branch back to try and allocate more
      0568 1290 ; UNIBUS map registers.
      0568 1291      .dsabl 1sb

```



```

-1
0568 1293 ;+
0568 1294 ; IOC$DALOCUBAMAP - Common internal subroutine to update the UBA Map allocation
0568 1295 ; structures to include the map registers specified here among the
0568 1296 ; available map registers.
0568 1297 ;
0568 1298 ; INPUTS:
0568 1299 ; R2 => ADP
0568 1300 ; R3 = # map registers to free.
0568 1301 ; R4 = first map register to free.
0568 1302 ;
0568 1303 ; OUTPUTS:
0568 1304 ; The UBA Map Allocation structures are updated.
0568 1305 ;
0568 1306 ; Registers R0, R1 and R5 are modified.
0568 1307 ;
0568 1308 ;-
0568 1309
0568 1310 IOC$DALOCUBAMAP:
55 D4 0568 1311 CLRL R5 ; Initialize loop variable.
51 53 54 C1 056A 1312 ADDL3 R4,R3,R1 ; R1 = map register beyond extent.
;ROW0140 53 D5 056E .1 TSTL R3 ; Is the # of regs. to deallocate zero?
;ROW0140 62 13 0570 .2 BEQL 90$ ; Branch to bugcheck if zero.
;RLR0003 5C A2 D5 0572 .3 TSTL ADP$L_MRACTMDRS(R2) ; Test for zero active descriptors.
;RLR0003 4E 13 0575 .4 BEQL 50$ ; EQL implies no registers available.
0577 1313 10$:
015E C245 51 B1 0577 1314 CMPW R1,ADP$W_MRFREGARY(R2)[R5] ; See if map registers to free
057D 1315 ; are before those described
057D 1316 ; by current descriptor.
07 15 057D 1317 BLEQ 20$ ; LEQ implies yes.
057F 1318
F3 55 5C A2 F2 057F 1319 AOBLS ADP$L_MRACTMDRS(R2),R5,10$ ; Else branch back and try next.
2B 11 0584 1320 BRB 40$ ; If here, registers to free
0586 1321 ; beyond those described by
0586 1322 ; last descriptor. So branch
0586 1323 ; to try and absorb at end of
0586 1324 ; last descriptor.
29 12 0586 1325 20$:
0586 1326 BNEQ 40$ ; NEQ implies that although we alloca-
0588 1327 ; registers before the current des-
0588 1328 ; criptor, we are not contiguous with
0588 1329 ; it. So we branch to try and absorb
0588 1330 ; these registers in the previous one.
0588 1331
0588 1332 ; Here we can absorb the registers in the current descriptor.
0588 1333
62 A245 A1 0588 1334 ADDW3 ADP$W_MRNREGARY-2(R2)[R5],- ; Calculate end of previous
50 015C C245 058C
0591 1335 ADP$W_MRFREGARY-2(R2)[R5],R0 ; extent and move to R0.
54 50 B1 0591 1336 CMPW R0,R4 ; Does it coincide with start
0594 1337 ; of this extent?
OC 13 0594 1338 BEQL 30$ ; EQL implies yes.
0596 1339
0596 1340 ; Here we have the most likely case. The map registers that we are freeing can
0596 1341 ; be absorbed into the top of the current descriptor but not also in the
0596 1342 ; previous descriptor.
0596 1343
015E C245 54 B0 0596 1344 MOVW R4,ADP$W_MRFREGARY(R2)[R5] ; First register freed becomes

```

```

059C 1345 ; first register of current
059C 1346 ; descriptor.
64 A245 53 A0 059C 1347 ADDW R3,ADP$W_MRNREGARY(R2)[R5] ; Number of registers is sum of
05A1 1348 ; registers freed and registers
05A1 1349 ; previously described here.
05 05A1 1350 RSB
05A2 1351
05A2 1352 ; Here we have the case where the map registers being freed fall between two
05A2 1353 ; discontinuous blocks and exactly span the difference. We then can
05A2 1354 ; describe the entire group with one descriptor, and so we also
05A2 1355 ; deallocate the current descriptor. Note new combined descriptor
05A2 1356 ; will still begin at same map register number so we do NOT alter
05A2 1357 ; this item.
05A2 1358
05A2 1359 30$:
62 A245 53 A0 05A2 1360 ADDW R3,ADP$W_MRNREGARY-2(R2)[R5] ; Partial sum of registers
05A7 1361 ; being freed and previous ones.
64 A245 A0 05A7 1362 ADDW ADP$W_MRNREGARY(R2)[R5],- ; Now add in registers described
62 A245 05AB ; in current descriptor.
05AE 1363 ADP$W_MRNREGARY-2(R2)[R5]
05AE 1364
FF0E 31 05AE 1365 BRW DEALLOC_DESCRIP ; BRW to subroutine and let it
05B1 1366 ; return to our caller.
05B1 1367
05B1 1368 ; Here we cannot absorb the freed map registers in the current descriptor.
05B1 1369 ; We test to see if we can absorb them in the previous descriptor.
05B1 1370
05B1 1371 40$:
50 62 A245 A1 05B1 1372 ADDW3 ADP$W_MRNREGARY-2(R2)[R5],- ; Calculate end of previous
015C C245 05B5 ; extent and move to R0.
05BA 1373 ADP$W_MRFREGARY-2(R2)[R5],R0
54 50 B1 05BA 1374 CMPW R0,R4 ; See if contiguous with previous.
06 12 05BD 1375 BNEQ 50$ ; NEQ implies NO.
05BF 1376
62 A245 53 A0 05BF 1377 ADDW R3,ADP$W_MRNREGARY-2(R2)[R5] ; Sum # of registers in extent.
05 05C4 1378 RSB
05C5 1379
05C5 1380 ; Here we must allocate a new descriptor to describe the map registers we
05C5 1381 ; are freeing. Conditions at this time are as follows:
05C5 1382 ;
05C5 1383 ; R2 => ADP
05C5 1384 ; R3 = # registers to free
05C5 1385 ; R4 = first register to free
05C5 1386 ; R5 = index of where we must allocate descriptor
05C5 1387 ;
05C5 1388 ; Allocation is accomplished by calling subroutine ALLOC_DESCRIP
05C5 1389 ;
05C5 1390
05C5 1391 50$:
FF10 30 05C5 1392 BSBW ALLOC_DESCRIP ; Alloc R5 = index of descriptor.
64 A245 53 B0 05C8 1393 MOVW R3,ADP$W_MRNREGARY(R2)[R5] ; Fill in allocated descriptor.
015E C245 54 B0 05CD 1394 MOVW R4,ADP$W_MRFREGARY(R2)[R5] ;
05 05D3 1395 RSB
05D4 .1
05D4 .2 90$: BUG_CHECK INCONSTATE ; Non-fatal bugcheck on zero map
05D8 .3 ; registers deallocation attempts.
05 05D8 .4 RSB ; Then ignore deallocate request.

```

;ROW0140  
;ROW0140  
;ROW0140  
;ROW0140

-1

```
05D9 1397          .SBTTL RETURN TO CALLER
05D9 1398 ;+
05D9 1399 ; IOC$RETURN - RETURN TO CALLER
05D9 1400 ;
05D9 1401 ; THIS ROUTINE IS CALLED AS A RESULT OF A DDT DISPATCH TO A NULL ENTRY. ITS
05D9 1402 ; FUNCTION IS MERELY TO RETURN TO ITS CALLER.
05D9 1403 ;
05D9 1404 ; INPUTS:
05D9 1405 ;
05D9 1406 ;      NONE.
05D9 1407 ;
05D9 1408 ; OUTPUTS:
05D9 1409 ;
05D9 1410 ;      NONE.
05D9 1411 ; -
05D9 1412
05D9 1413 IOC$RETURN:: ;RETURN TO CALLER
05 05D9 1414      RSB ;
```

-1

```

05DA 1416          .SBTTL  WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
05DA 1417 ;+
05DA 1418 ; IOC$WFIKPCH - WAITFOR INTERRUPT OR TIMEOUT AND KEEP CHANNEL
05DA 1419 ;
05DA 1420 ; THIS ROUTINE IS CALLED TO SOFTWARE ENABLE INTERRUPTS AND TIMEOUT ON
05DA 1421 ; A DEVICE UNIT AND TO KEEP THE CHANNEL. THIS ROUTINE CAN BE CALLED AT
05DA 1422 ; EITHER FORK OR DEVICE INTERRUPT LEVEL.
05DA 1423 ;
05DA 1424 ; INPUTS:
05DA 1425 ;
05DA 1426 ;          00(SP) = RETURN ADDRESS OF CALLER.
05DA 1427 ;          04(SP) = TIMEOUT VALUE IN SECONDS.
05DA 1428 ;          08(SP) = IPL TO LOWER TO AFTER SETTING WAIT.
05DA 1429 ;          12(SP) = RETURN ADDRESS OF CALLER'S CALLER.
05DA 1430 ;
05DA 1431 ;          R5 = UCB ADDRESS OF DEVICE UNIT.
05DA 1432 ;
05DA 1433 ; OUTPUTS:
05DA 1434 ;
05DA 1435 ;          THE TIMEOUT VALUE IS COMPUTED AND STORED IN DUE TIME, REGISTERS R3 AND
05DA 1436 ;          R4 ALONG WITH THE RETURN PC ARE SAVED IN THE FORK BLOCK, INTERRUPTS AND
05DA 1437 ;          TIMEOUT ARE ENABLED, AND A RETURN TO THE CALLER'S CALLER IS EXECUTED.
05DA 1438 ;-
05DA 1439
05DA 1440 IOC$WFIKPCH::          ;WAITFOR INTERRUPT/TIMEOUT AND KEEP CHANNEL
        6E 02 C0 05DA 1441          ADDL   #2,(SP)          ;CALCULATE OFFSET TO NORMAL RETURN
        10 A5 53 7D 05DD 1442          MOVQ   R3,UCB$L_FR3(R5)      ;SAVE REGISTERS R3 AND R4
                OC A5 8ED0 05E1 1443          POPL   UCB$L_FPC(R5)      ;SAVE INTERRUPT RETURN ADDRESS
        64 A5 03 A8 05E5 1444          BISW   #UCB$M_INT!UCB$M_TIM,UCB$W_STS(R5) ;ENABLE INTERRUPT AND TIMEOUT
                8E C1 05E9 1445          ADDL3  (SP)+,L^EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ;SET TIMEOUT TIME
00000000'EF 05EB
        6C A5          05F0
        0040 8F AA 05F2 1446          BICW   #UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR UNIT TIMED OUT
        64 A5          05F6
                05F8 1447          ENBINT          ;ENABLE INTERRUPTS
                05 05FB 1448          RSB

```

-1

```

05FC 1450          .SBTTL  WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
05FC 1451 ;+
05FC 1452 ; IOCS$WFIRLCH - WAITFOR INTERRUPT OR TIMEOUT AND RELEASE CHANNEL
05FC 1453 ;
05FC 1454 ; THIS ROUTINE IS CALLED TO SOFTWARE ENABLE INTERRUPTS AND TIMEOUT ON A DEVICE
05FC 1455 ; UNIT AND TO RELEASE THE CHANNEL. THIS ROUTINE CAN ONLY BE CALLED AT FORK LEVEL.
05FC 1456 ;
05FC 1457 ; INPUTS:
05FC 1458 ;
05FC 1459 ;         00(SP) = RETURN ADDRESS OF CALLER.
05FC 1460 ;         04(SP) = TIMEOUT VALUE IN SECONDS.
05FC 1461 ;         08(SP) = IPL TO LOWER TO AFTER SETTING WAIT.
05FC 1462 ;         12(SP) = RETURN ADDRESS OF CALLER'S CALLER.
05FC 1463 ;
05FC 1464 ;         R5 = UCB ADDRESS OF DEVICE UNIT.
05FC 1465 ;
05FC 1466 ; OUTPUTS:
05FC 1467 ;
05FC 1468 ;         THE TIMEOUT VALUE IS COMPUTED AND STORED IN DUE TIME, REGISTERS R3 AND
05FC 1469 ;         R4 ALONG WITH THE RETURN PC ARE SAVED IN THE FORK BLOCK, INTERRUPTS AND
05FC 1470 ;         TIMEOUT ARE ENABLED, THE CHANNEL IS RELEASED, AND A RETURN TO THE CALLER'S
05FC 1471 ;         CALLER IS EXECUTED.
05FC 1472 ; -
05FC 1473
05FC 1474 IOCS$WFIRLCH::                                ;WAITFOR INTERRUPT/TIMEOUT AND RELEASE CHANN
05FC 1475          ADDL    #2,(SP)                        ;CALCULATE OFFSET TO NORMAL RETURN
10 A5 53 7D 05FF 1476          MOVQ   R3,UCB$L_FR3(R5)    ;SAVE REGISTERS R3 AND R4
        OC A5 8ED0 0603 1477          POPL   UCB$L_FPC(R5)    ;SAVE INTERRUPT RETURN ADDRESS
64 A5 03 A8 0607 1478          BISW   #UCB$M_INT!UCB$M_TIM,UCB$W_STS(R5) ;ENABLE INTERRUPT AND TIMEOUT
        8E C1 060B 1479          ADDL3  (SP)+,L^EXE$GL_ABSTIM,UCB$L_DUETIM(R5) ;SET TIMEOUT TIME
00000000'EF          060D
        6C A5          0612
0040 8F AA          0614 1480          BICW   #UCB$M_TIMOUT,UCB$W_STS(R5) ;CLEAR UNIT TIMED OUT
        64 A5          0618
        061A 1481          ENBINT                                ;ENABLE INTERRUPTS
        FA63 31 061D 1482          BRW    IOCS$RELCHAN        ;RELEASE ALL CHANNELS AND RETURN TO CALLER
        0620 1483
        0620 1484

```

-1

```

0620 1486      .SBTTL  ALLOCATE SYSTEM PAGE TABLE
0620 1487      ;+
0620 1488      ; IOC$ALLOSPT - ALLOCATE SYSTEM PAGE TABLE
0620 1489      ;
0620 1490      ; THIS ROUTINE ALLOCATES SYSTEM PAGE TABLE (SPT) ENTRIES.
0620 1491      ;
0620 1492      ; INPUTS:
0620 1493      ;
0620 1494      ;      R1 = NUMBER OF SPT ENTRIES TO BE ALLOCATED
0620 1495      ;
0620 1496      ;      BOO$GL_SPTFREL = LOWEST FREE VPN
0620 1497      ;      BOO$GL_SPTFRELH = HIGHEST FREE VPN
0620 1498      ;
0620 1499      ;      IT IS ASSUMED THAT THE CALLER IS RUNNING AT IPL$_SYNCH.
0620 1500      ;
0620 1501      ; OUTPUTS:
0620 1502      ;
0620 1503      ;      R0 = SUCCESS INDICATION.
0620 1504      ;      R2 = STARTING PAGE NUMBER ALLOCATED (SVPN).
0620 1505      ;      R3 = ADDRESS OF BASE OF SYSTEM PAGE TABLE (MMG$GL_SPTBASE).
0620 1506      ;
0620 1507      ;      R1 IS PRESERVED ACROSS CALL.
0620 1508      ; -
0620 1509      IOC$ALLOSPT::
0620 1510      CLRL  R0      ;ALLOCATE SYSTEM PAGE TABLE
0620 1511      MOVL  L^BOO$GL_SPTFREL,R2      ;ASSUME FAILURE
0620 1512      ADDL3 R1,R2,R3      ;GET NEXT AVAILABLE SYSTEM VPN
0620 1513      Cmpl  R3,L^BOO$GL_SPTFRELH      ;COMPUTE NEXT WITH THIS ALLOCATION
0620 1514      BGEQU 10$      ;ARE THERE ENOUGH AVAILABLE?
0620 1515      MOVL  R3,L^BOO$GL_SPTFREL      ;BR IF NO
0620 1516      MOVL  L^MMG$GL_SPTBASE,R3      ;MARK THE ENTRIES ALLOCATED
0620 1517      MOVL  L^MMG$GL_SPTBASE,R3      ;GET ADDR OF BASE OF SPT
0620 1518      INCL  R0      ;SET SUCCESS
0620 1519      RSB      ;

```



```

;RAS0300      0647 .15      <RESR0>, -      ;Result R0
;RAS0300      0647 .16      <RESR1>, -      ;Result R1
;RAS0300      0647 .17      <SCRLEN,0> -      ;amount of working storage
;RAS0300      0647 .18      <RESR2>, -      ;saved R2
;RAS0300      0647 .19      <RESR3>, -      ;saved R3
;RAS0300      0647 .20      <RESR4>, -      ;saved R4
;RAS0300      0647 .21      >
0000          BINNUM:
0008          RESR0:
000C          RESR1:
0010          SCRLEN:
0010          RESR2:
0014          RESR3:
0018          RESR4:
;RAS0300      0647 .22
;RAS0300      0647 .23      IOC$CVT_DEVNAM::      ;Convert device name and unit
;RAS0300      0647 .24
;RAS0300      00FC 8F  BB  0647 .25      PUSHR  #^M<R2,R3,R4,R5,R6,R7> ;Save registers
;RAS0300      064B .26 ;
;RAS0300      064B .27 ; Push a quadword onto the stack. The quadword will land
;RAS0300      064B .28 ; on the stack so that when the POPR at the end of the routine
;RAS0300      064B .29 ; is executed, R0 will contain the routine value, and R1 will
;RAS0300      064B .30 ; contain the length of the formatted device name.
;RAS0300      064B .31 ;
;RAS0300      7E  01  7D  064B .32      MOVQ   #SS$_NORMAL,-(SP)      ;Put a 1 and a 0 on the stack
;RAS0300      7E  7C  064E .33      CLRQ   -(SP)      ;Init binary number working area.
;RAS0300      0650 .34      ASSUME SCRLEN EQ 16
;RAS0300      57  5E  D0  0650 .35      MOVL   SP, R7      ;Setup result R0 and R1 pointer in R7.
;RAS0300      0653 .36 ;
;RAS0300      0653 .37 ; Precede the device name with a "_" (underscore character) to
;RAS0300      0653 .38 ; indicate that this is a physical device name.
;RAS0300      0653 .39 ;
;RAS0300      53  5F 8F  9A 0653 .40      MOVZBL #^A/_/,R3      ;Put underscore character in R3
;RAS0300      00B4 30 0657 .41      BSBW   PUTCHAR      ;Put it in the output buffer
;RAS0300      065A .42 ;
;RAS0300      065A .43 ; Check for a possible nodename. If it exists, determine which format
;RAS0300      065A .44 ; of name was requested by the caller.
;RAS0300      065A .45 ;
;RAS0300      56  28 A5  D0 065A .46      MOVL   UCB$_DDB(R5),R6      ;Get DDB address
;RAS0300      52  34 A6  D0 065E .47      MOVL   DDB$_SB(R6),R2      ;Get System Block address
;RAS0300      5D  13 0662 .48      BEQL   LOCAL_NAME      ;None, leave
;RAS0300      09  E1 0664 .49      BBC    #DEV$V_NNM,-      ;Branch if nodename not wanted
;RAS0300      58  3C A5 0666 .50      UCB$_DEVCHAR2(R5),LOCAL_NAME
;RAS0300      0669 .51      CASE   R4, -      ;Dispatch on type of output requested:
;RAS0300      0669 .52      limit=#-1, displist=<-
;RAS0300      0669 .53      DISPLAY_NAME, -      ; -1 ==> node$dev: for disks, else dev:
;ROW0244      0669 .54      FULL_NAME, -      ; 0 ==> $alloc1s$dev: or node$dev:
;RAS0300      0669 .55      ALLOC_NAME, -      ; 1 ==> $alloc1s$dev: or node$dev:
;RAS0300      0669 .56      LOCAL_NAME, -      ; 2 ==> just dev:
;RAS0300      0669 .57      SECONDARY_NAME, -      ; 3 ==> secondary path
;RAS0300      0669 .58      DISPLAY_NAME -      ; 4 ==> same as -1 sans unit number
;RAS0300      0669 .59      >
;RAS0300      5B  11 067B .60      BRB    EXDVNM      ; All others are NOPs.
;RAS0300      067D .61
;ROW0244      067D .62      FULL_NAME:
;ROW0244      33 38 A5  0E  E1 067D .63      BBC    #DEV$V_FOD, -      ;A file oriented device?
;ROW0244      0682 .64      UCB$_DEVCHAR(R5), -

```



```

;ROW0244          0682 .65          ADD_NODE          ;Branch if not file oriented device.
;ROW0244          0682 .66
;RAS0300          0682 .67 ALLOC_NAME:
;RAS0300          0682 .68
;RAS0300          67   3C A6   9A 0682 .69          MOVZBL  DDB$_ALLOCLS(R6), - ;Setup allocation class value
;RAS0300          0686 .70          BINNUM(R7)          ; for conversion.
;RAS0300          2D   13 0686 .71          BEQL     ADD_NODE          ;If none return node+device name.
;RAS0300          0080 30 0688 .72          BSBW     PUTDOLLAR        ;Prepend allocation class with a '$'
;RAS0300          58   10 068B .73          BSBB     PUTNUM          ;Convert allocation class number to
;RAS0300          068D .74          ;ASCII and put it in the buffer
;RAS0300          30   11 068D .75          BRB     ADD_DOLLAR       ;Append dollar sign to alloc. class
;RAS0300          068F .76          ; and add device name to buffer.
;RAS0300          068F .77
;RAS0300          068F .78 SECONDARY_NAME:
;RAS0300          04   E1 068F .79          BBC     #DEV$V_2P,-      ;Branch if device not dual-pathed.
;RAS0300          3C   A5 0691 .80          UCB$_DEVCHAR2(R5),-    ; (I.E. there is no secondary path to
;RAS0300          4C   C5 0693 .81          NO_SECONDARY          ; return.)
;RAS0300          56   00A0 C5 D0 0694 .82          MOVL    UCB$_DP_DDB(R5),R6 ;Get secondary DDB.
;RAS0300          45   13 0699 .83          BEQL    NO_SECONDARY    ;Branch to no sec. path if none.
;RAS0300          52   34 A6 D0 069B .84          MOVL    DDB$_SB(R6),R2  ;Get alternate SB.
;RAS0300          069F .85
;RAS0300          069F .86 DISPLAY_NAME:
;RAS0300          52   D1 069F .87          CMPL    R2,#SCS$GA_LOCALSB ;Is it the perm local system block?
;RAS0300          00000000'8F 06A1 .88
;RAS0300          0D   12 06A6 .88          BNEQ    ADD_NODE          ;Return node+devnam for non-local devs.
;RAS0300          06A8 .89          IFNOCLSTR LOCAL_NAME   ;Return devnam if not part of a cluster.
;ROW0244          OC 38 A5 0E E1 06B0 .90          BBC     #DEV$V_FOD, -   ;A file oriented device?
;ROW0244          06B5 .91          UCB$_DEVCHAR(R5), -
;ROW0244          06B5 .92          LOCAL_NAME
;ROW0244          06B5 .93          ;Branch if not a file oriented device.
;ROW0244          06B5 .94          ;Its a local disk in a cluster: return
;ROW0244          06B5 .95          ;node+device name format.
;ROW0244          06B5 .96          ; Return node name plus device name. Copy node name to buffer and
;ROW0244          06B5 .97          ; suffix with a "$" before moving in rest of device name.
;ROW0244          06B5 .98          ;
;ROW0244          06B5 .99 ADD_NODE:
;ROW0244          52   44 A2 9E 06B5 .100         MOVAB   SB$_NODENAME(R2),R2 ;Point to name field
;ROW0239          62   95 06B9 .101         TSTB   (R2)              ;Is the node name null?
;ROW0239          04   13 06BB .102         BEQL    LOCAL_NAME      ;Skip inserting node name, if its null.
;RAS0300          3E   10 06BD .103         BSBB   PUTASCIC         ;Copy counted ASCII str. to output buf.
;RAS0300          06BF .104 ADD_DOLLAR:
;RAS0300          4A   10 06BF .105         BSBB   PUTDOLLAR        ;Append dollar sign to node name
;RAS0300          06C1 .106          ;
;RAS0300          06C1 .107          ; Copy device name to buffer.
;RAS0300          06C1 .108          ;
;RAS0300          06C1 .109 LOCAL_NAME:
;RAS0300          52   14 A6 9E 06C1 .110         MOVAB   DDB$_NAME(R6),R2 ;Get address of ASCII device name.
;RAS0300          36   10 06C5 .111         BSBB   PUTASCIC         ;Copy counted ASCII str. to output buf.
;RAS0300          04   18 A7 B1 06C7 .112         CMPW   RESR4(R7),#4     ;Do we want the unit number?
;RAS0300          0B   13 06CB .113         BEQL    EXDVNM          ;Nope
;RAS0300          67   54 A5 3C 06CD .114         MOVZWL  UCB$_W_UNIT(R5), - ;Setup device unit number for
;RAS0300          06D1 .115         BINNUM(R7)            ; conversion to ASCII.
;RAS0300          12   10 06D1 .116         BSBB   PUTNUM          ;Convert unit number to ASCII.
;RAS0300          06D3 .117          ;
;RAS0300          06D3 .118          ; Terminate the device name with a ":" (colon).
;RAS0300          06D3 .119          ;
;RAS0300          53   3A 9A 06D3 .120         MOVZBL  #^A/:/,R3      ;Put a ":" in R3

```

```

;RAS0300      36   10  06D6  .121      BSBB   PUTCHAR      ;Put the ":" in output buffer
;RAS0300      06D8  .122 ;
;RAS0300      06D8  .123 ; Clean up the stack and exit. The stack has been set up so that
;RAS0300      06D8  .124 ; the proper values will be stored in R0 and R1 by the POPR.
;RAS0300      06D8  .125 ;
;RAS0300      5E   08  C0  06D8  .126  EXDVNM: ADDL   #RESR0,SP      ;Remove everything upto result R0
;RAS0300      06DB  .127      ;from the stack
;RAS0300      00FF  8F  BA  06DB  .128      POPR   #^M<R0,R1,R2,R3,R4,R5,R6,R7> ;Restore registers
;RAS0300      05    05  06DF  .129      RSB           ;Return
;RAS0300      06E0  .130
;RAS0300      06E0  .131 ;
;RAS0300      06E0  .132 ; Come here when the secondary device name was requested but none exists.
;RAS0300      06E0  .133 ;
;RAS0300      06E0  .134  NO_SECONDARY:
;RAS0300      0C   A7  D4  06E0  .135      CLRL   RESR1(R7)      ;Clear count of characters
;RAS0300      F3    11  06E3  .136      BRB     EXDVNM      ;and return.
;RAS0300      06E5  .137
;RAS0300      06E5  .138
;RAS0300      06E5  .139 ;++
;RAS0300      06E5  .140 ; The following code is a local subroutine to convert binary to ASCII and
;RAS0300      06E5  .141 ; put the ASCII equivalent in the output name buffer.
;RAS0300      06E5  .142 ;
;RAS0300      06E5  .143 ; Inputs:
;RAS0300      06E5  .144 ;
;RAS0300      06E5  .145 ;         BINNUM(R7)         binary number to be converted (a quadword with high
;RAS0300      06E5  .146 ;                               longword zeroed
;RAS0300      06E5  .147 ;
;RAS0300      06E5  .148 ; Outputs:
;RAS0300      06E5  .149 ; The number at BINNUM(R7) is converted to ASCII and stored in the
;RAS0300      06E5  .150 ; device name buffer.
;RAS0300      06E5  .151 ;--
;RAS0300      06E5  .152  PUTNUM:
;RAS0300      53   01  8E  06E5  .153      MNEGB  #1, R3           ;Get end-of-number marker.
;RAS0300      7E   53  90  06E8  .154  10$:  MOVB   R3, -(SP)      ;Move digit/marker to scratch.
;RAS0300      67   67  0A  06EB  .155      EDIV   #10, BINNUM(R7), - ;Divide number by 10, overwrite number
;RAS0300      53    06EF
;RAS0300      06F0  .156      BINNUM(R7), R3      ;with quotient, put remainder in R3.
;RAS0300      F6    12  06F0  .157      BNEQ   10$          ;If quotient not zero, go save this
;RAS0300      06F2  .158      ; digit and get the next one.
;RAS0300      06F2  .159 ;
;RAS0300      06F2  .160 ; Get digits -- most significant first (then saved ones), convert them to
;RAS0300      06F2  .161 ; ASCII, and put them in the output buffer
;RAS0300      06F2  .162 ;
;RAS0300      53   30  80  06F2  .163  50$:  ADDB   #^A/0/, R3      ;Convert binary digit to ASCII
;RAS0300      17   10  06F5  .164      BSBB   PUTCHAR      ;Copy digit to output buffer
;RAS0300      53   8E  90  06F7  .165      MOVB   (SP)+, R3     ;Get another digit
;RAS0300      F6    18  06FA  .166      BGEQ   50$          ;Branch if the end
;RAS0300      05    05  06FC  .167      RSB           ;
;RAS0300      06FD  .168
;RAS0300      06FD  .169 ;++
;RAS0300      06FD  .170 ; The following code is a local subroutine to copy a counted ASCII string
;RAS0300      06FD  .171 ; to the output name buffer.
;RAS0300      06FD  .172 ;
;RAS0300      06FD  .173 ; Inputs:
;RAS0300      06FD  .174 ;
;RAS0300      06FD  .175 ;         R2           Beginning address of a counted ASCII string
;RAS0300      06FD  .176 ;

```

```

;RAS0300          06FD .177 ; Outputs:
;RAS0300          06FD .178 ;       The counted ASCII string pointed to by R2 is copied to the device
;RAS0300          06FD .179 ;       name buffer.
;RAS0300          06FD .180 ;--
;RAS0300          06FD .181 PUTASCIC:
;RAS0300          54  82  9A 06FD .182       MOVZBL (R2)+, R4           ;Get counted string length.
;RAS0300          08  13 0700 .183       BEQL 90$                   ;If no characters, leave.
;RAS0300          53  82  90 0702 .184 5$:   MOVB (R2)+, R3           ;Move one byte to output buffer.
;RAS0300          07  10 0705 .185       BSBB PUTCHAR              ;Put the character in the output buffer.
;RAS0300          F8 54  F5 0707 .186       SOBGTR R4, 5$           ;Branch if more to copy.
;RAS0300          05 070A .187 90$:   RSB                          ;All done, return.
-59              070B 1603
              070B 1604 ;++
              070B 1605 ;
              070B 1606 ; The following code is a local subroutine to place a given
              070B 1607 ; byte in the output buffer. A count is kept of all characters
              070B 1608 ; placed in the output buffer. If the output buffer is full,
;KPL0001          070B .1 ; the byte is not copied, the count is not increased, and the
;KPL0001          070B .2 ; return status for IOC$CVT_DEVNAM is changed to SS$_BUFFEROVF
;KPL0001          070B .3 ; (an alternate success status).
;KPL0001          070B .4 ;
;KPL0001          070B .5 ; Inputs:
;KPL0001          070B .6 ;       R0       Count of unstored character slots remaining in output buffer
;KPL0001          070B .7 ;       R1       Address of next unused character slot in output buffer
;KPL0001          070B .8 ;       R3       Character to be placed in the buffer
;KPL0001          070B .9 ;
;KPL0001          070B .10 ; Implicit inputs:
;KPL0001          070B .11 ;       RESR0(R7)   longword holding final IOC$CVT_DEVNAM status
;KPL0001          070B .12 ;       RESR1(R7)   longword holding final IOC$CVT_DEVNAM count of
;KPL0001          070B .13 ;               characters stored in the buffer (to be
;KPL0001          070B .14 ;               returned in R1
;KPL0001          070B .15 ;
;KPL0001          070B .16 ; Outputs:
;KPL0001          070B .17 ;       None.
;KPL0001          070B .18 ;
;KPL0001          070B .19 ; Implicit outputs:
;KPL0001          070B .20 ;       If R0 >= zero:
;KPL0001          070B .21 ;               R0           <== R0 - 1
;KPL0001          070B .22 ;               (R1)         <== R3
;KPL0001          070B .23 ;               R1           <== R1 + 1
;KPL0001          070B .24 ;               RESR1(R7)     <== RESR1(R7) + 1
;KPL0001          070B .25 ;       otherwise:
;KPL0001          070B .26 ;               RESR0(R7)     <== SS$_BUFFEROVF
;KPL0001          070B .27 ;++
;KPL0001          070B .28 ; PUTDOLLAR is an internal routine which is the equivalent of:
;KPL0001          070B .29 ;
;KPL0001          070B .30 ;       MOVB   #^A/$/, R3
;KPL0001          070B .31 ;       BSBB   PUTCHAR
;KPL0001          070B .32 ;--
;KPL0001          070B .33 PUTDOLLAR:
;KPL0001          53  24  90 070B .34   MOVB   #^A/$/, R3           ;Setup to put "$" in output buffer.
;KPL0001          070E .35 PUTCHAR:
;KPL0001          50  D7 070E .36   DECL   R0                   ;Decrease characters remaining count.
;KPL0001          07  19 0710 .37   BLSS   90$                   ;Branch if no more characters remaining.
;KPL0001          81  53  90 0712 .38   MOVB   R3, (R1)+         ;Copy character to output buffer
;KPL0001          OC A7 D6 0715 .39   INCL   RESR1(R7)         ;Count characters stored
;KPL0001          05 0718 .40   RSB                          ;Return

```

;KPL0001			0719	.41					
;KPL0001	0601 8F	3C	0719	.42 90\$:	MOVZWL	#SS\$_BUFFEROVF, -			;Set buffer overflow status
	08 A7		071D						
;KPL0001			071F	.43		RESR0(R7)			
;KPL0001		05	071F	.44	RSB				

-14

```

0720 1623      .SBTTL  BROADCAST TO A TERMINAL
0720 1624 ;++
0720 1625 ; IOC$BROADCAST
0720 1626 ;
0720 1627 ;      This routine will allow driver fork processes to broadcast a
0720 1628 ;      given message to given terminal.  The broadcast request is
0720 1629 ;      dispatched to the proper terminal and control returns immediately
0720 1630 ;      to the caller.  Some time later the broadcast will complete, and
0720 1631 ;      at that time all the necessary post-processing will be done.
0720 1632 ;
0720 1633 ;      This routine does not implement all the features of the $BRDCST system
0720 1634 ;      service, but only the bare minimum necessary to send a message to a
0720 1635 ;      single terminal.  For more information about the terminal broadcast
0720 1636 ;      mechanism, see the module SYSBRDCST.
0720 1637 ;
0720 1638 ; Input:
0720 1639 ;
0720 1640 ;      R1 = Message length
0720 1641 ;      R2 = Message address
0720 1642 ;      R5 = Address of target terminal's UCB
0720 1643 ;
0720 1644 ; Implicit input:
0720 1645 ;
0720 1646 ;      IPL$_ASTDEL  <=  CURRENT_IPL  <=  UCB$_FIPL(R5)
0720 1647 ;
0720 1648 ; Output:
0720 1649 ;
0720 1650 ;      None.  The contents of R1 .. R5 are preserved across the call.
0720 1651 ;
0720 1652 ; Routine value:
0720 1653 ;
0720 1654 ;      SS$_NORMAL      - The broadcast completed successfully.
0720 1655 ;      SS$_INSFMEM     - Insufficient dynamic nonpaged pool for the request.
0720 1656 ;      SS$_DEVOFFLINE  - The target terminal has rejected the request.
0720 1657 ;      SS$_ILLIOFUNC   - The specified UCB does not belong to a terminal.
0720 1658 ;                  (Therefore a BROADCAST is an illegal I/O function.)
0720 1659 ;--
0720 1660
00000000 0720 1661 SAVED_R0 = 0 ;.
00000004 0720 1662 SAVED_R1 = 4 ;.
00000008 0720 1663 SAVED_R2 = 8 ;.
0000000C 0720 1664 SAVED_R3 = 12 ;. Symbolic offsets to saved registers
00000010 0720 1665 SAVED_R4 = 16 ;.
00000014 0720 1666 SAVED_R5 = 20 ;.
0720 1667
0720 1668 IOC$BROADCAST:: ; Broadcast to a terminal
50 00F4 8F 3C 0720 1669 MOVZWL #SS$_ILLIOFUNC,R0 ; Assume device not a terminal
      02 E1 0725 1670 BBC #DEV$V TRM,- ; Branch if not a terminal
56 38 A5 0727 1671 UCB$_L_DEVCHAR(R5),14$ ;
      3F BB 072A 1672 PUSHR #AM<R0,R1,R2,R3,R4,R5> ; Save R0 .. R5
      51 30 C0 072C .1 ADDL2 #TTY$_WB_LENGTH,R1 ; Calculate the total pool required
6E 0124 8F 3C 072F 1674 MOVZWL #SS$_INSFMEM,SAVED_R0(SP); Assume allocation failure
      F8C9' 30 0734 1675 BSBW EXE$ALONONPAGED ; Allocate the pool
      44 50 E9 0737 1676 BLBC R0,13$ ; Exit if error
      073A 1677 ;
      073A 1678 ; Fill in the Terminal Write Packet (TWP).
      073A 1679 ; Note that EXE$ALONONPAGED the pool size

```

;TCM0003  
-1

```

                                073A 1680      ; in R1 and the pool address in R2.
                                073A 1681      ;
08 A2  51  B0 073A 1682      MOVW  R1,TTY$WB_SIZE(R2)      ; Set TWP size
                                30  90 073E 1683      MOVW  #DYN$C_TWP,-            ; Set TWP structure type
                                OA A2 0740 1684      TTY$B_WB_TYPE(R2)           ;
                                06  90 0742 1685      MOVW  #IPL$_QUEUEEAST,-      ; Set the TWP fork IPL (for later use)
                                OB A2 0744 1686      TTY$B_WB_FIPL(R2)           ;
10 A2  01  D0 0746 1687      MOVL  #1,TTY$L_WB_FR3(R2)     ; Request refresh of read prompt
                                30 A2 074A 1688      MOVAB TTY$L_WB_DATA(R2),-     ; Set address of message start
                                1C A2 074D 1689      TTY$L_WB_NEXT(R2)          ;
                                04 AE C1 074F 1690      ADDL3 SAVED_R1(SP),-        ; Set address of message end
                                1C A2 0752 1691      TTY$L_WB_NEXT(R2),-        ;
                                20 A2 0754 1692      TTY$L_WB_END(R2)           ;
                                8B AF 9E 0756 1693      MOVAB BAEND_BROADCAST,-     ; Set callback address
                                2C A2 0759 1694      TTY$L_WB_RETADDR(R2)       ;
                                24 A2 D4 075B 1695      CLRL  TTY$L_WB_IRP(R2)      ; Clear pointer to associated IRP
                                52  DD 075E 1696      PUSHL R2                    ; Save TWP address
                                08 AE 28 0760 1697      MOVW3 4+SAVED_R1(SP),-      ; Copy the message text to the TWP
                                0C BE 0763 1698      @4+SAVED_R2(SP),-         ; (note the stack depth changed)
                                30 A2 0765 1699      TTY$L_WB_DATA(R2)         ;
                                0767 1700      ;
                                0767 1701      ; Queue the broadcast request to the terminal.
                                0767 1702      ; The disposition of the broadcast request will be determined
                                0767 1703      ; by the contents of TTY$L_WB_END. Note that if the request is
                                0767 1704      ; accepted by a remote terminal, or is rejected outright, the
                                0767 1705      ; TWP is no longer needed, and may be deallocated. The TTY$L_WB_END
                                0767 1706      ; field of the TWP will contain one of the following values:
                                0767 1707      ;
                                0767 1708      ; System address: request accepted by TTDRIVER
                                0767 1709      ; 1: request accepted by RTTDRIVER
                                0767 1710      ; 2: request rejected
                                0767 1711      ;
53  53  6E  D0 0767 1712      MOVL  (SP),R3                ; Put TWP address in R3
55  18 AE D0 076A 1713      MOVL  4+SAVED_R5(SP),R5      ; Restore UCB address
                                F88F' 30 076E 1714      BSBW  EXE$ALTQUEPKT         ; Queue the request to the terminal
                                50 8ED0 0771 1715      POPL  R0                     ; Remove TWP address from the stack
6E  01  3C 0774 1716      MOVZWL #SS$NORMAL,SAVED_R0(SP); Assume success
20  A0  D5 0777 1717      TSTL  TTY$L_WB_END(R0)      ; Check for rejection
                                05  13 077A 1718      BEQL  69$                    ; Branch if request rejected
                                08  14 077C 1719      BGTR  80$                    ; Branch if remote terminal accepted
                                3F  BA 077E 1720 13$:  POPR  #^M<R0,R1,R2,R3,R4,R5> ; Restore the registers
                                05  0780 1721 14$:  RSB                     ; Return
0084 8F  3C 0781 1722 69$:  MOVZWL #SS$DEVOFFLINE,-      ; Set broadcast rejection status
                                6E  0785 1723      SAVED_R0(SP)                ;
                                F877' 30 0786 1724 80$:  BSBW  COM$DRVDEALMEM      ; Deallocate the TWP
                                F3  11 0789 1725      BRB  13$                     ; Take common exit path
                                078B 1726      ;
                                078B 1727      ;
                                078B 1728      ; The following code performs all of the necessary broadcast post-processing.
                                078B 1729      ; This entry point is FORKed to at IPL IPL$_QUEUEEAST from the terminal driver.
                                078B 1730      ; The fork block is the TWP.
                                078B 1731      ;
                                078B 1732      END_BROADCAST:           ; Post-processor for broadcast requests
50  55  D0 078B 1733      MOVL  R5,R0                  ; Copy TWP address
                                F86F' 31 078E 1734      BRW  EXE$DEANONPAGED        ; Deallocate the TWP and return

```

```

;ACG0399          0791      .2      .SBTTL  SCAN THE I/O DATA BASE
;ACG0399          0791      .3      ;+
;ACG0399          0791      .4      ; IOC$SCAN_IODB - Scan the I/O data base and return next block.
;ACG0399          0791      .5      ;
;ACG0399          0791      .6      ; This routine is called to scan the device lists in the IO data base and
;ACG0399          0791      .7      ; return a pointer to the next block in the list. Context is kept in R11
;ACG0399          0791      .8      ; and by using back pointers.
;ACG0399          0791      .9      ;
;ACG0399          0791     .10      ; Inputs:
;ACG0399          0791     .11      ;
;ACG0399          0791     .12      ; The I/O data base is locked for read access. This means that the caller
;ACG0399          0791     .13      ; owns the I/O data base mutex and/or is at IPL SYNCH or higher.
;ACG0399          0791     .14      ;
;ACG0399          0791     .15      ; R11 = 0 implies first call
;ACG0399          0791     .16      ; R11 <> 0 indicates that R11 is pointer to current DDB
;ACG0399          0791     .17      ; R10 = 0 implies end of UCB chain
;ACG0399          0791     .18      ; R10 <> 0 indicates that R10 is pointer to current UCB
;ACG0399          0791     .19      ;
;ACG0399          0791     .20      ; Outputs:
;ACG0399          0791     .21      ;
;ACG0399          0791     .22      ; R0 = Success status.
;ACG0399          0791     .23      ; R10 = Pointer to UCB
;ACG0399          0791     .24      ; R11 = Pointer to DDB
;ACG0399          0791     .25      ;
;ACG0399          0791     .26      ; All other registers preserved.
;ACG0399          0791     .27      ;
;ACG0399          0791     .28      ; -
;ACG0399          0791     .29      ;
;ACG0399          0791     .30      IOC$SCAN_IODB::
;ACG0399          0791     .31      ;
;ACG0399          50      01      D0      0791     .32      MOVL      #1,R0          ; Success
;ACG0399          5B      D5      0794     .33      TSTL      R11          ; Initial condition?
;ACG0399          2C      13      0796     .34      BEQL      50$         ; Yes
;ACG0399          5A      D5      0798     .35      TSTL      R10          ; End of chain?
;ACG0399          07      13      079A     .36      BEQL      10$         ; Yes
;ACG0399          5A      30      AA      D0      079C     .37      MOVL      UCB$L_LINK(R10),R10 ; Get next UCB
;ACG0399          01      13      07A0     .38      BEQL      10$         ; None
;ACG0399          05      05      07A2     .39      RSB
;ACG0399          07A3     .40      ;
;ACG0399          6B      D5      07A3     .41      10$: TSTL      DDB$L_LINK(R11) ; At end of DDB chain?
;ACG0399          0A      13      07A5     .42      BEQL      30$         ; Yes
;ACG0399          5B      6B      D0      07A7     .43      MOVL      DDB$L_LINK(R11),R11 ; No, get next one
;ACG0399          5A      04      AB      D0      07AA     .44      20$: MOVL      DDB$L_UCB(R11),R10 ; Pick up first UCB
;ACG0399          F3      13      07AE     .45      BEQL      10$         ; None, get next DDB
;ACG0399          05      05      07B0     .46      RSB
;ACG0399          07B1     .47      ;
;ACG0399          5B      34      AB      D0      07B1     .48      30$: MOVL      DDB$L_SB(R11),R11 ; Get back to parent system block
;ACG0399          5B      6B      D0      07B5     .49      40$: MOVL      SB$L_FLINK(R11),R11 ; Get next system block
;ACG0399          5B      D1      07B8     .50      CMPL      R11,#SCS$GQ_CONFIG ; End of chain?
;ACG0399          00000000'8F      07BA     .51      ;
;ACG0399          0A      12      07BF     .51      BNEQ      60$         ; No
;ACG0399          50      D7      07C1     .52      DECL      R0
;ACG0399          05      05      07C3     .53      RSB
;ACG0399          07C4     .54      ;
;ACG0399          00000000'9F      D0      07C4     .55      50$: MOVL      @#SCS$GQ_CONFIG,R11 ; Pick up first system block
;ACG0399          5B      07CA     .56      ;
;ACG0399          54      AB      D5      07CB     .56      60$: TSTL      SB$L_DDB(R11) ; Is there a DDB chain?

```

;ACG0399		E5	13	07CE	.57	BEQL	40\$	; No, go try next SB
;ACG0399	5B	54	AB	D0	07D0	.58	MOVL	SB\$L_DDB(R11),R11 ; Yes, get the first DDB
;ACG0399		D4	.11	07D4	.59	BRB	20\$	
;ACG0399				07D6	.60			



```

;ACG0399      07D6 .62      .SBTTL  SCAN THE I/O DATA BASE BOTH PRIMARY & SECONDARY PATHS
;ACG0399      07D6 .63 ;++
;ACG0399      07D6 .64 ; IOC$SCAN_IODB_2P
;ACG0399      07D6 .65 ;
;ACG0399      07D6 .66 ; This routine is called to scan the device lists in the IO data base and
;ACG0399      07D6 .67 ; return a pointer to the next block in the list. Context is kept in R10
;ACG0399      07D6 .68 ; and R11 and by using back pointers.
;ACG0399      07D6 .69 ;
;ACG0399      07D6 .70 ; SCAN_IODB_2P differs from SCAN_IODB in that it will scan both the primary
;ACG0399      07D6 .71 ; and secondary UCB chain for each DDB. This means that if a device is
;ACG0399      07D6 .72 ; dual-pathed, SCAN_IODB_2P will return the address of its UCB twice, once in
;ACG0399      07D6 .73 ; the context of the primary controller and once in the context of the
;ACG0399      07D6 .74 ; secondary.
;ACG0399      07D6 .75 ;
;ACG0399      07D6 .76 ; Inputs and Outputs are identical to IOC$SCAN_IODB.
;ACG0399      07D6 .77 ;--
;ACG0399      07D6 .78
;ACG0399      07D6 .79 IOC$SCAN_IODB_2P::
;ACG0399      07D6 .80
;ACG0399      50  01  D0 07D6 .81      MOVL  #1,R0          ; Success
;ACG0399      5B  05  D5 07D9 .82      TSTL  R11          ; Initial condition?
;ACG0399      41  13  07DB .83      BEQL  60$         ; Yes
;ACG0399      5A  05  07DD .84      TSTL  R10          ; Caller signalled end of chain?
;ACG0399      1C  13  07DF .85      BEQL  30$         ; Yes, done with this DDB
;ACG0399      07E1 .86 ;
;ACG0399      07E1 .87 ; At this point we must decide if we're following the primary or secondary
;ACG0399      07E1 .88 ; chain of UCBs on this DDB.
;ACG0399      07E1 .89 ;
;ACG0399      5B  28  AA  D1 07E1 .90      Cmpl  UCB$_DDB(R10),R11 ; Are we traversing the primary chain?
;ACG0399      07  12  07E5 .91      BNEQ  10$         ; Branch if we're following secondary
;ACG0399      5A  30  AA  D0 07E7 .92      MOVL  UCB$_LINK(R10),R10 ; Get next UCB on primary chain
;ACG0399      09  13  07EB .93      BEQL  20$         ; Branch if none to try secondary chain
;ACG0399      05  05  07ED .94      RSB          ; Else return UCB address to caller
;ACG0399      07EE .95 ;
;ACG0399      07EE .96 ; Get next UCB on secondary chain.
;ACG0399      07EE .97 ;
;ACG0399      5A  00A4 CA  D0 07EE .98 10$: MOVL  UCB$_DP_LINK(R10),R10 ; Get next UCB on secondary chain
;ACG0399      08  13  07F3 .99      BEQL  30$         ; Branch if none left
;ACG0399      05  05  07F5 .100     RSB          ; Else return UCB address to caller
;ACG0399      07F6 .101 ;
;ACG0399      07F6 .102 ; No UCBs left on primary chain; traverse secondary chain if present.
;ACG0399      07F6 .103 ;
;ACG0399      5A  40  AB  D0 07F6 .104 20$: MOVL  DDB$_DP_UCB(R11),R10 ; Get first UCB on secondary chain
;ACG0399      01  13  07FA .105     BEQL  30$         ; Branch if none to try next DDB
;ACG0399      05  05  07FC .106     RSB          ; Else return UCB address to caller
;ACG0399      07FD .107 ;
;ACG0399      07FD .108 ; Step to next DDB.
;ACG0399      07FD .109 ;
;ACG0399      6B  D5  07FD .110 30$: TSTL  DDB$_LINK(R11) ; At end of DDB chain?
;ACG0399      0A  13  07FF .111     BEQL  40$         ; Yes, try next system block
;ACG0399      5B  6B  D0 0801 .112     MOVL  DDB$_LINK(R11),R11 ; No, get next one
;ACG0399      5A  04  AB  D0 0804 .113 35$: MOVL  DDB$_UCB(R11),R10 ; Pick up first UCB on primary chain
;ACG0399      EC  13  0808 .114     BEQL  20$         ; None, try for UCB on secondary chain
;ACG0399      05  05  080A .115     RSB          ; Else return UCB address to caller
;ACG0399      080B .116
;ACG0399      080B .117 ;
;ACG0399      080B .118 ; Step to next system block.

```

```

;ACG0399          080B .119 ;
;ACG0399          5B 34 AB D0 080B .120 40$:   MOVL   DDB$_SB(R11),R11       ; Get back to parent system block
;ACG0399          5B 6B D0 080F .121 50$:   MOVL   SB$_FLINK(R11),R11      ; Get next system block
;ACG0399          5B D1 0812 .122          CMPL   R11,#SCS$GQ_CONFIG    ; End of chain?
00000000'8F      0814
;ACG0399          0A 12 0819 .123          BNEQ   70$                ; No
;ACG0399          50 D7 081B .124          DECL   R0                  ; Signal end of IO scan
;ACG0399          05 081D .125          RSB
;ACG0399          081E .126
;ACG0399          00000000'9F D0 081E .127 60$:   MOVL   @#SCS$GQ_CONFIG,R11    ; Pick up first system block
;ACG0399          5B 0824
;ACG0399          54 AB D5 0825 .128 70$:   TSTL   SB$_DDB(R11)         ; Is there a DDB chain?
;ACG0399          E5 13 0828 .129          BEQL   50$                ; No, go try next SB
;ACG0399          5B 54 AB D0 082A .130          MOVL   SB$_DDB(R11),R11      ; Yes, get the first DDB
;ACG0399          D4 11 082E .131          BRB    35$                ; Try for UCB on primary chain

```

```

;ACG0399      0830 .133      .SBTTL IOC$CTRLINIT - Call driver controller init. routine
;ACG0399      0830 .134 ;++
;ACG0399      0830 .135 ; FUNCTIONAL DESCRIPTION:
;ACG0399      0830 .136 ;
;ACG0399      0830 .137 ;         For UNIBUS devices, the device CSR is tested for existence. If this
;ACG0399      0830 .138 ;         test fails, a no routine call occurs and failure status is returned in
;ACG0399      0830 .139 ;         R0. Input values for a device driver's controller initialization
;ACG0399      0830 .140 ;         routine are loaded into the proper registers, the routine starting
;ACG0399      0830 .141 ;         address is located, and if a routine exists, it is called.
;ACG0399      0830 .142 ;
;ACG0399      0830 .143 ; INPUTS:
;ACG0399      0830 .144 ;         R1      CSR address to use if IDB contains zero
;ACG0399      0830 .145 ;         R8      CRB address (primary)
;ACG0399      0830 .146 ;         R11     DDB address
;ACG0399      0830 .147 ;
;ACG0399      0830 .148 ; OUTPUTS:
;ACG0399      0830 .149 ;         R0      Status (success, or failure ==> UNIBUS CSR non-existent)
;ACG0399      0830 .150 ;         R1-R6   Destroyed
;ACG0399      0830 .151 ;--
;ACG0399      0830 .152 ;
;ACG0399      0830 .153 ;++
;ACG0399      0830 .154 ; Controller initialization routine parameters:
;ACG0399      0830 .155 ;
;ACG0399      0830 .156 ; INPUTS:
;ACG0399      0830 .157 ;         R4      CSR address (for UNIBUS and MASSBUS devices)
;ACG0399      0830 .158 ;         SCSSYSTEMID address (for class drivers during SYSGEN driver
;ACG0399      0830 .159 ;         loading)
;ACG0399      0830 .160 ;         zero for all others, including class drivers during power
;ACG0399      0830 .161 ;         failure recovery
;ACG0399      0830 .162 ;         R5      IDB address (or zero if none exists)
;ACG0399      0830 .163 ;         R6      DDB address
;ACG0399      0830 .164 ;         R8      CRB address
;ACG0399      0830 .165 ;
;ACG0399      0830 .166 ; OUTPUTS:
;ACG0399      0830 .167 ;         Must preserve all registers except R0 through R4.
;ACG0399      0830 .168 ;
;ACG0399      0830 .169 ;--
;ACG0399      0830 .170 ;
;ACG0399      0830 .171 ;
;ACG0399      0830 .172 IOC$CTRLINIT::
;ACG0399      0830 .173
;ACG0399      55  2C A8  D0 0830 .174      MOVL   CRB$L_INTD+VEC$L_IDB(R8), R5      ; Get IDB address.
;ACG0399      0834 .175      BGEQ   10$                          ; Branch if none.
;ACG0399      54  65  D0 0836 .176      MOVL   IDB$L_CSR(R5), R4      ; Get CSR address.
;ACG0399      0839 .177      BLSS   20$                          ; Branch if really a CSR.
;ACG0399      54  51  D0 083B .178 10$: MOVL   R1, R4                          ; Else, use supplied value,
;ACG0399      083E .179      BRB    40$                          ; and skip CSR testing.
;ACG0399      0840 .180
;ACG0399      56  14 A5  D0 0840 .181 20$: MOVL   IDB$L_ADPT(R5), R6      ; Get ADP address.
;ACG0399      0844 .182      BGEQ   40$                          ; If none, skip CSR test.
;ACG0399      0E A6  01  B1 0846 .183      CMPW   #AT$_UBA, ADP$_ADPTYPE(R6) ; Is this a UBA?
;ACG0399      084A .184      BNEQ   40$                          ; If not a UBA, skip CSR test.
;ACG0399      56  66  D0 084C .185      MOVL   ADP$L_CSR(R6), R6      ; Get adapter config reg addr.
;ACG0399      50  54  D0 084F .186      MOVL   R4, R0                          ; Setup CSR for test.
;ACG0399      00000000 GF 16 0852 .187      JSB    G^EXE$TEST_CSR          ; Test UNIBUS CSR.
;ACG0399      0E 50  E9 0858 .188      BLBC   R0, 90$                  ; Branch if no CSR present.
;ACG0399      085B .189

```

```
;ACG0399      50  30 A8  D0 085B .190 40$:  MOVL  CRB$_INTD+VEC$_INITIAL(R8), R0 ; Get ctrl init rout addr.
;ACG0399      05  18 085F .191          BGEQ  80$          ; Branch if none.
;ACG0399      56  5B  D0 0861 .192          MOVL  R11, R6          ; Get DDB address.
;ACG0399      60  16 0864 .193          JSB   (R0)           ; Call ctrl init routine.
;ACG0399      05  16 0866 .194          ;
;ACG0399      50  01  D0 0866 .195 80$:  MOVL  #1, R0          ; Set success status.
;ACG0399      05  05 0869 .196 90$:  RSB                    ; Return w/ status.
```

```

;ACG0399      086A .198      .SBTTL IOC$UNITINIT - Call driver unit init. routine
;ACG0399      086A .199 ;++
;ACG0399      086A .200 ;   FUNCTIONAL DESCRIPTION:
;ACG0399      086A .201 ;
;ACG0399      086A .202 ;       Input values for a device driver's unit initialization routine are
;ACG0399      086A .203 ;       loaded into the proper registers, the routine starting address is
;ACG0399      086A .204 ;       located, and if a routine exists, it is called.
;ACG0399      086A .205 ;
;ACG0399      086A .206 ;   INPUTS:
;ACG0399      086A .207 ;       R5       UCB address
;ACG0399      086A .208 ;       R8       CRB address (primary)
;ACG0399      086A .209 ;
;ACG0399      086A .210 ;   OUTPUTS:
;ACG0399      086A .211 ;       R0-R4   Destroyed
;ACG0399      086A .212 ;
;ROW0221      086A .213 ;   NOTES:
;ROW0221      086A .214 ;
;ROW0221      086A .215 ;       There are two unit initialization routine addresses in the I/O data
;ROW0221      086A .216 ;       base; CRB$_INTD_VEC$_UNITINIT and DDT$_UNITINIT. Normally, only
;ROW0221      086A .217 ;       one of these two places should contain a unit initialization routine
;ROW0221      086A .218 ;       address. However, for the console block storage device, the both
;ROW0221      086A .219 ;       locations contain an address, and the DDT contains the address which
;ROW0221      086A .220 ;       must be used.
;ROW0221      086A .221 ;
;ROW0221      086A .222 ;       In this case, the CRB is shared by the console terminal and console
;ROW0221      086A .223 ;       block storage devices. The CRB contains the address of the unit
;ROW0221      086A .224 ;       initialization routine for the console terminal, and the console
;ROW0221      086A .225 ;       terminal DDT contains no unit initialization routine address. Thus
;ROW0221      086A .226 ;       the console terminal device "fits" the "normal" case. However, the
;ROW0221      086A .227 ;       console block storage device DDT contains a unit initialization
;ROW0221      086A .228 ;       routine which differs from the console terminal unit initialization
;ROW0221      086A .229 ;       routine and whose address is stored in the DDT.
;ROW0221      086A .230 ;
;ROW0221      086A .231 ;       Since the CRB is shared and contains the wrong unit initialization
;ROW0221      086A .232 ;       routine address for the console block storage device, the DDT must be
;ROW0221      086A .233 ;       inspected first. Initialization for the console terminal will be
;ROW0221      086A .234 ;       accomplished correctly regardless of which address is checked first.
;ROW0203      086A .235 ;
;ROW0203      086A .236 ;--
;ROW0203      086A .237 ;
;ROW0203      086A .238 ;++
;ROW0203      086A .239 ;   Unit initialization routine parameters:
;ROW0203      086A .240 ;
;ROW0203      086A .241 ;   INPUTS:
;ROW0203      086A .242 ;       R3       CSR address (primary)
;ROW0203      086A .243 ;       R4       CSR address (secondary, same as primary if no secondary exists)
;ROW0203      086A .244 ;       R5       UCB address
;ROW0203      086A .245 ;
;ROW0203      086A .246 ;   OUTPUTS:
;ROW0203      086A .247 ;       Must preserve all registers except R0 through R4.
;ROW0203      086A .248 ;
;ROW0203      086A .249 ;--
;ROW0203      086A .250 ;
;ROW0203      086A .251 ;
;ROW0203      086A .252 IOC$UNITINIT::
;ROW0203      086A .253 ;
;ROW0221      50   0088 C5   D0 086A .254      MOVL   UCB$_DDT(R5), R0      ; Get DDT address.

```

```

;ROW0221      50  18 A0  D0 086F .255      MOVL  DDT$_UNITINIT(R0), R0      ; Get DDT unit init rout addr.
;ROW0221      50  50  D1 0873 .256      CMPL  R0, #IOC$RETURN           ; Null unit init routine?
000005D9'8F    0875
;ROW0221      06  12 087A .257      BNEQ  10$                      ; Branch if real unit init rout.
;ROW0221      50  3C A8  D0 087C .258      MOVL  CRB$_INTD+VEC$_UNITINIT(R8), R0 ; Get CRB unit init rout addr.
;ROW0221      1A  18 0880 .259      BGEQ  90$                      ; Branch if no unit init rout.
;ROW0203      0882 .260
;ROW0203      54  D4 0882 .261 10$:    CLRL  R4                      ; Assume no IDB exists.
;ROW0203      53  2C A8  D0 0884 .262      MOVL  CRB$_INTD+VEC$_IDB(R8), R3    ; Get IDB address.
;ROW0203      10  18 0888 .263      BGEQ  50$                      ; Branch if none.
;ROW0203      53  63  D0 088A .264      MOVL  IDB$_CSR(R3), R3           ; Get primary CSR.
;ROW0203      54  53  D0 088D .265      MOVL  R3, R4                   ; Assume no sec. CRB exists.
;ROW0203      51  20 A8  D0 0890 .266      MOVL  CRB$_LINK(R8), R1        ; Get secondary CRB addr.
;ROW0203      04  18 0894 .267      BGEQ  50$                      ; Branch if none.
;ROW0203      0896 .268      ASSUME IDB$_CSR EQ 0
;ROW0203      54  2C B1  D0 0896 .269      MOVL  @CRB$_INTD+VEC$_IDB(R1), R4 ; Get secondary CSR addr.
;ROW0203      089A .270
;ROW0203      60  17 089A .271 50$:    JMP   (R0)                    ; Call unit init routine, and
;ROW0203      089C .272      ; return to caller.
;ROW0221      089C .273
;ROW0221      05  089C .274 90$:    RSB                               ; No unit init routine to call:
;ROW0221      089D .275      ; return to caller.

```

```

;ACG0414      089D .277      .SBTTL Parse Device Name String
;ACG0414      089D .278
;ACG0414      089D .279 ;+
;ACG0414      089D .280 ;
;ACG0414      089D .281 ; IOC$PARSDEVNAM - parse device name string
;ACG0414      089D .282 ;
;ACG0414      089D .283 ; This routine parses a device name string, checking syntax and
;ACG0414      089D .284 ; extracting node name, allocation class number, and unit number.
;ACG0414      089D .285 ; If device type format is specified, it is converted into the internal
;ACG0414      089D .286 ; compressed format.
;ACG0414      089D .287 ;
;ACG0414      089D .288 ; INPUTS:
;ACG0414      089D .289 ;
;ACG0414      089D .290 ;      R8 = size of name string
;ACG0414      089D .291 ;      R9 = address of name string
;ACG0414      089D .292 ;      R10 = flags
;ACG0414      089D .293 ;
;ACG0414      089D .294 ; OUTPUTS:
;ACG0414      089D .295 ;
;ACG0414      089D .296 ;      R0 = SS$_NORMAL - valid name string
;ACG0414      089D .297 ;      = SS$_IVDEVNAM - invalid device name string
;ACG0414      089D .298 ;      R2 = unit number
;ACG0414      089D .299 ;      R3 = length of SCS node name at head of name string
;ACG0414      089D .300 ;           or allocation class number
;ACG0414      089D .301 ;           or device type code
;ACG0414      089D .302 ;      R8 = size of name string
;ACG0414      089D .303 ;      R9 = address of name string
;ACG0414      089D .304 ;      R10 = flags
;ACG0414      089D .305 ;      R4 - R7, R11 preserved
;ACG0414      089D .306 ;
;ACG0414      089D .307 ;-
;ACG0414      089D .308
;ACG0414      089D .309      .ENABLE LSB
;ACG0414      089D .310
;ACG0414      089D .311 IOC$PARSDEVNAM::
;ACG0414      0070 8F BB 089D .312 PUSHR #^M<R4,R5,R6> ; save working registers
;ACG0414      58 D5 08A1 .313 TSTL R8 ; check name string length
;ACG0414      28 13 08A3 .314 BEQL 30$ ; branch if null - error
;ACG0414      54 58 7D 08A5 .315 MOVQ R8,R4 ; copy name string descriptor
;ACG0414      56 59 01 C3 08A8 .316 SUBL3 #1,R9,R6 ; default is no node no allocation
;ACG0414      08AC .317 ; class, set pointer before beginning
;ACG0414      08AC .318 ; of the string
;ACG0414      69 58 24 3A 08AC .319 LOCC #^A'$',R8,(R9) ; scan name for a "$"
;ACG0414      03 13 08B0 .320 BEQL 10$ ; failed to find one - no nodename
;ACG0414      56 51 D0 08B2 .321 MOVL R1,R6 ; found it, save pointer
;ACG0414      52 7C 08B5 .322 10$: CLRQ R2 ; init unit number and node name
;ACG0414      50 65 9A 08B7 .323 20$: MOVZBL (R5),R0 ; get next character
;ACG0414      11 50 06 E1 08BA .324 BBC #6,R0,40$ ; br if code 0-AX3F - numeric or $
;ACG0414      50 20 8A 08BE .325 BICB #^X20,R0 ; collapse lower case to upper case
;ACG0414      5A 8F 50 91 08C1 .326 CMPB R0,#^A'Z' ; possible alphabetic?
;ACG0414      77 1A 08C5 .327 BGTRU 150$ ; br if not
;ACG0414      41 8F 50 91 08C7 .328 CMPB R0,#^A'A' ; possible alphabetic?
;ACG0414      37 1E 08CB .329 BGEQU 70$ ; branch if OK - store it
;ACG0414      6F 11 08CD .330 30$: BRB 150$ ; no - error
;ACG0414      08CF .331 ;
;ACG0414      08CF .332 ; Non alphabetic - may be numeric or "$"
;ACG0414      08CF .333 ;

```

```

;ACG0414      56  55  D1  08CF .334 40$:  CMPL   R5,R6           ; hit the "$" yet?
;ACG0414      56  55  OE  13 08D2 .335      BEQL   50$           ; yes, deal with it
;ACG0414      56  34  1A  08D4 .336      BGTRU  80$           ; past it, digits are unit number
;ACG0414      39  50  91  08D6 .337      CMPB   R0,#^A'9     ; legal?
;ACG0414      39  63  1A  08D9 .338      BGTRU  150$          ; no, error
;ACG0414      30  50  91  08DB .339      CMPB   R0,#^A'0'    ; legal?
;ACG0414      30  24  1E  08DE .340      BGEQU  70$           ; yes, accept it as alpha
;ACG0414      30  5C  11  08E0 .341      BRB    150$          ; no, error
;ACG0414      08E2 .342 ;
;ACG0414      08E2 .343 ; $ in device name - either node name or allocation class.
;ACG0414      08E2 .344 ;
;ACG0414      53  55  59  C3 08E2 .345 50$:  SUBL3  R9,R5,R3      ; compute length of node name
;ACG0414      53  55  1C  12 08E6 .346      BNEQ   70$           ; branch if non-null - process the $
;ACG0414      08E8 .347 ;
;ACG0414      08E8 .348 ; Process allocation class number.
;ACG0414      08E8 .349 ;
;ACG0414      55  D6 08E8 .350 60$:  INCL   R5           ; move over "$" to allocation
;ACG0414      54  D7 08EA .351      DECL  R4           ; class digits.
;ACG0414      6A  10 08EC .352      BSBB  GETNUMBER     ; convert allocation class.
;ACG0414      53  52  D0 08EE .353      MOVL  R2,R3         ; store requested allocation class.
;ACG0414      4B  15 08F1 .354      BLEQ  150$          ; leq zero is not legal.
;ACG0414      5A  04  88 08F3 .355      BISB  #IOC$M_CLASS,R10 ; set allocation class flag
;ACG0414      50  24  91 08F6 .356      CMPB  #^A'$',R0     ; was terminator a "$"?
;ACG0414      43  12 08F9 .357      BNEQ  150$          ; if not, invalid device name.
;ACG0414      58  54  7D 08FB .358      MOVQ  R4,R8         ; reset device name - unit size.
;ACG0414      54  D5 08FE .359      TSTL  R4           ; check remaining string count
;ACG0414      B5  14 0900 .360      BGTR  20$           ; if characters remain, process them.
;ACG0414      3A  11 0902 .361      BRB   150$          ; else, invalid device name.
;ACG0414      0904 .362 ;
;ACG0414      85  50  90 0904 .363 70$:  MOVB   R0,(R5)+      ; store potentially upcased character
;ACG0414      AD  54  F5 0907 .364      SOBGTR R4,20$       ; any more characters to scan?
;ACG0414      090A .365 ;
;ACG0414      090A .366 ; End of alpha scan. Make sure we actually got a non-null device name.
;ACG0414      090A .367 ;
;ACG0414      58  54  C2 090A .368 80$:  SUBL   R4,R8         ; subtract unit number from string
;ACG0414      2F  13 090D .369      BEQL  150$          ; if eql no device name specified
;ACG0414      56  D6 090F .370      INCL  R6           ; point past $ in node name
;ACG0414      55  56  D1 0911 .371      CMPL  R6,R5         ; see if we've processed any more chars
;ACG0414      09  1F 0914 .372      BLSSU 90$           ; branch if yes
;ACG0414      25  5A  E8 0916 .373      BLBS  R10,150$      ; branch if physical - not valid
;ACG0414      21  5A  06  E1 0919 .374      BBC   #IOC$V_ANY,R10,150$ ; or if not generic search for any
;ACG0414      0D  11 091D .375      BRB   100$          ; node name only - verify end of string
;ACG0414      091F .376 ;
;ACG0414      091F .377 ; Process unit number and make sure there's no trailing junk.
;ACG0414      091F .378 ;
;ACG0414      52  D4 091F .379 90$:  CLRL  R2           ; init unit number to 0
;ACG0414      54  D5 0921 .380      TSTL  R4           ; see if there's anything left
;ACG0414      0B  15 0923 .381      BLEQ  110$          ; branch if not
;ACG0414      5A  01  88 0925 .382      BISB  #IOC$M_PHY,R10 ; set physical flag
;ACG0414      2E  10 0928 .383      BSBB  GETNUMBER     ; convert unit number
;ACG0414      54  D6 092A .384      INCL  R4           ; return terminator to string count
;ACG0414      54  D5 092C .385 100$:  TSTL  R4           ; reached end of string?
;ACG0414      0E  14 092E .386      BGTR  150$          ; branch if not - error
;ACG0414      37  5A  01  E0 0930 .387 110$:  BBS   #IOC$V_TYPE,R10,190$ ; branch if name is a device type
;ACG0414      50  01  D0 0934 .388 120$:  MOVL  #SS$_NORMAL,R0 ; successful parse
;ACG0414      0070 8F  BA 0937 .389 130$:  POPR  #^M<R4,R5,R6> ; restore registers
;ACG0414      05  093B .390      RSB                    ; and return

```



```

;ACG0414          093C .391 ;
;ACG0414          093C .392 ; Invalid device name
;ACG0414          093C .393 ;
;ACG0414          093C .394 140$: TSTL (SP)+          ; pop GETNUMBER return address from stack
50  0144  8E  D5  093E .395 150$: MOVZWL #SS$_IVDEVNAM,R0 ; set invalid device name
;ACG0414          8F  3C  093E .395 150$: MOVZWL #SS$_IVDEVNAM,R0 ; set invalid device name
;ACG0414          F2  11  0943 .396          BRB 130$
;ACG0414          0945 .397
;ACG0414          0945 .398 ;+
;ACG0414          0945 .399 ; Routine to convert ASCII to integer
;ACG0414          0945 .400 ;
;ACG0414          0945 .401 ; Inputs:
;ACG0414          0945 .402 ;
;ACG0414          0945 .403 ;          R2          assumed zero
;ACG0414          0945 .404 ;          R4          size of string
;ACG0414          0945 .405 ;          R5          starting address of string
;ACG0414          0945 .406 ;
;ACG0414          0945 .407 ; Outputs:
;ACG0414          0945 .408 ;
;ACG0414          0945 .409 ;          R0          terminator character
;ACG0414          0945 .410 ;          R2          converted number
;ACG0414          0945 .411 ;          R4          size of string with number and terminator character removed
;ACG0414          0945 .412 ;          R5          address of first character after number terminator character
;ACG0414          0945 .413 ;-
;ACG0414          0945 .414
;ACG0414          50  85  9A  0945 .415 160$: MOVZBL (R5)+,R0          ; get next character.
;ACG0414          50  30  82  0948 .416          SUBB #^A'0',R0          ; base it at decimal digits.
;ACG0414          09  10  1F  094B .417          BLSSU 170$          ; branch if not a decimal digit.
;ACG0414          09  50  91  094D .418          CMPB  R0,#9          ; is it a digit?
;ACG0414          09  0B  1A  0950 .419          BGTRU 170$          ; branch if not a decimal digit.
;ACG0414          52  0A  C4  0952 .420          MULL  #10,R2          ; scale current unit number by 10
;ACG0414          52  50  C0  0955 .421          ADDL  R0,R2          ; add new digit to accumulation.
;ACG0414          0958 .422 GETNUMBER:
;ACG0414          EA  54  F4  0958 .423          SOBGEQ R4,160$          ; count off a character
;ACG0414          04  11  095B .424          BRB 180$          ; branch if no more characters
;ACG0414          095D .425
;ACG0414          50  FF  A5  9A  095D .426 170$: MOVZBL -1(R5),R0          ; restore terminator character.
;ACG0414          52  50  D1  0961 .427 180$: CMPL  R2,#32768          ; check number value
00008000 8F          0963
;ACG0414          D2  1E  0968 .428          BGEQU 140$          ; branch if not valid
;ACG0414          05  096A .429          RSB          ; return to caller.
;ACG0414          096B .430
;ACG0414          096B .431 ;
;ACG0414          096B .432 ; Parse device type name. We do this last because all the regular device
;ACG0414          096B .433 ; name validation is necessary anyway. Now we just have to do the
;ACG0414          096B .434 ; additional checks and pack the characters.
;ACG0414          096B .435 ;
;ACG0414          53  D5  096B .436 190$: TSTL  R3          ; check if we saw node or alloc class
;ACG0414          CF  12  096D .437          BNEQ 150$          ; branch if so - not valid
;ACG0414          50  55  59  C3  096F .438          SUBL3 R9,R5,R0          ; compute total length of string
;ACG0414          50  58  C2  0973 .439          SUBL  R8,R0          ; compute length of unit number string
;ACG0414          02  50  D1  0976 .440          CMPL  R0,#2          ; must be two digits
;ACG0414          C3  12  0979 .441          BNEQ 150$          ; branch if not - not valid
;ACG0414          55  59  D0  097B .442          MOVL  R9,R5          ; copy name address again
;ACG0414          02  58  D1  097E .443          CMPL  R8,#2          ; check minimum name length
;ACG0414          BB  1F  0981 .444          BLSSU 150$          ; too short - out
;ACG0414          85  40  8F  83  0983 .445          SUBB3 #^A'A'-1,(R5)+,R0 ; get char and convert to compressed
;ACG0414          50  0987

```

```
;ACG0414      05  11  50  F0 0988 .446      INSV   R0,#17,#5,R3      ; store compressed code
                53      098C
;ACG0414      85  40  8F  83 098D .447      SUBB3   #^A'A'-1,(R5)+,R0    ; get char and convert to compressed
                50      0991
;ACG0414      05  0C  50  F0 0992 .448      INSV   R0,#12,#5,R3      ; store compressed code
                53      0996
;ACG0414      03  58  D1  0997 .449      CMPL   R8,#3             ; check how many chars left
;ACG0414      A2  1A  099A .450      BGTRU  150$             ; string was longer than 5 - error
;ACG0414      0A  1F  099C .451      BLSSU  200$             ; short - don't take 3rd alpha
;ACG0414      85  40  8F  83 099E .452      SUBB3   #^A'A'-1,(R5)+,R0    ; get char and convert to compressed
                50      09A2
;ACG0414      05  07  50  F0 09A3 .453      INSV   R0,#7,#5,R3       ; store compressed code
                53      09A7
;ACG0414      53  52  C0  09A8 .454 200$:  ADDL   R2,R3             ; add in unit number
;ACG0414      5A  01  8A  09AB .455      BICB   #IOCSM_PHY,R10     ; clear physical flag
;ACG0414      FF83 31  09AE .456      BRW    120$             ; done
```

```

;ACG0414      09B1 .458      .SBTTL Search I/O Database for Device
;ACG0414      09B1 .459
;ACG0414      09B1 .460 ;+
;ACG0414      09B1 .461 ;
;ACG0414      09B1 .462 ; IOC$SEARCHINT - internal I/O database search
;ACG0414      09B1 .463 ;
;ACG0414      09B1 .464 ; This routine searches the I/O database for the specified device, using
;ACG0414      09B1 .465 ; the specified search rules. Depending on the search, a lock may or may
;ACG0414      09B1 .466 ; not be taken out on the device when it is found.
;ACG0414      09B1 .467 ;
;ACG0414      09B1 .468 ; Note! While this routine is non-paged and therefore may be called at
;ACG0414      09B1 .469 ; elevated IPL, the device locking code it calls is not. Therefore, only
;ACG0414      09B1 .470 ; searches with IOC$V_ANY may be called from elevated IPL.
;ACG0414      09B1 .471 ;
;ACG0414      09B1 .472 ; INPUTS:
;ACG0414      09B1 .473 ;
;ACG0414      09B1 .474 ;         R2 = unit number
;ACG0414      09B1 .475 ;         R3 = length of SCS node name at head of name string
;ACG0414      09B1 .476 ;                or allocation class number
;ACG0414      09B1 .477 ;                or device type code
;ACG0414      09B1 .478 ;         R8 = size of name string
;ACG0414      09B1 .479 ;         R9 = address of name string
;ACG0414      09B1 .480 ;         R10 = flags
;ACG0414      09B1 .481 ;         R11 = address to store lock value block
;ACG0414      09B1 .482 ;         I/O database mutex held, IPL 2
;ACG0414      09B1 .483 ;
;ACG0414      09B1 .484 ; OUTPUTS:
;ACG0414      09B1 .485 ;
;ACG0414      09B1 .486 ;         R0 = SS$_NORMAL - device found
;ACG0414      09B1 .487 ;         = SS$_NOSUCHDEV - device not found
;ACG0414      09B1 .488 ;         = SS$_NODEVAVL - device exists but not available according to rules
;ACG0414      09B1 .489 ;         = SS$_DEVALLOC - device allocated to other user
;ACG0414      09B1 .490 ;         = SS$_NOPRIV - failed device protection
;ACG0414      09B1 .491 ;         = SS$_TEMPLATEDEV - can't allocate template device
;ACG0414      09B1 .492 ;         = SS$_DEVMOUNT - device already mounted
;ACG0414      09B1 .493 ;         = SS$_DEVOFFLINE - device marked offline
;ACG0414      09B1 .494 ;         R5 = UCB
;ACG0414      09B1 .495 ;         R6 = DDB
;ACG0414      09B1 .496 ;         R7 = system block
;ACG0414      09B1 .497 ;         R10 - R4, R8 - R11 preserved
;ACG0414      09B1 .498 ;
;ACG0414      09B1 .499 ;         Note: If failure, R5 - R7 point to the last structures looked at.
;ACG0414      09B1 .500 ;
;ACG0414      09B1 .501 ; -
;ACG0414      09B1 .502
;ACG0414      09B1 .503 ;
;ACG0414      09B1 .504 ; Stack use:
;ACG0414      09B1 .505 ;
;ACG0414      00000000 09B1 .506 SAVR2 = 0
;ACG0414      00000004 09B1 .507 SAVR3 = 4
;ACG0414      00000008 09B1 .508 SAVR4 = 8
;ACG0414      0000000C 09B1 .509 SAVR8 = 12
;ACG0414      00000010 09B1 .510 SAVR9 = 16
;ACG0414      09B1 .511
;ACG0414      09B1 .512
;ACG0414      09B1 .513      .ENABLE LSB
;ACG0414      09B1 .514

```

```
;ACG0414          09B1 .515 IOC$SEARCHINT::
;ACG0414          09B1 .516          PUSHR  #^M<R2,R3,R4,R8,R9>      ; save registers
;ACG0414          09B5 .517 ;
;ACG0414          09B5 .518 ; Search the system blocks for a suitable node. If we are doing a search
;ACG0414          09B5 .519 ; by allocation class, generic device type, or no node name is given,
;ACG0414          09B5 .520 ; all system blocks qualify.
;ACG0414          09B5 .521 ;
;ACG0414          09B5 .522          MOVAL  SCS$GQ_CONFIG,R7          ; get head of SCS SB list
00000000'EF DE 09B5
;ACG0414          57 09BB
;ACG0414          50 67 DO 09BC .523 10$: MOVL  SB$L_FLINK(R7),R0      ; get next system block
;ACG0414          50 D1 09BF .524          CMPL  R0,#SCS$GQ_CONFIG    ; are we back at list head?
00000000'8F
;ACG0414          78 13 09C6 .525          BEQL  50$                  ; branch if yes - all done
;ACG0414          09C8 .526
;ACG0414          57 50 DO 09C8 .527          MOVL  R0,R7
;ACG0414          56 54 A7 DE 09CB .528          MOVAL  SB$L_DDB-DDB$L_LINK(R7),R6 ; pick up DDB listhead
;ACG0414          55 56 DO 09CF .529          MOVL  R6,R5          ; make sure UCB is non-zero
;ACG0414          09D2 .530          ; if allocation class or generic dev,
;ACG0414          5A 06 93 09D2 .531          BITB  #IOC$M_CLASS!IOC$M_TYPE,R10
;ACG0414          27 12 09D5 .532          BNEQ  30$                  ; check every system block
;ACG0414          58 0C AE 7D 09D7 .533          MOVQ  SAVR8(SP),R8          ; get orig dev name descriptor
;ACG0414          53 04 AE DO 09DB .534          MOVL  SAVR3(SP),R3        ; get node name length
;ACG0414          1D 13 09DF .535          BEQL  30$                  ; branch if none - go ahead
;ACG0414          44 A7 53 91 09E1 .536          CMPB  R3,SB$T_NODENAME(R7) ; check node name length
;ACG0414          D5 12 09E5 .537          BNEQ  10$                  ; branch if not
;ACG0414          45 A7 53 29 09E7 .538          CMPC3  R3,SB$T_NODENAME+1(R7),(R9) ; node names match?
;ACG0414          69 09EB
;ACG0414          CE 12 09EC .539          BNEQ  10$                  ; branch if not
;ACG0414          09EE .540 ;
;ACG0414          09EE .541 ; Found a suitable system block. Search its DDB list.
;ACG0414          09EE .542 ;
;ACG0414          50 01 3C 09EE .543 20$: MOVZWL #SS$ NORMAL,R0
;ACG0414          04 AE 01 C1 09F1 .544          ADDL3  #1,SAVR3(SP),R3      ; include the "$"
;ACG0414          53 09F5
;ACG0414          59 53 C0 09F6 .545          ADDL  R3,R9          ; skip over the nodename
;ACG0414          58 53 C2 09F9 .546          SUBL  R3,R8          ; adjust the length
;ACG0414          52 15 09FC .547          BLEQ  60$                  ; if no device name, just return SB
;ACG0414          09FE .548
;ACG0414          50 66 DO 09FE .549 30$: MOVL  DDB$L_LINK(R6),R0      ; get address of next DDB
;ACG0414          5A 13 0A01 .550          BEQL  80$                  ; if eql end of list
;ACG0414          56 50 DO 0A03 .551          MOVL  R0,R6
;ACG0414          55 D4 A6 DE 0A06 .552          MOVAL  <DDB$L_UCB-UCB$L_LINK>(R6),R5 ; initialize primary UCB address
;ACG0414          5A 20 8A 0A0A .553          BICB  #IOC$M_2P,R10      ; new DDB - clear secondary flag
;ACG0414          5E 5A 01 E0 0A0D .554          BBS  #IOC$V_TYPE,R10,100$ ; branch if generic type search
;ACG0414          07 5A 02 E1 0A11 .555          BBC  #IOC$V_CLASS,R10,40$ ; branch if no class to check
;ACG0414          3C A6 04 AE D1 0A15 .556          CMPL  SAVR3(SP),DDB$L_ALLOCLS(R6) ; else, is allo. class right?
;ACG0414          E2 12 0A1A .557          BNEQ  30$                  ; branch if not, try next DDB
;ACG0414          69 58 29 0A1C .558 40$: CMPC3  R8,(R9),DDB$T_NAME+1(R6) ; check device name
;ACG0414          15 A6 0A1F
;ACG0414          DB 12 0A21 .559          BNEQ  30$                  ; if no match, try next DDB
;ACG0414          50 14 A6 9A 0A23 .560          MOVZBL DDB$T_NAME(R6),R0 ; get length of name in DDB
;ACG0414          50 58 D1 0A27 .561          CMPL  R8,R0          ; check name lengths
;ACG0414          43 13 0A2A .562          BEQL  100$              ; if they match - OK
;ACG0414          50 D7 0A2C .563          DECL  R0                ; try subtracting out controller letter
;ACG0414          50 58 D1 0A2E .564          CMPL  R8,R0          ; and see if this matches
;ACG0414          CB 12 0A31 .565          BNEQ  30$                  ; if not, keep trying
;ACG0414          39 5A E9 0A33 .566          BLBC  R10,100$          ; branch if not physical search - OK
```

```

;ACG0414      15 A640  91 0A36 .567      CMPB      DDB$_NAME+1(R6)[R0],#^A^A' ; is this controller A?
                41 8F      0A3A
;ACG0414      31      13 0A3C .568      BEQL      100$ ; if so, search it
;ACG0414      BE      11 0A3E .569      BRB      30$ ; if not, keep looking
;ACG0414      0A40 .570 ;
;ACG0414      0A40 .571 ; End of search - no suitable device has been found
;ACG0414      0A40 .572 ;
;ACG0414      50 0908 8F 3C 0A40 .573 50$: MOVZWL #SS$ NOSUCHDEV,R0 ; no device found
;ACG0414      4D 5A 04 E1 0A45 .574      BBC      #IOC$_V_EXISTS,R10,140$ ; branch if not seen
;ACG0414      50 09B0 8F 3C 0A49 .575      MOVZWL #SS$_NODEVAVL,R0 ; otherwise status is not available
;ACG0414      46      11 0A4E .576      BRB      140$
;ACG0414      0A50 .577 ;
;ACG0414      0A50 .578 ; To here if we're just returning a system block, with no device specified.
;ACG0414      0A50 .579 ;
;ACG0414      56 66  D0 0A50 .580 60$: MOVL      (R6),R6 ; get first DDB
;ACG0414      55 04 A6  D0 0A53 .581      MOVL      DDB$_L_UCB(R6),R5 ; and first UCB
;ACG0414      3D      11 0A57 .582      BRB      140$ ; and return
;ACG0414      0A59 .583 ;
;ACG0414      0A59 .584 ; To here when all UCB's on a DDB have been searched.
;ACG0414      0A59 .585 ;
;ACG0414      A1 5A 01  E0 0A59 .586 70$: BBS      #IOC$_V_TYPE,R10,30$ ; if generic type search, try next DDB
;ACG0414      0A5D .587 ;
;ACG0414      0A5D .588 ; To here when all DDB's on a system block have been searched.
;ACG0414      0A5D .589 ;
;ACG0414      5A 06  93 0A5D .590 80$: BITB      #IOC$_M_CLASS!IOC$_M_TYPE,R10 ; if generic type or alloc class
;ACG0414      0A      12 0A60 .591      BNEQ      90$ ; keep searching system blocks
;ACG0414      5A 09  93 0A62 .592      BITB      #IOC$_M_PHY!IOC$_M_LOCAL,R10 ; if physical or local only
;ACG0414      0A      12 0A65 .593      BNEQ      50$ ; we're done
;ACG0414      04 AE  D5 0A67 .594      TSTL      SAVR3(SP) ; if there was an explicit node
;ACG0414      D4      12 0A6A .595      BNEQ      50$ ; we're done
;ACG0414      FF4D 31 0A6C .596 90$: BRW      10$ ; else go try next system block
;ACG0414      0A6F .597 ;
;ACG0414      0A6F .598 ; Found a suitable DDB. Search both its UCB lists for the right UCB.
;ACG0414      0A6F .599 ;
;ACG0414      52 6E  7D 0A6F .600 100$: MOVQ      SAVR2(SP),R2 ; get unit number and device type
;ACG0414      00000000 EF D0 0A72 .601      MOVL      SCH$_GL_CURPCB,R4 ; get PCB address
;ACG0414      54      0A78
;ACG0414      0A79 .602 NEXTOCB:
;ACG0414      07 5A 05  E1 0A79 .603 110$: BBC      #IOC$_V_2P,R10,120$ ; re-entry for next UCB
;ACG0414      55 00A4 C5  D0 0A7D .604      MOVL      UCB$_L_2P_LINK(R5),R5 ; branch if on primary path
;ACG0414      04      11 0A82 .605      BRB      130$ ; link to next secondary unit.
;ACG0414      55 30 A5  D0 0A84 .606 120$: MOVL      UCB$_L_LINK(R5),R5 ; link to next primary unit.
;ACG0414      11      13 0A88 .607 130$: BEQL      150$ ; branch if no more units.
;ACG0414      28      10 0A8A .608      BSBB      IOC$_TESTUNIT ; is this unit ok?
;ACG0414      07 50  E8 0A8C .609      BLBS      R0,140$ ; branch if successful
;ACG0414      E6 5A 04  E1 0A8F .610      BBC      #IOC$_V_EXISTS,R10,110$ ; keep going if we haven't seen it yet
;ACG0414      E3 5A  E9 0A93 .611      BLBC      R10,110$ ; or if not physical search
;ACG0414      031C 8F  BA 0A96 .612 140$: POPR      #^M<R2,R3,R4,R8,R9> ; restore registers
;ACG0414      05      0A9A .613      RSB ; and return
;ACG0414      0A9B .614
;ACG0414      BA 5A 05  E2 0A9B .615 150$: BBSS      #IOC$_V_2P,R10,70$ ; branch if secondary path already searched
;ACG0414      55 9C A6  DE 0A9F .616      MOVAL     <DDB$_L_2P_UCB - ; initialize secondary UCB address.
;ACG0414      0AA3 .617      -UCB$_L_2P_LINK>(R6),R5
;ACG0414      D4      11 0AA3 .618      BRB      110$ ; go search secondary path
;ACG0414      0AA5 .619
;ACG0414      0AA5 .620      .DISABLE LSB

```

```

;ACG0414      OAA5 .622      .SBTTL Continue I/O Database Search
;ACG0414      OAA5 .623
;ACG0414      OAA5 .624 ;+
;ACG0414      OAA5 .625 ;
;ACG0414      OAA5 .626 ; IOC$SEARCHCONT - internal I/O database search
;ACG0414      OAA5 .627 ;
;ACG0414      OAA5 .628 ; This routine continues a search started with a call to IOC$SEARCHINT.
;ACG0414      OAA5 .629 ; It uses IOC$SEARCHINT's outputs as the starting point at which to
;ACG0414      OAA5 .630 ; resume.
;ACG0414      OAA5 .631 ;
;ACG0414      OAA5 .632 ; INPUTS:
;ACG0414      OAA5 .633 ;
;ACG0414      OAA5 .634 ;         R2 = unit number
;ACG0414      OAA5 .635 ;         R3 = length of SCS node name at head of name string
;ACG0414      OAA5 .636 ;         or allocation class number
;ACG0414      OAA5 .637 ;         or device type code
;ACG0414      OAA5 .638 ;         R5 = last UCB
;ACG0414      OAA5 .639 ;         R6 = last DDB
;ACG0414      OAA5 .640 ;         R7 = last system block
;ACG0414      OAA5 .641 ;         R8 = size of name string
;ACG0414      OAA5 .642 ;         R9 = address of name string
;ACG0414      OAA5 .643 ;         R10 = flags
;ACG0414      OAA5 .644 ;         R11 = address to store lock value block
;ACG0414      OAA5 .645 ;         I/O database mutex held, IPL 2
;ACG0414      OAA5 .646 ;
;ACG0414      OAA5 .647 ; OUTPUTS:
;ACG0414      OAA5 .648 ;
;ACG0414      OAA5 .649 ;         R0 = SS$_NORMAL - device found
;ACG0414      OAA5 .650 ;         = SS$_NOSUCHDEV - device not found
;ACG0414      OAA5 .651 ;         = SS$_NODEVAVL - device exists but not available according to rules
;ACG0414      OAA5 .652 ;         = SS$_DEVALLOC - device allocated to other user
;ACG0414      OAA5 .653 ;         = SS$_NOPRIV - failed device protection
;ACG0414      OAA5 .654 ;         = SS$_TEMPLATEDEV - can't allocate template device
;ACG0414      OAA5 .655 ;         = SS$_DEV MOUNT - device already mounted
;ACG0414      OAA5 .656 ;         = SS$_DEV OFFLINE - device marked offline
;ACG0414      OAA5 .657 ;         R5 = UCB
;ACG0414      OAA5 .658 ;         R6 = DDB
;ACG0414      OAA5 .659 ;         R7 = system block
;ACG0414      OAA5 .660 ;         R10 - R4, R8 - R11 preserved
;ACG0414      OAA5 .661 ;
;ACG0414      OAA5 .662 ;         Note: If failure, R5 - R7 point to the last structures looked at.
;ACG0414      OAA5 .663 ;
;ACG0414      OAA5 .664 ; -
;ACG0414      OAA5 .665
;ACG0414      OAA5 .666 IOC$SEARCHCONT:
;ACG0414      OAA5 .667      031C 8F  BB  OAA5 .667      PUSHR      #AM<R2,R3,R4,R8,R9>      ; save registers
;ACG0414      OAA5 .668      05 5A  08  E5  OAA9 .668      BBCC      #IOC$V ALT,R10,10$      ; check if alternate UCB in use
;ACG0414      OAA5 .669      55  00A8 C5  D0  OAAD .669      MOVL      UCB$_DP_ALTUCB(R5),R5      ; link back to other to continue
;ACG0414      OAA5 .670      10$  OAB2 .670      10$  BRB      NEXTUCB      ; continue search

```

```

;ACG0414      OAB4 .672      .SBTTL  Check UCB Against Search Rules
;ACG0414      OAB4 .673
;ACG0414      OAB4 .674 ;+
;ACG0414      OAB4 .675 ;
;ACG0414      OAB4 .676 ; IOC$TESTUNIT - Check UCB Against Search Rules
;ACG0414      OAB4 .677 ;
;ACG0414      OAB4 .678 ; INPUTS:
;ACG0414      OAB4 .679 ;
;ACG0414      OAB4 .680 ;      R2 = unit number
;ACG0414      OAB4 .681 ;      R3 = device type code
;ACG0414      OAB4 .682 ;      R4 = PCB address
;ACG0414      OAB4 .683 ;      R5 = UCB address
;ACG0414      OAB4 .684 ;      R10 = flags
;ACG0414      OAB4 .685 ;      R11= address of lock value block
;ACG0414      OAB4 .686 ;
;ACG0414      OAB4 .687 ; OUTPUTS:
;ACG0414      OAB4 .688 ;
;ACG0414      OAB4 .689 ;      R0 = SS$_NORMAL - eligible for use according to flags
;ACG0414      OAB4 .690 ;      = SS$_NOSUCHDEV - wrong unit number
;ACG0414      OAB4 .691 ;      = SS$_DEVALLOC - device allocated to other user
;ACG0414      OAB4 .692 ;      = SS$_NOPRIV - failed device protection
;ACG0414      OAB4 .693 ;      = SS$_TEMPLATEDEV - can't allocate template device
;ACG0414      OAB4 .694 ;      = SS$_DEVMOUNT - device already mounted
;ACG0414      OAB4 .695 ;      = SS$_DEVOFFLINE - device marked offline
;ACG0414      OAB4 .696 ;
;ACG0414      OAB4 .697 ; -
;ACG0414      OAB4 .698
;ACG0414      OAB4 .699 IOC$TESTUNIT::
;ACG0414      50 0908 8F 3C OAB4 .700 MOVZWL #SS$_NOSUCHDEV,R0      ; assume wrong device
;ACG0414      06 5A E9 OAB9 .701 BLBC R10,I0$      ; branch if not physical search
;ACG0414      54 A5 52 B1 OABC .702 CMPW R2,UCB$W_UNIT(R5)      ; is the unit number exactly right?
;ACG0414      5B 12 OAC0 .703 BNEQ 80$      ; branch to error if not right.
;ACG0414      OAC2 .704
;ACG0414      09 5A 01 E1 OAC2 .705 10$: BBC #IOC$V_TYPE,R10,20$      ; branch if not searching for dev type
;ACG0414      00 ED OAC6 .706 CMPZV #MSCP$V_MTYN,-
;ACG0414      16 OAC8 .707 #MSCP$V_MTYN_D1,-
;ACG0414      53 008C C5 OAC9 .708 UCB$L_MEDIA_ID(R5),R3      ; is this the requested type?
;ACG0414      4E 12 OACD .709 BNEQ 80$      ; branch if not
;ACG0414      5A 10 88 OACF .710 20$: BISB #IOC$M_EXISTS,R10      ; note eligible device seen
;ACG0414      0A 3C A5 03 E1 OAD2 .711 BBC #DEV$V_CDP,UCB$L_DEVCHAR2(R5),30$ ; is this served path to a local d
;ACG0414      55 00A8 C5 D0 OAD7 .712 MOVL UCB$L_DP_ALTUCB(R5),R5      ; yes, get local path UCB address.
;ACG0414      5A 0100 8F A8 OADC .713 BISW #IOC$M_ALT,R10      ; note alternate UCB in use
;ACG0414      03 5A 06 E1 OAE1 .714 30$: BBC #IOC$V_ANY,R10,40$      ; if SEARCHALL, finish with success.
;ACG0414      0085 31 OAE5 .715 BRW 120$
;ACG0414      OAE8 .716 ;
;ACG0414      OAE8 .717 ; Check the device reference count and allocation status.
;ACG0414      OAE8 .718 ;
;ACG0414      50 006C 8F 3C OAE8 .719 40$: MOVZWL #SS$_DEVMOUNT,R0      ; check if device is already mounted
;ACG0414      7F 38 A5 13 E0 OAE8 .720 BBS #DEV$V_MNT,UCB$L_DEVCHAR(R5),130$
;ACG0414      50 0840 8F 3C OAF2 .721 MOVZWL #SS$_DEVALLOC,R0
;ACG0414      75 64 A5 09 E0 OAF7 .722 BBS #UCB$V_MOUNTING,UCB$W_STS(R5),130$ ; branch if mount in progress
;ACG0414      5C A5 B5 OAF8 .723 TSTW UCB$W_REFC(R5)      ; is reference count zero?
;ACG0414      1E 13 OAFF .724 BEQL 90$      ; branch if reference count is zero.
;ACG0414      0B 5A 07 E1 OB01 .725 BBC #IOC$V_MOUNT,R10,50$      ; if mounting...
;ACG0414      0A 5A 0A E0 OB05 .726 BBS #IOC$V_ALLOC,R10,60$      ; if shared mount
;ACG0414      11 38 A5 17 E1 OB09 .727 BBC #DEV$V_ALL,UCB$L_DEVCHAR(R5),90$ ; OK if not allocated
;ACG0414      08 11 OB0E .728 BRB 70$      ; otherwise check allocation

```

```

;ACG0414          OB10 .729
;ACG0414          5E 5A   E9  OB10 .730 50$:  BLBC   R10,130$           ; allocate: error if not phy
;ACG0414          59 38 A5 17  E1  OB13 .731 60$:  BBC    #DEV$V_ALL,UCB$L_DEVCHAR(R5),130$ ; err if dev. not allocated.
;ACG0414          60 A4   2C A5  D1  OB18 .732 70$:  Cmpl  UCB$L_PID(R5),PCB$L_PID(R4) ; does this process own the device?
;ACG0414          52 12  OB1D .733 80$:  BNEQ   130$           ; branch to error if not our device.
;ACG0414          OB1F .734 ;
;ACG0414          OB1F .735 ; Check all the other miscellaneous junk that can make a device not
;ACG0414          OB1F .736 ; available.
;ACG0414          OB1F .737 ;
;ACG0414          50 24  3C  OB1F .738 90$:  MOVZWL #SS$ NOPRIV,RO           ; check if device is spooled
;ACG0414          06 38 A5 06  E1  OB22 .739   BBC    #DEV$V_SPL,UCB$L_DEVCHAR(R5),100$ ; branch if not
;ACG0414          OB27 .740   IFNPRIV ALLSPOOL,130$,R4           ; else, process must have ALLSPOOL priv.
;ACG0414          50 0084 8F 3C  OB2D .741 100$: MOVZWL #SS$ DEVOFFLINE,RO           ; check if device is available
;ACG0414          3A 38 A5 12  E1  OB32 .742   BBC    #DEV$V_AVL,UCB$L_DEVCHAR(R5),130$
;ACG0414          35 64 A5 04  E1  OB37 .743   BBC    #UCB$V_ONLINE,UCB$W_STS(R5),130$
;ACG0414          50 21DC 8F 3C  OB3C .744   MOVZWL #SS$ TEMPLATEDEV,RO           ; check if device is a template
;ACG0414          2B 64 A5 0D  E0  OB41 .745   BBS    #UCB$V_TEMPLATE,UCB$W_STS(R5),130$
;ACG0414          F4B7' 30  OB46 .746   BSBW   EXE$CHKRDACCES           ; check device protection
;ACG0414          25 50  E9  OB49 .747   BLBC   R0,130$           ; out if not accessible
;ACG0414          OB4C .748 ;
;ACG0414          OB4C .749 ; We've passed all the local tests. Now try to take out the appropriate
;ACG0414          OB4C .750 ; lock on the device.
;ACG0414          OB4C .751 ;
;ACG0414          1C 3C A5 00  E1  OB4C .752   BBC    #DEV$V_CLU,UCB$L_DEVCHAR2(R5),120$ ; br. if not cluster visible
;ACG0414          17 38 A5 17  E0  OB51 .753   BBS    #DEV$V_ALL,UCB$L_DEVCHAR(R5),120$ ; br. if allocated
;ACG0414          51 5B  D0  OB56 .754   MOVL   R11,R1           ; value block address
;ACG0414          50 05  D0  OB59 .755   MOVL   #LCK$K_EXMODE,RO           ; assume exclusive lock
;ACG0414          07 5A 0A  E0  OB5C .756   BBS    #IOC$V_ALLOC,R10,110$           ; branch if allocation requested
;ACG0414          03 5A 07  E1  OB60 .757   BBC    #IOC$V_MOUNT,R10,110$           ; branch if not mount mode
;ACG0414          50 04  D0  OB64 .758   MOVL   #LCK$K_PWMODE,RO           ; which uses PW
;ACG0414          F496' 30  OB67 .759 110$: BSBW   IOC$LOCK_DEV           ; and try to take device lock
;ACG0414          04 50  E9  OB6A .760   BLBC   R0,130$
;ACG0414          50 01  D0  OB6D .761 120$: MOVL   #SS$_NORMAL,RO           ; indicate success
;ACG0414          05  OB70 .762   RSB
;ACG0414          OB71 .763 ;
;ACG0414          OB71 .764 ; To here on any error.
;ACG0414          OB71 .765 ;
;ACG0414          05 5A 08  E5  OB71 .766 130$: BBCC   #IOC$V_ALT,R10,140$           ; check if alternate UCB in use
;ACG0414          55 00A8 C5  D0  OB75 .767   MOVL   UCB$L_DP_ALTUCB(R5),R5           ; link back to other to continue
;ACG0414          05  OB7A .768 140$: RSB
;ACG0414          OB7B .769
;ACG0414          OB7B .770
;ACG0414          OB7B .771
;ACG0414          OB7B 1735   .END

```



\$\$BASE	=	00000001			DDB\$T_NAME	=	00000014		
\$\$DISPL	=	00000008			DDT\$L_CANCEL	=	0000000C		
\$\$GENSW	=	00000001			DDT\$L_REGDUMP	=	00000010		
\$\$HIGH	=	00000007			DDT\$L_START	=	00000000		
\$\$LIMIT	=	00000006			DDT\$L_UNITINIT	=	00000018		
\$\$LOW	=	00000001			DEALLOC_DESCRIP	=	000004BF	R	02
\$\$MNSW	=	00000001			DEV\$M_MBX	=	00100000		
\$\$MXSW	=	00000001			DEV\$M_TRM	=	00000004		
ADD_DOLLAR	=	000006BF	R	02	DEV\$V_2P	=	00000004		
ADD_NODE	=	000006B5	R	02	DEV\$V_ALL	=	00000017		
ADP\$C_NUMDATAP	=	00000010			DEV\$V_AVL	=	00000012		
ADP\$L_CSR	=	00000000			DEV\$V_CDP	=	00000003		
ADP\$L_DPQBL	=	00000018			DEV\$V_CLU	=	00000000		
ADP\$L_DPQFL	=	00000014			DEV\$V_FOD	=	0000000E		
ADP\$L_MRACTMDRS	=	0000005C			DEV\$V_MNT	=	00000013		
ADP\$L_MRQBL	=	00000034			DEV\$V_NNM	=	00000009		
ADP\$L_MRQFL	=	00000030			DEV\$V_OPR	=	00000007		
ADP\$W_ADPTYPE	=	0000000E			DEV\$V_SPL	=	00000006		
ADP\$W_DPBITMAP	=	00000060			DEV\$V_TRM	=	00000002		
ADP\$W_MRFREGARY	=	0000015E			DIR...	=	00000001		
ADP\$W_MRNREGARY	=	00000064			DISKCHK	=	00000191	R	02
ALLOC_DESCRIP	=	000004D8	R	02	DISPLAY_NAME	=	0000069F	R	02
ALLOC_NAME	=	00000682	R	02	DO_PMS	=	000001A9	R	02
AT\$_UBA	=	00000001			DYN\$C_TWP	=	00000030		
BINNUM	=	00000000			DYN\$C_UCB	=	00000010		
BOO\$GL_SPTFREL	=	*****	X	02	EMB\$B_DV_ERTCNT	=	00000010		
BOO\$GL_SPTFREL	=	*****	X	02	EMB\$Q_DV_IOSB	=	00000012		
BUG\$_INCONSTATE	=	*****	X	02	EMB\$W_DV_STS	=	0000001A		
BUG\$_IVBYTEALGN	=	*****	X	02	END_BROADCAST	=	0000078B	R	02
BUG\$_UNSUPRTCPU	=	*****	X	02	ERL\$RELEASEMB	=	*****	X	02
CAN\$C_AMBXDGN	=	00000002			EXDVNM	=	000006D8	R	02
CAN\$C_DASSGN	=	00000001			EXE\$ALONONPAGED	=	*****	X	02
CDRPS\$L_BCNT	=	FFFFFFFFD2			EXE\$ALTQUEPKT	=	*****	X	02
CDRPS\$L_FPC	=	0000000C			EXE\$CHKRDACCES	=	*****	X	02
CDRPS\$L_FQFL	=	00000000			EXE\$DEANONPAGED	=	*****	X	02
CDRPS\$L_FR3	=	00000010			EXE\$GB_CPUTYPE	=	*****	X	02
CDRPS\$L_IOQFL	=	FFFFFFFA0			EXE\$GL_ABSTIM	=	*****	X	02
CDRPS\$L_RWCPTTR	=	00000028			EXE\$GQ_SYSTIME	=	*****	X	02
CDRPS\$L_UBARSRCE	=	0000003C			EXE\$MOUNTVER	=	*****	X	02
CDRPS\$W_BOFF	=	FFFFFFFD0			EXE\$TEST_CSR	=	*****	X	02
CLU\$GL_CLUB	=	*****	X	02	FULL_NAME	=	0000067D	R	02
COM\$DRVDEALMEM	=	*****	X	02	GETNUMBER	=	00000958	R	02
COMMON_ALOUBAMAP	=	00000366	R	02	IDB\$L_ADP	=	00000014		
CRB\$B_MASK	=	0000000E			IDB\$L_CSR	=	00000000		
CRB\$L_INTD	=	00000024			IDB\$L_OWNER	=	00000004		
CRB\$L_LINK	=	00000020			IOC\$ALLOSPT	=	00000620	RG	02
CRB\$L_WQBL	=	00000004			IOC\$ALODATAP	=	00000261	R	02
CRB\$L_WQFL	=	00000000			IOC\$ALOMAPUDA	=	00000314	R	02
CRB\$M_BSY	=	00000001			IOC\$ALOUBAMAP	=	0000033E	RG	02
CRB\$V_BSY	=	00000000			IOC\$ALOUBAMAPN	=	00000337	RG	02
DC\$_DISK	=	00000001			IOC\$ALOUBAMAPSP	=	000003A8	RG	02
DDB\$L_2P_UCB	=	00000040			IOC\$ALOUBMAPRM	=	0000044E	RG	02
DDB\$L_ALLOCLS	=	0000003C			IOC\$ALOUBMAPRMN	=	00000447	RG	02
DDB\$L_DP_UCB	=	00000040			IOC\$ALTREQCOM	=	00000111	RG	02
DDB\$L_LINK	=	00000000			IOC\$BROADCAST	=	00000720	RG	02
DDB\$L_SB	=	00000034			IOC\$CANCELIO	=	00000000	RG	02
DDB\$L_UCB	=	00000004			IOC\$CREDIT_UCB	=	*****	X	02

IOC\$CTRLINIT	00000830	RG	02	IRP\$L_MEDIA	=	00000038		
IOC\$CVT_DEVNAM	00000647	RG	02	IRP\$L_PID	=	0000000C		
IOC\$DALOCUBAMAP	00000568	R	02	IRP\$L_SVAPTE	=	0000002C		
IOC\$DELETE_UCB	*****	X	02	IRP\$L_UCB	=	0000001C		
IOC\$DIAGBUFILL	00000054	RG	02	IRP\$V_DIAGBUF	=	00000007		
IOC\$GL_PSBL	*****	X	02	IRP\$W_CHAN	=	00000028		
IOC\$INITIATE	000001D4	RG	02	IRP\$W_STS	=	0000002A		
IOC\$LAST_CHAN	0000001E	RG	02	LCK\$K_EXMODE	=	00000005		
IOC\$LAST_CHAN_AMBX	00000017	RG	02	LCK\$K_PWMODE	=	00000004		
IOC\$LOCK_DEV	*****	X	02	LOCAL_NAME		000006C1	R	02
IOC\$MNTVER	000001CB	RG	02	MMG\$GL_SPTBASE	*****		X	02
IOC\$M_2P	= 00000020			MNTVERPNDCHK		000001B1	R	02
IOC\$M_ALT	= 00000100			MSCP\$V_MTYD_D1	=	00000016		
IOC\$M_CLASS	= 00000004			MSCP\$V_MTYD_N	=	00000000		
IOC\$M_EXISTS	= 00000010			NEXTUCB		00000A79	R	02
IOC\$M_LOCAL	= 00000008			NO_SECONDARY		000006E0	R	02
IOC\$M_PHY	= 00000001			NXTIRP		00000182	R	02
IOC\$M_TYPE	= 00000002			PCB\$L_PID	=	00000060		
IOC\$PARSDEVNAM	0000089D	RG	02	PCB\$Q_PRIV	=	00000084		
IOC\$RELCHAN	00000083	RG	02	PDT\$L_ADP	=	000000E0		
IOC\$RELDATAP	0000028C	RG	02	PMS\$END_IO	*****		X	02
IOC\$RELDATAPUDA	00000281	RG	02	PMS\$GL_IOPFMPDB	*****		X	02
IOC\$RELMAPREG	00000513	RG	02	PMS\$START_IO	*****		X	02
IOC\$RELMAPUDA	000004F8	RG	02	PMS\$END		00000173	R	02
IOC\$RELSCHAN	00000079	RG	02	PR\$_IPL	=	00000012		
IOC\$REQCOM	0000013C	RG	02	PR\$_SID_TYP730	=	00000003		
IOC\$REQDATAP	00000201	RG	02	PR\$_SID_TYP750	=	00000002		
IOC\$REQDATAPNW	00000213	RG	02	PR\$_SID_TYP780	=	00000001		
IOC\$REQDATAPUDA	00000221	RG	02	PR\$_SID_TYP790	=	00000004		
IOC\$REQMAPREG	00000302	RG	02	PR\$_SID_TYP8NN	=	00000006		
IOC\$REQMAPUDA	000002ED	RG	02	PR\$_SID_TYP8SS	=	00000005		
IOC\$REQPCHANH	000000DA	RG	02	PR\$_SID_TYPUV1	=	00000007		
IOC\$REQPCHANL	000000E3	RG	02	PR\$_SIRR	=	00000014		
IOC\$REQSCHANH	000000C6	RG	02	PRV\$V_ALLSPOOL	=	00000004		
IOC\$REQSCHANL	000000D0	RG	02	PUTASCIC		000006FD	R	02
IOC\$RETURN	000005D9	RG	02	PUTCHAR		0000070E	R	02
IOC\$SCAN_IODB	00000791	RG	02	PUTDOLLAR		0000070B	R	02
IOC\$SCAN_IODB_2P	000007D6	RG	02	PUTNUM		000006E5	R	02
IOC\$SEARCHCONT	00000AA5	RG	02	REALLOC_CD_MAPREGS		0000055A	R	02
IOC\$SEARCHINT	000009B1	RG	02	RELDATAP_COMMON		00000298	R	02
IOC\$TESTUNIT	00000AB4	RG	02	RELEASE		0000018E	R	02
IOC\$UNITINIT	0000086A	RG	02	RESR0		00000008		
IOC\$V_2P	= 00000005			RESR1		0000000C		
IOC\$V_ALLOC	= 0000000A			RESR2		00000010		
IOC\$V_ALT	= 00000008			RESR3		00000014		
IOC\$V_ANY	= 00000006			RESR4		00000018		
IOC\$V_CLASS	= 00000002			SAVABS...	=	0000001C		
IOC\$V_EXISTS	= 00000004			SAVED_R0	=	00000000		
IOC\$V_MOUNT	= 00000007			SAVED_R1	=	00000004		
IOC\$V_TYPE	= 00000001			SAVED_R2	=	00000008		
IOC\$WFIKPC	000005DA	RG	02	SAVED_R3	=	0000000C		
IOC\$WFIRLCH	000005FC	RG	02	SAVED_R4	=	00000010		
IPL\$_ASTDEL	= 00000002			SAVED_R5	=	00000014		
IPL\$_IOPOST	= 00000004			SAVR2	=	00000000		
IPL\$_QUEUEAST	= 00000006			SAVR3	=	00000004		
IRP\$L_DIAGBUF	= 0000004C			SAVR4	=	00000008		
IRP\$L_IOQFL	= 00000000			SAVR8	=	0000000C		

SAVR9	=	00000010	UCB\$L_OPCNT	=	00000070
SB\$L_DDB	=	00000054	UCB\$L_PID	=	0000002C
SB\$L_FLINK	=	00000000	UCB\$L_STS	=	00000064
SB\$T_NODENAME	=	00000044	UCB\$L_SVAPTE	=	00000078
SCH\$GL_CURPCB	*****	X 02	UCB\$M_BSY	=	00000100
SCRLEN	00000010		UCB\$M_CANCEL	=	00000008
SCS\$GA_LOCALSB	*****	X 02	UCB\$M_INT	=	00000002
SCS\$GQ_CONFIG	*****	X 02	UCB\$M_TIM	=	00000001
SCS\$RESUMEWAITR	*****	X 02	UCB\$M_TIMEOUT	=	00000040
SECONDARY_NAME	0000068F	R 02	UCB\$V_BSY	=	00000008
SS\$_BUFFEROVF	=	00000601	UCB\$V_DELETEUCB	=	00000010
SS\$_DEVALLOC	=	00000840	UCB\$V_ERLOGIP	=	00000002
SS\$_DEVMOUNT	=	0000006C	UCB\$V_MNTVERIP	=	0000000E
SS\$_DEVOFFLINE	=	00000084	UCB\$V_MNTVERPND	=	00000013
SS\$_ILLIOFUNC	=	000000F4	UCB\$V_MOUNTING	=	00000009
SS\$_INSMEM	=	00000124	UCB\$V_ONLINE	=	00000004
SS\$_IVDEVNAM	=	00000144	UCB\$V_TEMPLATE	=	0000000D
SS\$_NODEVAVL	=	000009B0	UCB\$W_BCNT	=	0000007E
SS\$_NOPRIV	=	00000024	UCB\$W_BOFF	=	0000007C
SS\$_NORMAL	=	00000001	UCB\$W_REFC	=	0000005C
SS\$_NOSUCHDEV	=	00000908	UCB\$W_STS	=	00000064
SS\$_TEMPLATEDEV	=	000021DC	UCB\$W_UNIT	=	00000054
TTY\$B_WB_FIPL	=	0000000B	VEC\$B_DATAPATH	=	00000013
TTY\$B_WB_TYPE	=	0000000A	VEC\$B_NUMREG	=	00000012
TTY\$K_WB_LENGTH	=	00000030	VEC\$L_ADP	=	00000014
TTY\$L_WB_DATA	=	00000030	VEC\$L_IDB	=	00000008
TTY\$L_WB_END	=	00000020	VEC\$L_INITIAL	=	0000000C
TTY\$L_WB_FR3	=	00000010	VEC\$L_UNITINIT	=	00000018
TTY\$L_WB_IRP	=	00000024	VEC\$M_MAPLOCK	=	00008000
TTY\$L_WB_NEXT	=	0000001C	VEC\$S_DATAPATH	=	00000005
TTY\$L_WB_RETADDR	=	0000002C	VEC\$V_DATAPATH	=	00000000
TTY\$W_WB_SIZE	=	00000008	VEC\$V_MAPLOCK	=	0000000F
UBMD\$B_DATAPATH	=	00000003	VEC\$V_PATHLOCK	=	00000007
UBMD\$B_NUMREG	=	00000002	VEC\$W_MAPREG	=	00000010
UBMD\$W_MAPREG	=	00000000			
UCB\$B_DEVCLASS	=	00000040			
UCB\$B_ERTCNT	=	00000080			
UCB\$B_FIPL	=	0000000B			
UCB\$B_TYPE	=	0000000A			
UCB\$L_2P_LINK	=	000000A4			
UCB\$L_CRB	=	00000024			
UCB\$L_DDB	=	00000028			
UCB\$L_DDT	=	00000088			
UCB\$L_DEVCHAR	=	00000038			
UCB\$L_DEVCHAR2	=	0000003C			
UCB\$L_DP_ALTUCB	=	000000A8			
UCB\$L_DP_DDB	=	000000A0			
UCB\$L_DP_LINK	=	000000A4			
UCB\$L_DUETIM	=	0000006C			
UCB\$L_EMB	=	00000094			
UCB\$L_FPC	=	0000000C			
UCB\$L_FQFL	=	00000000			
UCB\$L_FR3	=	00000010			
UCB\$L_IOQFL	=	0000004C			
UCB\$L_IRP	=	00000058			
UCB\$L_LINK	=	00000030			
UCB\$L_MEDIA_ID	=	0000008C			

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

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE					
\$ABS\$	0000001C ( 28.)	01 ( 1.)	NOPIC USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					
WIONONPAGED	00000B7B, ( 2939.)	02 ( 2.)	NOPIC USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.05	00:00:03.03
Command processing	73	00:00:00.67	00:00:05.51
Pass 1	676	00:00:36.81	00:02:30.72
Symbol table sort	0	00:00:04.17	00:00:16.52
Pass 2	408	00:00:12.25	00:00:52.03
Symbol table output	0	00:00:00.24	00:00:00.47
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1170	00:00:54.21	00:03:48.30

The working set limit was 2250 pages.  
176206 bytes (345 pages) of virtual memory were used to buffer the intermediate code.  
There were 150 pages of symbol table space allocated to hold 2680 non-local and 162 local symbols.  
2794 source lines were read in Pass 1, producing 24 object records in Pass 2.  
59 pages of virtual memory were used to define 55 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	35
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	47

2923 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:IOSUBNPAG/OBJ=OBJ\$:IOSUBNPAG MSRC\$:IOSUBNPAG/UPDATE=(ENH\$:IOSUBNPAG)+EXECML\$/LIB

**FORKCNTRL**

(1)	66.1	CREATE FORK-AND-WAIT PROCESS
(1)	67	CREATE I/O DRIVER FORK PROCESS
(2)	90	CREATE FORK PROCESS
(3)	117	SOFTWARE INTERRUPT ENTRY POINTS
(4)	165	SOFTWARE INTERRUPT FORK DISPATCHER

;DWT0158  
-1

```

0000 1 .TITLE FORKCNTRL - FORK PROCESS CONTROL
0000 .1 .IDENT 'V03-002'
0000 3
0000 4 ;
0000 5 ;*****
0000 6 ;*
0000 7 ;* COPYRIGHT (c) 1978, 1980, 1982 BY
0000 8 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 ;* ALL RIGHTS RESERVED.
0000 10 ;*
0000 11 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 ;* TRANSFERRED.
0000 17 ;*
0000 18 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 ;* CORPORATION.
0000 21 ;*
0000 22 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 ;*
0000 25 ;*
0000 26 ;*****
0000 27 ;
0000 28 ; D. N. CUTLER 9-AUG-76
0000 29 ;
0000 .1 ; MODIFIED BY:
0000 .2 ;
0000 .3 ; V03-002 DWT0158 David W. Thiel 27-DEC-1983
0000 .4 ; Speed up fork processing a bit by doing fewer interrupts
0000 .5 ; and REMQUE's.
0000 .6 ;
0000 .7 ; V03-001 ROW0176 Ralph O. Weber 4-APR-1983
0000 .8 ; Add EXE$FORK_WAIT, the entry fork-and-wait routine which is
0000 .9 ; the object of the FORK_WAIT macro.
0000 .10 ;
0000 59 ;
0000 60 ; MACRO LIBRARY CALLS
0000 61 ;
0000 62 ;
0000 63 $FKBDEF ;DEFINE FKB OFFSETS
0000 64 $PRDEF ;DEFINE PROCESSOR REGISTERS
0000 65 $UCBDEF ;DEFINE UCB OFFSETS

```

;ROW0176  
;ROW0176  
;DWT0158  
;DWT0158  
;DWT0158  
;DWT0158  
;ROW0176  
;ROW0176  
;ROW0176  
;ROW0176  
;ROW0176  
-29





```
000E 67          .SBTTL  CREATE I/O DRIVER FORK PROCESS
000E 68 ;+
000E 69 ; EXE$IOFORK - CREATE I/O DRIVER FORK PROCESS
000E 70 ;
000E 71 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO CREATE A FORK PROCESS.
000E 72 ;
000E 73 ; INPUTS:
000E 74 ;
000E 75 ;         00(SP) = RETURN ADDRESS OF CALLER.
000E 76 ;         04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
000E 77 ;
000E 78 ;         R5 = UCB ADDRESS OF DEVICE UNIT.
000E 79 ;
000E 80 ; OUTPUTS:
000E 81 ;
000E 82 ;         ***TBS***
000E 83 ;-
000E 84
00000000 85          .PSECT  A$EXENONPAGED, LONG
0000 86 EXE$IOFORK:; CREATE I/O DRIVER FORK PROCESS
64 A5 01 AA 0000 87          BICW   #UCB$M_TIM,UCB$W_STS(R5);DISABLE TIMEOUT AND FALL THROUGH
0004 88          ;TO EXE$FORK
```

```

0004 90 .SBTTL CREATE FORK PROCESS
0004 91 ;+
0004 92 ; EXE$FORK - CREATE FORK PROCESS
0004 93 ;
0004 94 ; THIS ROUTINE IS CALLED TO CREATE A FORK PROCESS.
0004 95 ;
0004 96 ; INPUTS:
0004 97 ;
0004 98 ; 00(SP) = RETURN ADDRESS OF CALLER.
0004 99 ; 04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
0004 100 ;
0004 101 ; R5 = ADDRESS OF FORK BLOCK.
0004 102 ;
0004 103 ; OUTPUTS:
0004 104 ;
0004 105 ; ***TBS***
0004 106 ;-
0004 107
0004 108 EXE$FORK:: ;CREATE FORK PROCESS
10 A5 53 7D 0004 109 MOVQ R3,FKB$L_FR3(R5) ;SAVE REGISTERS R3 AND R4
OC A5 8ED0 0008 110 POPL FKB$L_FPC(R5) ;SET FORK PROCESS PC
54 0B A5 9A 000C 111 MOVZBL FKB$B_FIPL(R5),R4 ;GET FORK IPL
53 FFDD'CF44 7E 0010 112 MOVAQ W^SWI$GL_FQFL-<6*8>[R4],R3 ;GET ADDRESS OF FORK QUEUE LISTHEAD
04 B3 65 0E 0016 113 INSQUE (R5),@4(R3) ;INSERT FORK BLOCK IN FORK QUEUE
03 12 001A .1 BNEQ 10$ ;IF QUEUE ALREADY POPULATED AVOID EXTRA INTE
001C .2 SOFTINT R4 ;INITIATE SOFTWARE INTERRUPT
05 001F .3 10$: RSB ; AND RETURN
;DWT0158
;DWT0158
;DWT0158

```

```

0020 117          .SBTTL  SOFTWARE INTERRUPT ENTRY POINTS
0020 118 ;+
0020 119 ; THE APPROPRIATE ENTRY POINT IS AUTOMATICALLY VECTORED TO WHEN THE SOFTWARE
0020 120 ; INTERRUPT PRIORITY ARBITRATION LOGIC IN THE CENTRAL PROCESSOR DETECTS A
0020 121 ; PENDING INTERRUPT AT LEVEL 6, 8, 9, 10, OR 11 AND THE CURRENT PRIORITY LEVEL
0020 122 ; IS LOWER THAN THE PENDING LEVEL. THE STATE OF THE STACK ON ENTRY IS:
0020 123 ;
0020 124 ;          00(SP) = INTERRUPT PC.
0020 125 ;          04(SP) = INTERRUPT PSL.
0020 126 ;
0020 127 ; THERE IS AN ENTRY POINT FOR EACH FORK LEVEL SO THAT THE HARDWARE IPL REGISTER
0020 128 ; WILL NOT HAVE TO BE READ TO DETERMINE THE CURRENT PROCESSOR IPL.  READING THE
0020 129 ; IPL REGISTER IS SLOW ENOUGH TO WARRANT THE ADDITIONAL CODE.
0020 130 ;-
0020 131
0020 132          .ALIGN  LONG                ;ENTRY POINT MUST BE LONGWORD ALIGNED
0020 133 EXE$FRKIPL6DSP::                ;FORK IPL 6 ENTRY POINT
56      56      DD      0020 134          PUSHL  R6                ;SAVE R6.
56      0000'CF  7E      0022 135          MOVAQ  W^SWI$GL_FQFL,R6        ;GET ADDRESS OF FORK QUEUE LISTHEAD
56      2F      11      0027 136          BRB    EXE$FORKDSPTH        ;BRANCH TO COMMON CODE
0029 137
0029 138          .ALIGN  LONG                ;ENTRY POINT MUST BE LONGWORD ALIGNED
002C 139 EXE$FRKIPL9DSP::                ;FORK IPL 9 ENTRY POINT
56      56      DD      002C 140          PUSHL  R6                ;SAVE R6.
56      0018'CF  7E      002E 141          MOVAQ  W^SWI$GL_FQFL+24,R6    ;GET ADDRESS OF FORK QUEUE LISTHEAD
56      23      11      0033 142          BRB    EXE$FORKDSPTH        ;BRANCH TO COMMON CODE
0035 143
0035 144          .ALIGN  LONG                ;ENTRY POINT MUST BE LONGWORD ALIGNED
0038 145 EXE$FRKIPL10DSP::               ;FORK IPL 10 ENTRY POINT
56      56      DD      0038 146          PUSHL  R6                ;SAVE R6.
56      0020'CF  7E      003A 147          MOVAQ  W^SWI$GL_FQFL+32,R6    ;GET ADDRESS OF FORK QUEUE LISTHEAD
56      17      11      003F 148          BRB    EXE$FORKDSPTH        ;BRANCH TO COMMON CODE
0041 149
0041 150          .ALIGN  LONG                ;ENTRY POINT MUST BE LONGWORD ALIGNED
0044 151 EXE$FRKIPL11DSP::               ;FORK IPL 11 ENTRY POINT
56      56      DD      0044 152          PUSHL  R6                ;SAVE R6.
56      0028'CF  7E      0046 153          MOVAQ  W^SWI$GL_FQFL+40,R6    ;GET ADDRESS OF FORK QUEUE LISTHEAD
56      0B      11      004B 154          BRB    EXE$FORKDSPTH        ;BRANCH TO COMMON CODE
004D 155
004D 156          .ALIGN  LONG                ;ENTRY POINT MUST BE LONGWORD ALIGNED
0050 157 EXE$FRKIPL8DSP::                ;FORK IPL 8 ENTRY POINT
56      56      DD      0050 158          PUSHL  R6                ;SAVE R6.
56      0010'CF  7E      0052 159          MOVAQ  W^SWI$GL_FQFL+16,R6    ;GET ADDRESS OF FORK QUEUE LISTHEAD
56      01      01      0057 160          NOP                    ;PAD OUT TO LONGWORD BOUNDARY
0058 161
0058 162          ; DROP THROUGH TO COMMON CODE
0058 163          ;

```



FORKCNTRL  
Symbol table

- FORK PROCESS CONTROL

3-JUN-1984 11:20:55 VAX-11 Macro V03-01 Page 7  
12-MAR-1982 17:11:47 DISK\$VMSMASTER:[SYS.SRC]FORKCNTRL.(4)

```

EXE$FORK          00000004 RG    03
EXE$FORKDSPH     00000058 RG    03
EXE$FORK WAIT    00000000 RG    01
EXE$FRKIPL10DSP  00000038 RG    03
EXE$FRKIPL11DSP  00000044 RG    03
EXE$FRKIPL6DSP   00000020 RG    03
EXE$FRKIPL8DSP   00000050 RG    03
EXE$FRKIPL9DSP   0000002C RG    03
EXE$GL_FKWAITBL  ***** X    01
EXE$IOfORK       00000000 RG    03
FKB$B_FIPL       = 0000000B
FKB$L_FPC        = 0000000C
FKB$L_FR3        = 00000010
FKB$L_FR4        = 00000014
PR$ SIRR         = 00000014
SWI$GL_FQFL      ***** X    03
UCB$M_TIM        = 00000001
UCB$W_STS        = 00000064
  
```

+-----+  
! 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 .	0000000E ( 14.)	01 ( 1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$AB\$\$	00000000 ( 0.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
A\$EXENONPAGED	00000081 ( 129.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.07	00:00:00.47
Command processing	73	00:00:00.58	00:00:06.58
Pass 1	200	00:00:04.79	00:00:24.27
Symbol table sort	0	00:00:00.68	00:00:02.39
Pass 2	51	00:00:01.32	00:00:07.47
Symbol table output	3	00:00:00.04	00:00:00.03
Psect synopsis output	2	00:00:00.02	00:00:00.10
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	341	00:00:07.50	00:00:41.34

The working set limit was 900 pages.  
22659 bytes (45 pages) of virtual memory were used to buffer the intermediate code.  
There were 30 pages of symbol table space allocated to hold 426 non-local and 4 local symbols.  
244 source lines were read in Pass 1, producing 14 object records in Pass 2.  
12 pages of virtual memory were used to define 11 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	3
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	8

491 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:FORKCNTRL/OBJ=OBJ\$:FORKCNTRL MSRC\$:FORKCNTRL/UPDATE=(ENH\$:FORKCNTRL)+EXECML\$/LIB

**INITADP**

(3)	168	Macros to describe nexus configurations
(4)	220.1	Adapter-specific data structures
(5)	378	CPU-specific data structures
(6)	528	Message strings
(7)	539	INI\$IOMAP, Initialize and map nexuses
(8)	691	INITADP_780, _750, _730, and _UV1
(9)	708	CONFIG_IOSPACE
(10)	832	CREATE_ARRAYS
(11)	867	MAP_PAGES
(13)	1016	INI\$SUBSPACE
(14)	1072	INI\$UBADP - BUILD ADP AND INITIALIZE UBA
(14)	1503	INI\$MBADP - BUILD ADP AND INITIALIZE MBA
(14)	1504	INI\$DRADP - BUILD ADP AND INITIALIZE DR32
(14)	1505	INI\$CIADP - BUILD ADP AND INITIALIZE CI
(14)	1649.36	INI\$KDZ11
(14)	1650	INI\$CONSOLE, init data structures for console
(15)	1762	EXE\$INI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
(16)	1905	EXE\$INIT_TODR - SET SYSTEM TIME TO CORRECT VALUE AT STARTUP



```
0000 1 .NLIST CND
0000 3 .TITLE INIADP780 - ADAPTER INITIALIZATION FOR VAX 11/780
0000 5
0000 9
0000 13
0000 17
0000 21
;RLRSCORPIO 0000 .4
;RLRSCORPIO 0000 .5 .IDENT 'V03-021'
-1 0000 23 ;
0000 24 ;*****
0000 25 ;*
0000 26 ;* COPYRIGHT (c) 1978, 1980, 1982, 1983 BY *
0000 27 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 28 ;* ALL RIGHTS RESERVED. *
0000 29 ;*
0000 30 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 31 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 32 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 33 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 34 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 35 ;* TRANSFERRED. *
0000 36 ;*
0000 37 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 38 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 39 ;* CORPORATION. *
0000 40 ;*
0000 41 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 42 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 43 ;*
0000 44 ;*
0000 45 ;*****
0000 46 ;
0000 47 ; Facility: System bootstrapping and initialization
0000 48 ;
0000 49 ; Abstract: This module contains initialization routines that are loaded
0000 50 ; during system initialization (rather than linked into the system).
0000 51 ;
0000 52 ; Environment: Mode = KERNEL, Executing on INTERRUPT stack, IPL=31
0000 53 ;
0000 54 ; Author: Trudy C. Matthews Creation date: 22-Jan-1981
0000 55 ;
0000 56 ; Modification history:
0000 57 ;
;KDM0100 0000 .1 ; V03-021 KDM0100 Kathleen D. Morse 01-May-1984
;KDM0100 0000 .2 ; Correct address of memory CSRs to be past the 8 missing
;KDM0100 0000 .3 ; Qbus adapter pages that do not exist.
;KDM0100 0000 .4 ;
;KDM0099 0000 .5 ; V03-020 KDM0099 Kathleen D. Morse 27-Apr-1984
;KDM0099 0000 .6 ; On a MicroVAX I, if the sysgen parameter TIMEDWAIT is set
;KDM0099 0000 .7 ; to request no time-prompting, then use the last recorded
;KDM0099 0000 .8 ; system time instead. This is found in EXE$GQ_TODCBASE
;KDM0099 0000 .9 ; which can be updated with a SET TIME command.
;KDM0099 0000 .10 ;
;RLRSCORPIO 0000 .11 ; V03-019 RLRSCORPIO Robert L. Rappaport 16-Mar-1984
;RLRSCORPIO 0000 .12 ; Begin additions (to INI$IOMAP) for Scorpio support.
;RLRSCORPIO 0000 .13 ; Also move ADAPDESC to SYSMAR.MAR, changing it to remove
```

```

;RLRSCORPIO      0000 .14 ;           the ADAP_GENERAL array.
;RLRSCORPIO      0000 .15 ;
;RLRSCORPIO      0000 .16 ;           V03-018 RLRINIADP      Robert Rappaport      28-Feb-1984
;RLRSCORPIO      0000 .17 ;           Add refinements to previous update that introduces
;RLRSCORPIO      0000 .18 ;           longword array CONFREG. Mainly add logic to allow for
;RLRSCORPIO      0000 .19 ;           independently assembled invocations of ADAPDESC macro
;RLRSCORPIO      0000 .20 ;           to be linked into this code. This provides possible
;RLRSCORPIO      0000 .21 ;           support of BI as a public bus, with user defined nodes.
;RLRSCORPIO      0000 .22 ;
;KPL0100         0000 .23 ;           V03-017 KPL0100      Peter Lieberwirth      30-Jan-1984
;KPL0100         0000 .24 ;           Implement first step towards a longword-array CONFREG to
;KPL0100         0000 .25 ;           replace current byte array CONFREG. INIADP will construct
;KPL0100         0000 .26 ;           two confregs, CONFREG and CONFREGL. CONFREGL will be
;KPL0100         0000 .27 ;           a longword array. The high byte will be a VMS-bus
;KPL0100         0000 .28 ;           designation, and the low word will contain the 16-bit
;KPL0100         0000 .29 ;           device type. The BI introduces 16 bit device types.
;KPL0100         0000 .30 ;
;KPL0100         0000 .31 ;           When all references to CONFREG have been modified to touch
;KPL0100         0000 .32 ;           CONFREGL, INIADP will be modified again to stop creating
;KPL0100         0000 .33 ;           the byte array.
;KPL0100         0000 .34 ;
;KPL0100         0000 .35 ;           While here, map 9 pages of CI register space, up from 8.
;KPL0100         0000 .36 ;
;KPL0001         0000 .37 ;           V03-016 KPL0001      Peter Lieberwirth      17-Jan-1984
;KPL0001         0000 .38 ;           Fix bug in V03-015 that caused a failure to boot on 750s.
;KPL0001         0000 .39 ;           Specifically, add NDT$_MEM1664NI to ADAPDESC macro.
;KPL0001         0000 .40 ;
;TCM0009         0000 .41 ;           V03-015 TCM0009      Trudy C. Matthews      12-Dec-1983
;TCM0009         0000 .42 ;           Add support for booting from VENUS console device to
;TCM0009         0000 .43 ;           INI$CONSOLE. When mapping I/O space on VENUS, use the
;TCM0009         0000 .44 ;           PAMM to determine if any adaptors are present on the
;TCM0009         0000 .45 ;           ABUS.
;TCM0009         0000 .46 ;
;                 0000 58 ;           V03-014 KDM0081      Kathleen D. Morse      13-Sep-1983
;                 0000 59 ;           Create version for Micro-VAX I.
;                 0000 60 ;
;                 0000 61 ;           V03-013 DWT0126      David W. Thiel         30-Aug-1983
;                 0000 62 ;           Modify EXE$INIT_TODR to set internal time without
;                 0000 63 ;           modifying the contents of the system disk.
;                 0000 64 ;
;                 0000 65 ;           V03-012 KDM0062      Kathleen D. Morse      18-Jul-1983
;                 0000 66 ;           Add loadable, cpu-dependent routine for initializing
;                 0000 67 ;           the time-wait loop data cells, EXE$INI_TIMWAIT.
;                 0000 68 ;
;                 0000 69 ;           V03-011 KDM0057      Kathleen D. Morse      15-Jul-1983
;                 0000 70 ;           Added loadable, cpu-dependent routine for initializing
;                 0000 71 ;           the system time, EXE$INIT_TODR.
;                 0000 72 ;
;                 0000 73 ;           V03-010 KTA3071      Kerbey T. Altmann     12-Jul-1983
;                 0000 74 ;           Include CPU-specific console init code.
;                 0000 75 ;
;                 0000 76 ;           V03-009 TCM0008      Trudy C. Matthews     10-Jan-1983
;                 0000 77 ;           Change PSECT of 11/790 data that must stick around after
;                 0000 78 ;           INIADP is deleted. Build arrays ABUS_VA, ABUS_TYPE, and
;                 0000 79 ;           ABUS_INDEX that describe the 11/790 ABUS configuration.
;                 0000 80 ;
;                 0000 81 ;           V03-008 MSH0002      Maryann Hinden        08-Dec-1982

```

```
0000 82 ;          Add powerfail support for DW750.
0000 83 ;
0000 84 ;          V03-007 ROW0142      Ralph O. Weber      24-NOV-1982
0000 85 ;          Change UBA interrupt services routines prototype so that
0000 86 ;          UBAERRADR is correctly computed as an offset from UBAINTBASE.
0000 87 ;
0000 88 ;          V03-006 TCM0007      Trudy C. Matthews    10-Nov-1982
0000 89 ;          Add 11/790-specific initialization of SCB.
0000 90 ;
0000 91 ;          V03-005 TCM0006      Trudy C. Matthews    8-Nov-1982
0000 92 ;          Initialize field ADP$L_AVECTOR with the address of
0000 93 ;          each adapter's first SCB vector.
0000 94 ;
0000 95 ;          V03-004 KTA3018      Kerbey T. Altmann    30-Oct-1982
0000 96 ;          Move from INILOA facility, rename from INITADP,
0000 97 ;          put in conditional assembly, rewrite some routines.
0000 98 ;
0000 99 ;          V03-003 MSH0001      Maryann Hinden     24-Sep-1982
0000 100 ;         Change EXE$DW780_INT to EXE$UBAERR_INT.
0000 101 ;
0000 102 ;         V03-002 TCM0005      Trudy C. Matthews    10-Aug-1982
0000 103 ;         Added support for 11/790 processor.
0000 104 ;
0000 105 ;         V03-001 KDM0002      Kathleen D. Morse   28-Jun-1982
0000 106 ;         Added $DCDEF.
0000 107 ;
0000 108 ;--
```

```
0000 110 ;
0000 111 ; MACRO LIBRARY CALLS
0000 112 ;
;RLRSCORPIO 0000 113 $ADPDEF ; Define ADP offsets.
;RLRSCORPIO 0000 .1 $BIICDEF ; Define BIIC offsets.
;RLRSCORPIO 0000 .2 $BTDDDEF ; Define boot devices
-1 0000 .3 $BUADEF ; Define BUA Register offsets.
0000 115 $CRBDEF ; Define CRB offsets.
0000 116 $DCDEF ; Define adapter types
0000 117 $DDBDEF ; Define DDB offsets
0000 118 $DYNDEF ; Define data structure type codes.
0000 119 $I0BDEF ; Define interrupt dispatcher offsets.
0000 125 $I0780DEF ; Define 11/780 I/O space.
0000 137 $MCHKDEF ; Define machine check masks.
0000 138 $NDTDEF ; Define nexus device types.
0000 139 $PRDEF ; Define IPR numbers.
0000 140
0000 142 $PR780DEF ; Define 11/780 specific IPR numbers.
0000 144
0000 148
0000 152
0000 156
0000 160
;RLRSCORPIO 0000 .4
0000 161 $PTEDEF ; Define Page Table Entry bits.
0000 162 $RPBDEF ; Define Restart Parameter Block fields.
0000 163 $UBADEF ; Define UBA register offsets.
0000 164 $UCBDEF ; Define UCB offsets.
0000 165 $VADEF ; Define virtual address fields.
0000 166 $VECDEF ; Define vec offsets.
```

```

0000 168      .SBTTL  Macros to describe nexus configurations
0000 169 ;
0000 170 ;   The macros FLOAT_NEXUS and FIXED_NEXUS add one or more entries to a
0000 171 ;   nexus descriptor table.  Each entry is of the form:
0000 172 ;           +-----+
0000 173 ;           | PFN of nexus I/O space |
0000 174 ;           +-----+
;KPL0100 0000 .1 ;           | bus | 0 | type |
;KPL0100 0000 .2 ;           +-----+
;KPL0100 0000 .3 ;
;KPL0100 0000 .4 ;   type = 0 -> floating nexus
;KPL0100 0000 .5 ;   type = non-zero -> fixed nexus; type = fixed adapter type
;KPL0100 0000 .6 ;   bus = 0, if SBI; %x80 if BI (this is a VMS-only designation)
;KPL0100 0000 .7 ;
;KPL0100 0000 .8 ;
;KPL0100 0000 .9 ;   device_type:  SBI adapters have 8-bit device type codes.  These
;KPL0100 0000 .10 ;   device types are simple integers.
;KPL0100 0000 .11 ;
;KPL0100 0000 .12 ;   BI adapters have 16-bit device type codes, that are
;KPL0100 0000 .13 ;   subject to the following interpretation:
;KPL0100 0000 .14 ;
;KPL0100 0000 .15 ;   - the MSB of the device-type field will be 0 for DEC
;KPL0100 0000 .16 ;   devices and 1 for non-DEC devices,
;KPL0100 0000 .17 ;
;KPL0100 0000 .18 ;   - DEC memory devices will have 0s in the high-order
;KPL0100 0000 .19 ;   byte of the device type,
;KPL0100 0000 .20 ;
;KPL0100 0000 .21 ;   - non-DEC supplied memory devices will have a 1 in the
;KPL0100 0000 .22 ;   MSB of the high-order byte, and the rest of the high
;KPL0100 0000 .23 ;   order byte will contain 0s.
;KPL0100 0000 .24 ;
;KPL0100 0000 .25 ;   - The "all 0s" and "all 1s" device-type codes are
;KPL0100 0000 .26 ;   reserved for DEC.
;KPL0100 0000 .27 ;
;KPL0100 0000 .28 ;   If SBI type codes were simply expanded to a word for purposes of the routines
;KPL0100 0000 .29 ;   in this module, there would be possible conflicts between SBI devices and
;KPL0100 0000 .30 ;   BI memory adapters supplied by DEC.  Voila:  the bus type.
-5      0000 179 ;
0000 180 ; Macro FLOAT_NEXUS.
0000 181 ; INPUTS:
0000 182 ;   PHYSADR -- physical address of 1 or more contiguous floating nexus
0000 183 ;   slots
0000 184 ;   NUMNEX -- number of contiguous floating nexuses, default = 1
0000 185 ;   PERNEX -- amount of address space per nexus (does not have to be
0000 186 ;   specified if NUMNEX = 1)
0000 187 ;
0000 188 ; .MACRO  FLOAT_NEXUS      PHYSADR,NUMNEX=1,PERNEX=0
0000 189 ; PA = PHYSADR
0000 190 ; .REPEAT NUMNEX          ; For each nexus...
0000 191 ; .LONG  <PA/^X200>      ; Store PFN.
0000 192 ; .LONG  0                ; Store floating nexus type.
0000 193 ; PA = PA + PERNEX      ; Increment to physical address of next nexus.
0000 194 ; .ENDR
0000 195 ; .ENDM  FLOAT_NEXUS
0000 196 ;
0000 197 ;
0000 198 ; Macro FIXED_NEXUS.
0000 199 ;

```

```

0000 200 ; INPUTS:
0000 201 ; PHYSADR - physical address of 1 or more contiguous fixed nexus slots
0000 202 ; PERNEX - amount of address space per nexus
0000 203 ; NEXUSTYPES - a list of fixed nexus types, enclosed in <>
0000 204 ;
0000 205 .MACRO FIXED_NEXUS PHYSADR,PERNEX=0,NEXUSTYPES
0000 206 PA = PHYSADR
0000 207 .IRP TYPECODE,NEXUSTYPES ; For each fixed nexus type...
0000 208 .LONG <PA/^X200> ; Store PFN.
;RLRSCORPIO 0000 .1 .LONG TYPECODE ; Store fixed nexus type.
;RLRSCORPIO 0000 .2 PA = PA + PERNEX ; Increment to address of next nexus.
;RLRSCORPIO 0000 .3 .ENDR
;RLRSCORPIO 0000 .4 .ENDM FIXED_NEXUS
;RLRSCORPIO 0000 .5
;RLRSCORPIO 0000 .6 ;
;RLRSCORPIO 0000 .7 ; Macro NEXUSDESC_TABLE - declare the beginning of a NEXUS descriptor table
;RLRSCORPIO 0000 .8 ;
;RLRSCORPIO 0000 .9 ; 1st byte in table (at offset -5 from label) contains length of
;RLRSCORPIO 0000 .10 ; adapter type code field in CSR's on this bus. [Note for SBI like
;RLRSCORPIO 0000 .11 ; busses, this is 1.] The next longword (at offset -4) in the
;RLRSCORPIO 0000 .12 ; table contains the Software defined bus type byte defined in the
;RLRSCORPIO 0000 .13 ; high order byte of the longword. [Note for SBI like busses, this
;RLRSCORPIO 0000 .14 ; value is 0, for the BI it is ^x80.]
;RLRSCORPIO 0000 .15 ;
;RLRSCORPIO 0000 .16
;RLRSCORPIO 0000 .17 ; Define parameters that may be specified or used in macro invocation.
;RLRSCORPIO 0000 .18
;RLRSCORPIO 00000000 0000 .19 BI_LIKE = 0 ; BI like bus.
;RLRSCORPIO 00000001 0000 .20 SBI_LIKE = 1 ; SBI like bus.
;RLRSCORPIO 0000 .21
;RLRSCORPIO 00000001 0000 .22 SBI_CSR_LEN = 1 ; Length of type code field in adapter CSR's
;RLRSCORPIO 0000 .23 ; on SBI, CMI, etc.
;RLRSCORPIO 00000002 0000 .24 BI_CSR_LEN = 2 ; Length of type code field in adapter CSR's
;RLRSCORPIO 0000 .25 ; on BI.
;RLRSCORPIO 0000 .26
;RLRSCORPIO 00000000 0000 .27 SBI_BUS_CODE = 0 ; Software defined bus code for SBI like busses.
;RLRSCORPIO 80000000 0000 .28 BI_BUS_CODE = ^x80000000 ; Software defined bus code for the BI.
;RLRSCORPIO 0000 .29
;RLRSCORPIO 0000 .30 .MACRO NEXUSDESC_TABLE LABEL,BUS_TYPE=SBI_LIKE
;RLRSCORPIO 0000 .31 .IF EQ,BUS_TYPE-SBI_LIKE
;RLRSCORPIO 0000 .32 .BYTE SBI_CSR_LEN
;RLRSCORPIO 0000 .33 .LONG SBI_BUS_CODE
;RLRSCORPIO 0000 .34 .IFF
;RLRSCORPIO 0000 .35 .IF EQ,BUS_TYPE-BI_LIKE
;RLRSCORPIO 0000 .36 .BYTE BI_CSR_LEN
;RLRSCORPIO 0000 .37 .LONG BI_BUS_CODE
;RLRSCORPIO 0000 .38 .IFF
;RLRSCORPIO 0000 .39 .ERROR ; UNRECOGNIZED BUS TYPE, NEXUSDESC_TABLE;
;RLRSCORPIO 0000 .40 .ENDC
;RLRSCORPIO 0000 .41 .ENDC
;RLRSCORPIO 0000 .42
;RLRSCORPIO 0000 .43 LABEL:
;RLRSCORPIO 0000 .44 .ENDM NEXUSDESC_TABLE
;RLRSCORPIO 0000 .45
;RLRSCORPIO FFFFFFFB 0000 .46 CSR_LEN_OFFSET = -5 ; Offset before nexus descriptor of
;RLRSCORPIO 0000 .47 ; byte containing length of adapter
;RLRSCORPIO 0000 .48 ; type field in adapter CSR.

```

INIADP780  
V03-021

- ADAPTER INITIALIZATION FOR VAX 11/780 3-JUN-1984 14:53:36 VAX-11 Macro V03-01 Page 7  
Macros to describe nexus configurations 19-OCT-1983 11:50:19 DISK\$VMSMASTER:[SYSLOA.SRC]INIADP.(3)

```
;RLRSCORPIO          FFFFFFFC 0000 .49 BUS_CODE_OFFSET = -4          ; Offset before nexus descriptor table
;RLRSCORPIO          0000 .50          ; of longword containing software
;RLRSCORPIO          0000 .51          ; defined bus type to be or'ed with
;RLRSCORPIO          0000 .52          ; adapter type to produce NDT$_ value.
-5                   0000 214 ;
                   0000 215 ; Macro END_NEXUSDESC.
                   0000 216 ;
                   0000 217 .MACRO END_NEXUSDESC
                   0000 218 .LONG 0          ; PFN=0 -> end of nexus descriptors.
                   0000 219 .ENDM END_NEXUSDESC
```

```

;RLRSCORPIO      0000      .1      .SBTTL  Adapter-specific data structures
;RLRSCORPIO      0000      .2      ;
;RLRSCORPIO      0000      .3      ; Put a symbol for arrays built by macros in the correct psects.
;RLRSCORPIO      0000      .4      ;
;RLRSCORPIO      0000      .5      ;***** ADAPTERS array *****
;RLRSCORPIO      00000000 .6      .PSECT $$$INIT$DATA0
;RLRSCORPIO      0000      .7      ADAPTERS: ; Build adapter type code arrays here.
;RLRSCORPIO      0000      .8      ;
;RLRSCORPIO      00000000 .9      .PSECT $$$INIT$DATA1 ; User contributions in this .PSECT.
;RLRSCORPIO      0000      .10     ; End of ADAPTERS array.
;RLRSCORPIO      0000      .11     ;***** End of ADAPTERS array *****
;RLRSCORPIO      0000      .12     ;
;RLRSCORPIO      0000      .13     ;***** NUM_PAGES array *****
;RLRSCORPIO      00000000 .14     .PSECT $$$INIT$DATA2
;RLRSCORPIO      0000      .15     NUM_PAGES: ; Build "number of pages to map" array.
;RLRSCORPIO      00000000 .16     .PSECT $$$INIT$DATA3 ; User contributions in this .PSECT.
;RLRSCORPIO      0000      .17     ;***** End of NUM_PAGES array *****
;RLRSCORPIO      0000      .18     ;
;RLRSCORPIO      0000      .19     ;***** INIT_ROUTINES array *****
;RLRSCORPIO      00000000 .20     .PSECT $$$INIT$DATA4
;RLRSCORPIO      0000      .21     INIT_ROUTINES: ; Build "address of init routine" array.
;RLRSCORPIO      00000000 .22     .PSECT $$$INIT$DATA5 ; User contributions in this .PSECT.
;RLRSCORPIO      0000      .23     ;***** End of INIT_ROUTINES array *****
-65              0000      286
              0000      287 ;
              0000      288 ; To add a new adapter type:
              0000      289 ; 1) Add a new ADAPDESC macro invocation to the end of this list.
              0000      290 ;
00000000        291     .PSECT $$$INIT$DATA, LONG
              0000      292
              0000      293 ;
              0000      294 ; Default interrupt vectors for UNIBUS system devices
              0000      295 ; (This array is indexed by the RPB field RPB$B_DEVTYPE, if the RPB field
              0000      296 ; RPB$W_ROUBVEC is zero. If RPB$W_ROUBVEC is not zero, then RPB$W_ROUBVEC
              0000      297 ; is used and this array is not referenced at all. RPB$W_ROUBVEC is set up
              0000      298 ; by PQDRIVER. RPB$L_BOOTRO is set by VMB to contain the device name in
              0000      299 ; ASCII, not the vector number and device type, as it does on full
              0000      300 ; architecture VAX machines.
              0000      301 ;
              0000      302 BOOTVECTOR:
0088 0000      303     .WORD    ^X88 ; RK06/7 Interrupt vector
0070 0002      304     .WORD    ^X70 ; RL01/2 Interrupt vector
              0004      305
;RLRSCORPIO      0004      .1      BUS_CSR_LEN: ; Static byte containing the length (in bytes)
;RLRSCORPIO      00 0004      .2      BYTE    0 ; of the adapter type field in the CSR's of
;RLRSCORPIO      0005      .3      ; the bus currently being configured. The
;RLRSCORPIO      0005      .4      ; proper value for the bus of interest is
;RLRSCORPIO      0005      .5      ; copied here, from the current nexus
;RLRSCORPIO      0005      .6      ; descriptor table, when we enter subroutine
;RLRSCORPIO      0005      .7      ; CONFIG_IOSPACE.
;RLRSCORPIO      0005      .8      ;
;RLRSCORPIO      0005      .9      SW_BUS_CODE: ; Static longword containing the software
;RLRSCORPIO      00000000 0005      .10     LONG    0 ; defined bus type, of the bus currently being
;RLRSCORPIO      0009      .11     ; configured, in the high order byte. The
;RLRSCORPIO      0009      .12     ; proper value for the bus of current interest
;RLRSCORPIO      0009      .13     ; is copied here, from the nexus descriptor
;RLRSCORPIO      0009      .14     ; table, when we enter subroutine

```



```

;RLRSCORPIO          0009 .15          ;. CONFIG_IOSPACE.
;RLRSCORPIO          0009 .16
;RLRSCORPIO          0009 .17 DIRECT_VEC_NODE_CNT:          ; Static longword that counts the number of
;RLRSCORPIO          0009 .18          ; direct vectoring adapter nodes that we have
;RLRSCORPIO          00000000 0009 .19          .LONG 0          ; run across so far.
;RLRSCORPIO          0000 000D .20
;RLRSCORPIO          00000001 000D .21 $$$VMSDEFINED = 1          ; Define symbol that means VMS system software.
;RLRSCORPIO          000000080 000D .22 NUMUBAVEC = 128          ; ALLOW FOR 128 UNIBUS VECTORS
;RLRSCORPIO          000D .23
;RLRSCORPIO          000D .24 ADAPDESC -          ; Memory. ** MUST BE 1ST IN DESCRIPTOR LIST **
;RLRSCORPIO          000D .25 ADPTYPES=<NDT$ MEM1664NI,NDT$ MEM4NI,NDT$ MEM4I,NDT$ MEM16NI, -
;RLRSCORPIO          000D .26 NDT$ MEM16I,NDT$ MEM64NIL,NDT$ MEM64EIL,NDT$ MEM64NIU, -
;RLRSCORPIO          000D .27 NDT$ MEM64EIU, NDT$ MEM64I, NDT$ _SCORMEM> -
-8 000D 314 NUMPAGES=1
000D 315
000D 316 ADAPDESC -          ; MASSbus.
000D 317 ADPTYPES=NDT$ _MB, -
000D 318 NUMPAGES=8, -
000D 319 INITRTN=INI$MBADP
000D 320
ADAPDESC -          ; UNibus.
000D 321 ADPTYPES=<NDT$ _UB0,NDT$ _UB1,NDT$ _UB2,NDT$ _UB3,NDT$ _BUA>, -
;RLRSCORPIO          000D .1 NUMPAGES=8, -
-1 000D 323 INITRTN=INI$UBSPACE
000D 324
000D 325
ADAPDESC -          ; Multi-port memory.
000D 326 ADPTYPES=<NDT$ _MPM0,NDT$ _MPM1,NDT$ _MPM2,NDT$ _MPM3>, -
000D 327 NUMPAGES=1, -
000D 328 INITRTN=INI$MPMADP
000D 329
000D 330
ADAPDESC -          ; DR32.
000D 331 ADPTYPES=NDT$ _DR32, -
000D 332 NUMPAGES=4, -
000D 333 INITRTN=INI$DRADP
000D 334
000D 335
ADAPDESC -          ; CI780
000D 336 ADPTYPES=NDT$ _CI, -
;KPL0100 000D .1 NUMPAGES=9, -
;RLRSCORPIO 000D .2 INITRTN=INI$CIADP
;RLRSCORPIO 000D .3
;RLRSCORPIO 000D .4
;RLRSCORPIO 000D .5 ADAPDESC -          ; KDZ11 Processor
;RLRSCORPIO 000D .6 ADPTYPES=NDT$ _KDZ11, -
;RLRSCORPIO 000D .7 NUMPAGES=1, -
;RLRSCORPIO 000D .8 INITRTN=INI$KDZ11

```

```
000D 343 ;
000D 344 ; TABLES OF ADAPTER-DEPENDENT INFORMATION
000D 345 ;
000D 346 ; THE TABLE OFFSETS ARE:
000D 347 ;
000D 348 ;           $DEFINI ADPTAB
0000 349
00000001 0000 350 ADPTAB_IDBUNITS: .BLKB 1 ; # UNITS TO SET IN IDB
00000003 0001 351 ADPTAB_ADPLEN: .BLKW 1 ; # LENGTH OF ADP
00000004 0003 352 ADPTAB_ATYPE: .BLKB 1 ; ADP TYPE
0004 353
0004 354 ;           $DEFEND ADPTAB
000D 355
000D 356 ;
000D 357 ; TABLES THEMSELVES:
000D 358 ;
000D 359
000D 360 MBATAB: ; TABLE OF MBA CONSTANTS
08 000D 361 .BYTE 8 ; # UNITS IN MBA IDB
0030 000E 362 .WORD ADP$C_MBAADPLEN ; # BYTES IN MBA ADP
00 0010 363 .BYTE AT$_MBA ; MBA ADAPTER TYPE
0011 364
0011 365 DRTAB: ; TABLE OF DR32 CONSTANTS
01 0011 366 .BYTE 1 ; # UNITS IN DR IDB
0030 0012 367 .WORD ADP$C_DRADPLEN ; # BYTES IN DR ADP
02 0014 368 .BYTE AT$_DR ; DR ADAPTER TYPE
0015 369
0015 370 CITAB: ; TABLE OF CI CONSTANTS
01 0015 371 .BYTE 1 ; # UNITS IN CI IDB
0030 0016 372 .WORD ADP$C_CIADPLEN ; # BYTES IN CI ADP
04 0018 373 .BYTE AT$_CI ; CI ADAPTER TYPE
0019 374
```

```

0019 378 .SBTTL CPU-specific data structures
0019 379 ;
0019 380 ; To add a new CPU type:
0019 381 ; 1) Create a new nexus descriptor table, using FLOAT_NEXUS and
0019 382 ; FIXED_NEXUS macros. Put an END_NEXUSDESC macro at the end.
0019 383 ;
0019 384 ;
0019 386 ;
0019 387 CPU_ADPSIZE:
04EC' 0019 388 .WORD ADP$C_UBAADPLEN+UBINTSZ+<NUMUBAVEC*4>
001B 389
;KPL0100 001B .1
;KPL0100 001B .2 ;
;KPL0100 001B .3 ; Declare the beginning of a nexus-descriptor table.
;KPL0100 001B .4 ;
;KPL0100 001B .5 NEXUSDESC_TABLE LABEL=NEXUSDESC
;RLRSCORPIO 0020 .6
;RLRSCORPIO 0020 .7 ;
;RLRSCORPIO 0020 .8 ; Describe all possible nexuses on an 11/780.
;RLRSCORPIO 0020 .9 ;
;RLRSCORPIO 00000001 0020 .10 SBI_CPU = 1
;RLRSCORPIO 00000000 0020 .11 BI_CPU = 0
-7 0020 397 FLOAT_NEXUS -
0020 398 PHYSADR=I0780$AL_IOBASE, -
0020 399 NUMNEX=I0780$AL_NNEX, -
0020 400 PERNEX=I0780$AL_PERNEX
00A0 401 END_NEXUSDESC
00A4 403
00A4 436
00A4 458
00A4 495
00A4 496
00A4 513
;RLRSCORPIO 00A4 .24
;RLRSCORPIO 00A4 .25 ;
;RLRSCORPIO 00A4 .26 ; Nexus "descriptor" arrays -- these arrays hold the nexus-device type and
;RLRSCORPIO 00A4 .27 ; virtual address of every adapter on the system. The arrays, CONFREG and
;RLRSCORPIO 00A4 .28 ; SBICONF, are allocated enough space to hold the maximum number of adapters
;RLRSCORPIO 00A4 .29 ; that can be attached to any CPU. When the code discovers how many adapters
;RLRSCORPIO 00A4 .30 ; actually exist on the system, it will allocate space from non-paged pool
;RLRSCORPIO 00A4 .31 ; and move a permanent copy of these arrays into that space.
;RLRSCORPIO 00A4 .32 ;
;KPL0100 00000040 00A4 .33 MAXNEXUS = 64
;KPL0100 00A4 .34 CONFREG: ; Byte array of nexus-device type codes..
;KPL0100 000000E4 00A4 .35 .BLKB MAXNEXUS
;KPL0100 00E4 .36 SBICONF:
;KPL0100 000001E4 00E4 .37 .BLKL MAXNEXUS ; Longword array of VAs of adapter space.
;KPL0100 01E4 .38 CONFREG:
;KPL0100 000002E4 01E4 .39 .BLKL MAXNEXUS ; Longword array of nexus-device type codes

```

```
02E4 528 .SBTTL Message strings
02E4 529
0000000D 02E4 530 CR = 13
0000000A 02E4 531 LF = 10
02E4 532 NOSPT:
45 58 45 25 0A 0D 02E4 533 .ASCIZ <CR><LF>/%EXECINIT-F-Insufficient SPT entries/<CR><LF>
2D 54 49 4E 49 43 02EA
75 73 6E 49 2D 46 02F0
65 69 63 69 66 66 02F6
54 50 53 20 74 6E 02F6
69 72 74 6E 65 20 0302
00 0A 0D 73 65 0308
030D 535 BADUMR:
45 58 45 25 0A 0D 030D 536 .ASCIZ <CR><LF>/%EXECINIT-F-UNIBUS memory does not start at 0/<CR><LF>
2D 54 49 4E 49 43 0313
42 49 4E 55 2D 46 0319
6D 65 6D 20 53 55 031F
6F 64 20 79 72 6F 0325
74 6F 6E 20 73 65 032B
74 72 61 74 73 20 0331
0D 30 20 74 61 20 0337
00 0A 033D
```



INIADP780  
V03-021

```
          5A          002F
    00E4'CF DE 0030 592      MOVAL  W^SBICONF,G^MMG$GL_SBICONF ; Set pointers to local copies
00000000'GF          0034
    00A4'CF DE 0039 593      MOVAL  W^CONFREG,G^EXE$GL_CONFREG ; of these arrays for init routines.
00000000'GF          003D
;KPL0100 01E4'CF DE 0042 .1    MOVAL  W^CONFREGL,G^EXE$GL_CONFREGL ; ...
00000000'GF          0046
```

```

004B 691          .SBTTL  INITADP_780, _750, _730, and _UV1
004B 692 ;
004B 693 ; I/O address space for the 11/780, 11/750, 11/730, and Micro-VAX I cpus
004B 694 ; is statically defined in their respective nexus descriptor tables.
004B 695 ;
56  0020'CF  DE 004B 696          MOVAL  W^NEXUSDESC,R6          ; Get address of nexus table.
      5B  D4  0050 697          CLRL   R11                   ; Signal use 1st page of SCB.
      0B  10  0052 698          BSBB   CONFIG_IOSPACE       ; Configure processor I/O space.
      0054 699
      0054 701
      00C3 30  0054 702          BSBW   CREATE_ARRAYS          ; Create CONFREG and SBICONF arrays.
OFFF 8F  BA  0057 703          POPR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
50  01  DO  005B 704          MOVL   #1,R0                   ; Set success status
      05  005E 705          RSB    ; Return.

```

```

005F 708 .SBTTL CONFIG_IOSPACE
005F 709 ;
005F 710 ; CONFIG_IOSPACE
005F 711 ; Given a nexus descriptor table, which describes what "nexuses" or
005F 712 ; "slots" are available on a system to hold I/O adapters, find and
005F 713 ; initialize all adapters on the system.
005F 714 ;
005F 715 ; Inputs:
005F 716 ; R2 - next available virtual address, to be used for mapping I/O space
005F 717 ; R3 - address of PTE associated with VA in R2
005F 718 ; R4 - Current index into CONFREG and SBICONF arrays (should be 0 the
005F 719 ; first time CONFIG_IOSPACE is called)
005F 720 ; R6 - address of nexus descriptor table
005F 721 ; R9 - address of Restart Parameter Block (RPB)
005F 722 ; R10 - PFN of boot adapter space
005F 723 ; R11- page offset from beginning of SCB; tells which page of the SCB
005F 724 ; to use for this set of nexuses (passed to routines that init ADP)
005F 725 ;
005F 726 ; Outputs:
005F 727 ; R2,R3,R4 - updated
005F 728 ; R9,R10,R11 - preserved; all other registers potentially modified
005F 729 ; CONFREG - initialized with adapter NDT$ code for each nexus
005F 730 ; SBICONF - initialized with adapter space VA for each nexus
005F 731 ;
005F 732 CONFIG_IOSPACE:
005F 734 ;
005F 735 ; Main loop. Map and initialize all adapters on system.
005F 736 ;
;RLRSCORPIO 005F .1
;RLRSCORPIO FB A6 90 005F .2 MOVB CSR_LEN_OFFSET(R6),- ; Move length of adapter type field
;RLRSCORPIO 0004'CF 0062 .3 ; in CSR's to static location.
;RLRSCORPIO FC A6 D0 0065 .4 MOVL BUS_CODE_OFFSET(R6),- ; Move software defined bus type code
;RLRSCORPIO 0005'CF 0068 .5 W^SW_BUS_CODE ; to static longword.
;RLRSCORPIO 006B .6
006B 742 NXT_NEXUS: ; For each nexus...
58 86 D0 006B 743 MOVL (R6)+,R8 ; Get PFN of nexus.
01 12 006E 745 BNEQ TEST_NEXUS ; If PFN non-zero, go test the slot.
05 0070 746 RSB ; If 0, we've found all nexuses.
0071 747 ;
0071 748 ; Read configuration register to determine if anything is present at this
0071 749 ; nexus.
0071 750 ;
0071 751 TEST_NEXUS:
90000000 8F C9 0071 752 BISL3 #PTESM_VALID!PTESC_KW,- ; Temporarily associate VA in R2 with
63 58 0077 753 R8,(R3) ; PFN in R8 via SPTE in R3.
0079 754 $PRTCTINI B^10$, - ; Protect following code from non-
0079 755 #<MCHK$M_NEXM!MCHK$M_LOG>; existent memory machine checks.
51 62 D0 0085 756 MOVL (R2),R1 ; Read adapter configuration register.
0088 757 $PRTCTEND 10$ ; End of protected code.
0089 758 INVALID R2 ; Clear TB of temporary mapping.
11 50 E8 008C 759 BLBS R0,GET_TYPE ; Branch if no machine check occurred.
008F 760 ;
008F 761 ; No adapter present at this nexus.
008F 762 ;
;KPL0100 00A4'CF44 94 008F .1 CLRB W^CONFREG[R4] ; Store "unknown" type in CONFREG
;KPL0100 01E4'CF44 D4 0094 .2 CLRL W^CONFREGL[R4] ; and in CONFREGL also.
;KPL0100 55 D4 0099 .3 CLRL R5 ; Use general memory type to map

```



```

;KPL0100          009B .4 ; one page of I/O space.
;KPL0100          56 04 C0 009B .5 ADDL2 #4,R6 ; Step past type code in nexus table.
-4              59 11 009E 767 BRB MAP_NEXUS ; Go map I/O space for this nexus.
                . 00A0 769 ;
                00A0 770 ; Execution continues here if adapter was present.
                00A0 771 ;
                00A0 772 GET_TYPE:
;RLRSCORPIO      57 86 D0 00A0 .1 MOVL (R6)+,R7 ; Get nexus-device type from nexus table.
;RLRSCORPIO      14 12 00A3 .3 BNEQ GET_GEN_TYPE ; Branch if fixed slot.
;RLRSCORPIO      00A5 .4 ;
;RLRSCORPIO      00A5 .5 ; Floating-type slot. Use type from configuration register.
;RLRSCORPIO      00A5 .6 ; Determine if type in configuration register is 8-bits or 16-bits.
;RLRSCORPIO      00A5 .7 ;
;RLRSCORPIO      00A5 .8 ;
;RLRSCORPIO      0004'CF 01 91 00A5 .9 CMPB #1,W^BUS_CSR_LEN ; Determine length of adapter type
;RLRSCORPIO      00AA .10 ; field in CSR contained in R7.
;RLRSCORPIO      05 13 00AA .11 BEQL 10$ ; EQL implies 1 byte (8-bit) field.
;RLRSCORPIO      57 51 3C 00AC .12 MOVZWL R1,R7 ; BI_LIKE, so use word instruction.
;RLRSCORPIO      03 11 00AF .13 BRB 20$ ; Skip byte instruction.
;RLRSCORPIO      57 51 9A 00B1 .14 10$: MOVZBL R1,R7 ; Use byte instruction to get type.
;RLRSCORPIO      00B4 .15 20$: ;
;RLRSCORPIO      57 0005'CF C8 00B4 .16 BISL W^SW_BUS_CODE,R7 ; Or in software bus code.
;RLRSCORPIO      00B9 .18 ;
;RLRSCORPIO      00B9 .19 ; Here R7 has hardware adapter code or'ed with software bus code.
;RLRSCORPIO      00B9 .20 ; Translate specific nexus device type code into general adapter type code.
;RLRSCORPIO      00B9 .21 ;
;RLRSCORPIO      00B9 .22 GET_GEN_TYPE:
;RLRSCORPIO      00A4'CF44 57 90 00B9 .23 MOVB R7,W^CONFREG[R4] ; Save nexus-device type in CONFREG.
;RLRSCORPIO      01E4'CF44 57 D0 00BF .24 MOVL R7,W^CONFREGL[R4] ; CONFREGL also filled in.
;RLRSCORPIO      55 D4 00C5 .25 CLRL R5 ; Clear loop index.
;RLRSCORPIO      00C7 .26 30$: ;
;RLRSCORPIO      50 0000'CF45 DE 00C7 .27 MOVAL W^ADAPTERS[R5],R0 ; Get address of adapter type code.
;RLRSCORPIO      0000'CF 9F 00CD .28 PUSHAB W^NUM_PAGES ; Push addr of end of ADAPTERS array.
;RLRSCORPIO      8E 50 D1 00D1 .29 CMPL R0,(SP)+ ; See if we went beyond array.
;RLRSCORPIO      3F 1E 00D4 .30 BGEQU END_NEXUS ; unrecognized adapter, do not map.
;RLRSCORPIO      60 57 D1 00D6 .31 CMPL R7,(R0) ; Adapter type match?
;RLRSCORPIO      04 13 00D9 .32 BEQL 40$ ; If EQL yes, adapter type match.
;RLRSCORPIO      55 D6 00DB .33 INCL R5 ; Increment loop index.
;RLRSCORPIO      E8 11 00DD .34 BRB 30$ ; Look at next adapter.
;RLRSCORPIO      00DF .35 40$:
-16            00DF 789 ;
                00DF 790 ;
                00DF 791 ; Store boot parameters.
                00DF 792 ;
                5A 58 D1 00DF 794 CMPL R8,R10 ; Does PFN match boot adapter's PFN?
                15 12 00E2 795 BNEQ MAP_NEXUS ; No; continue.
                60 A9 52 D0 00E4 797 MOVL R2,RPB$L_ADPVIR(R9) ; Store VA of boot adapter space.
                20 A9 54 D0 00E8 798 MOVL R4,RPB$L_BOOTR1(R9) ; Store boot adapter nexus number.
                0D 00 EF 00EC 799 EXTZV #0,#13, - ; Get offset into UNIBUS/QBUS I/O page.
                51 54 A9 00EF
                00F2 800 ;
                1000 C241 9E 00F2 801 MOVAB RPB$L_CSRPHY(R9),R1 ; Set VA of UNIBUS/QBUS registers.
                58 A9 00F7
                00F9 802 RPB$L_CSRVIR(R9) ;
                00F9 803 ;
                00F9 804 ;
                00F9 805 ;

```

```

00F9 806 ; R5/ general adapter type; index into "general" adapter arrays.
00F9 807 ; For each adapter -
00F9 808 ; Map the # of pages specified in ADAPDESC macro
00F9 809 ; JSB to initialization routine specified in ADAPDESC macro
00F9 810 ;
00F9 811 MAP_NEXUS:
00E4'CF44 52 D0 00F9 816 MOVL R2,W^SBICONF[R4] ; Save VA of adapter space in SBICONF.
51 0000'CF45 3C 00FF 817 MOVZWL W^NUM_PAGES[R5],R1 ; Get number of pages to map.
6C 10 0105 818 BSBB MAP_PAGES ; Map the I/O pages.
51 0000'CF45 DE 0107 819 MOVAL W^INIT_ROUTINES[R5],R1 ; Get address of initialization routine.
61 D5 010D 820 TSTL (R1) ; Initialization routine specified?
04 13 010F 821 BEQL END_NEXUS ; Branch if none.
00 B141 16 0111 822 JSB @(R1)[R1] ; Call initialization routine.
0115 823 END_NEXUS:
54 D6 0115 824 INCL R4 ; Increment CONFREG and SBICONF index.
FF51 31 0117 826 BRW NXT_NEXUS ; Go do next nexus.
011A 830

```



```
0173 867 .SBTTL MAP_PAGES
0173 868 ;++
0173 869 ; INPUTS:
0173 870 ; R1/ Number of pages to map.
0173 871 ; R2/ VA of page to map.
0173 872 ; R3/ VA of system page table entry to be used.
0173 873 ; R8/ PFN of page(s) to map.
0173 874 ;
0173 875 ; OUTPUTS:
0173 876 ; R2,R3 updated; R1,R8 destroyed; all other registers preserved
0173 877 ;
0173 878 ;--
0173 879
0173 880 MAP_PAGES:
0173 881
90000000 8F C9 0173 882 BISL3 #<PTE$M_VALID!PTE$C_KW>,R8,(R3)+
83 58 0179
58 D6 017B 883 ; Map a page.
52 0200 C2 9E 017D 884 INCL R8 ; Next PFN.
00000000'GF D6 0182 885 MOVAB 512(R2),R2 ; Next VA.
00000000'GF D1 0188 886 INCL G^B00$GL_SPTFREL ; Next free entry.
00000000'GF 018E 887 CMPL G^B00$GL_SPTFREL, - ; Check for no more system page
0193 888 G^B00$GL_SPTFREL ; table entries.
04 15 0193 889 BLEQ ERROR HALT ; Branch if out of SPTEs.
DB 51 F5 0195 890 SOBGTR R1,MAP_PAGES ; Map another page.
05 0198 891 RSB ; All done.
0199 892
0199 893 ERROR_HALT:
51 02E4'CF 9E 0199 894 MOVAB W^NOSPT,R1 ; Set error message.
019E 895 ERROR_HALT_1:
5B D4 019E 896 CLRL R11 ; Indicate console terminal.
00000000'GF 16 01A0 897 JSB G^EXE$OUTZSTRING ; Output error message.
00 01A6 898 HALT ; ***** FATAL ERROR *****
```

```

01A7 1016 .SBTTL INI$SUBSPACE
01A7 1017 ;++
01A7 1018 ; Map UNIBUS space; initialize UNIBUS ADP.
01A7 1019 ;
01A7 1020 ; INPUTS:
01A7 1021 ; R2 - VA of next free system page
01A7 1022 ; R3 - VA of system page table entry to be used to map VA in R2
01A7 1023 ; R4 - nexus identification number of this adapter
01A7 .1 ; -8(R6) - PFN of this UNIBUS adapter's register space
01A7 1025 ;
01A7 1026 ; OUTPUTS:
01A7 1027 ; UNIBUS space is mapped.
01A7 1028 ; INI$UBADP is called to build an ADP block and initialize UNIBUS
01A7 1029 ; adapter hardware.
01A7 1030 ;
01A7 1031 ;--
01A7 1032
01A7 1033 INI$SUBSPACE:
01A7 1034
;RLRSCORPIO 58 01E4'CF44 DE 01A7 .3 MOVAL WACONFREGL[R4],R8 ; R8 => CONFREGL slot.
;RLRSCORPIO 68 02 00 EF 01AD .4 EXTZV #0,#2,(R8),R8 ; Get UBA number.
;RLRSCORPIO 58 58 09 78 01B1 .5 ASHL #9,R8,R8 ; Position UB number.
;RLRSCORPIO -2 001009F0 8F 58 C0 01B6 1037 ADDL #<I0780$AL_UBOSP+^0760000/AX200>,R8
01BC 01BD 1040 ; Get PFN of Ub I/O page.
01BD 1042
01BD 1047
01BD 1052
01BD 1058
01BD 1063
51 10 D0 01BD 1064 MOVL #16,R1 ; Number of pages to map (UB/Qbus space).
FFB0 30 01C0 1065 BSBW MAP_PAGES ; Map I/O pages.
01C3 1066 ;
01C3 1067 ; Call adapter initialization routine.
01C3 1068 ;
01C3 1069 ; BSBW INI$UBADP ; Init ADP block.
01C3 1070 ; RSB

```

```

01C3 1072          .SBTTL  INI$UBADP - BUILD ADP AND INITIALIZE UBA
01C3 1073 ;+
01C3 1074 ; INI$UBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
01C3 1075 ; DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS.  A CALL IS
01C3 1076 ; THEN MADE TO UBA$INITIAL TO INITIALIZE THE ADAPTER HARDWARE.
01C3 1077 ;
01C3 1078 ; INPUT:
01C3 1079 ;           R4 - nexus identification number of this adapter
01C3 1080 ;           R11- offset from beginning of SCB to correct SCB page for this adapter
01C3 1081 ; -
01C3 1082
01C3 1083 INI$UBADP:
01C3 1084
01FF 8F  BB 01C3 1085          PUSHR  #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; SAVE R0-R8
01C7 1086 ;
01C7 1087 ; Allocate and initialize Adapter Control Block (ADP).
01C7 1088 ;
51 0019'CF 3C 01C7 1089          MOVZWL  W^CPU_ADPSIZE,R1          ; PICK UP LENGTH OF ADP
      0205 30 01CC 1090          BSBW    ALONPAGD              ; ALLOCATE SPACE FOR ADP
08 A2 51  B0 01CF 1091          MOVW    R1,ADP$W_SIZE(R2)      ; SET SIZE INTO ADP BLOCK
0A A2 01  90 01D3 1092          MOVB    #DYN$C_ADP, -        ; AND SET TYPE OF BLOCK
      01D7 1093          ADP$B_TYPE(R2)
0E A2 01  B0 01D7 1094          MOVW    #AT$_UBA, -          ; SET TYPE OF ADAPTER
      01DB 1095          ADP$W_ADPTYPE(R2)
62 00E4'CF44 D0 01DB 1096          MOVL   W^SBICONF[R4], -      ; SET VA OF CONFIGURATION REG
      01E1 1097          ADP$L_CSR(R2)
0C A2 54  B0 01E1 1098          MOVW    R4,ADP$W_TR(R2)      ; SET TR NUMBER FOR ADAPTER
      01E5 1099
50 14 A2  DE 01E5 1100          MOVAL  ADP$L_DPQFL(R2),R0     ; ADDRESS OF DATA PATH WAIT QUEUE
      60 50  D0 01E9 1101          MOVL   R0,(R0)              ; INIT QUEUE HEADER
04 A0 50  D0 01EC 1102          MOVL   R0,4(R0)              ;
      01F0 1103
50 30 A2  DE 01F0 1104          MOVAL  ADP$L_MRQFL(R2),R0     ; ADDRESS OF MAP WAIT QUEUE
      60 50  D0 01F4 1105          MOVL   R0,(R0)              ; INIT QUEUE HEADER
04 A0 50  D0 01F7 1106          MOVL   R0,4(R0)              ;
      04 A2  D4 01FB 1107          CLRL  ADP$L_LINK(R2)        ; ZAP ADAPTER CHAIN LINK
      FDFD' 30 01FE 1108          BSBW   ADPLINK              ; LINK ADP TO END OF LIST
      0201 1109 ;
      0201 1110 ; Initialize adapter interrupt vectors in System Control Block.
      0201 1111 ;
00000000'GF D0 0201 1112          MOVL   G^EXE$GL_SCB,R8       ; GET SCB ADDRESS
      58
      0207
      0208 1113
      0208 1120
      0208 1122
      0208 1123 ;
      0208 1124 ; Following ASSUME breaks if the ADP length is not a multiple of 4, thereby
      0208 1125 ; causing the vectors to NOT be long word aligned.
      0208 1126
      0208 1127          ASSUME  ADP$C_UBAADPLEN/4*4  EQ  ADP$C_UBAADPLEN
      0208 1128
53 02EC'C2 9E 0208 1129          MOVAB  ADP$C_UBAADPLEN+UBINTSZ(R2),R3 ; LOCATE VECTORS
      10 A2 53  D0 020D 1130          MOVL   R3,ADP$L_VECTOR(R2)    ; AND RECORD IN ADP
      FFFE 8F  B0 0211 1131          MOVW   #^XFFFE,ADP$W_DPBITMAP(R2) ; MARK DATAPATHS 1-15 AVAILABLE
      60 A2
      0215
53 FF6C'C3 9E 0217 1132          MOVAB  -UBINTSZ(R3),R3        ; BASE OF INTERRUPT CODE
      3F  BB 021C 1133          PUSHR  #^M<R0,R1,R2,R3,R4,R5> ; SAVE MOVN REGISTERS
  
```

```

0094'8F 28 021E 1134      MOVCS  #UBINTSZ,WAUBAINTBASE,(R3)      ; COPY INTERRUPT CODE
63 0340'CF          0222
      3F BA 0226 1135      POPR  #AM<R0,R1,R2,R3,R4,R5>      ; RESTORE MOVCS REGISTERS
54 04 00 EF 0228 1136      EXTZV #0,#4,R4,R4      ; Use low 4 bits of nexus number.
      54          022C
50 0100 C844 DE 022D 1137      MOVAL AX100(R8)[R4],R0      ; COMPUTE 1ST VECTOR ADDRESS
      1C A2 50 DO 0233 1138      MOVL  R0,ADP$L_AVECTOR(R2)      ; SAVE ADDR OF ADAPTER SCB VECTORS
60 01'A3 9E 0237 1139      MOVAB B^UBAINT4+1(R3),(R0)      ; STORE VECTOR FOR BR4
40 A0 21'A3 9E 023B 1140      MOVAB B^UBAINT5+1(R3),64(R0)      ; STORE VECTOR FOR BR5
      41'A3 9E 0240 1141      MOVAB B^UBAINT6+1(R3),128(R0)      ; STORE VECTOR FOR BR6
      0080 C0          0243
      61'A3 9E 0246 1142      MOVAB B^UBAINT7+1(R3),192(R0)      ; STORE VECTOR FOR BR7
      00C0 C0          0249
      50 62 DO 024C 1143      MOVL  ADP$L_CSR(R2),R0      ; GET UBACSR ADDRESS
      0A'A3 50 C0 024F 1144      ADDL  R0,B^UBAINT4REL(R3)      ; ADD CSR VA
      2A'A3 50 C0 0253 1145      ADDL  R0,B^UBAINT5REL(R3)      ; TO EACH OF THE
      4A'A3 50 C0 0257 1146      ADDL  R0,B^UBAINT6REL(R3)      ; BICL INSTRUCTIONS
      6A'A3 50 C0 025B 1147      ADDL  R0,B^UBAINT7REL(R3)      ; IN THE INTERRUPT DISPATCHERS
0089'C3 52 DO 025F 1148      MOVL  R2,UBAINTADP(R3)      ; SET ADDRESS OF ADAPTOR CONTROL BLOCK
      0000'CF 9E 0264 1149      MOVAB W^EXE$UBAERR_INT,-      ;
      0090'C3          0268 1150      UBAERRADR(R3)      ; SET ADDRESS OF ERROR HANDLER
      01'A3 9E 026B 1151      MOVAB B^UBAINT4+1(R3),-      ;
      44 A2          026E 1152      ADP$L_UBASCB(R2)      ; SAVE 4 SCB VECTOR CONTENTS
      21'A3 9E 0270 1153      MOVAB B^UBAINT5+1(R3),-      ;
      48 A2          0273 1154      ADP$L_UBASCB+4(R2)      ; DITTO
      41'A3 9E 0275 1155      MOVAB B^UBAINT6+1(R3),-      ;
      4C A2          0278 1156      ADP$L_UBASCB+8(R2)      ; DITTO
      61'A3 9E 027A 1157      MOVAB B^UBAINT7+1(R3),-      ;
      50 A2          027D 1158      ADP$L_UBASCB+12(R2)      ; DITTO
      54 52 DO 027F 1159      MOVL  R2,R4      ; COPY ADP ADDRESS
      52 62 DO 0282 1160      MOVL  ADP$L_CSR(R2),R2      ; VIRTUAL ADDRESS OF ADAPTER
      7C000000 8F DO 0285 1161      MOVL  #^X7C000000,UBA$L_CR(R2); DISABLE ALL UMR'S
      04 A2          028B
      00000000'GF 16 028D 1162      JSB  G^MMG$SVAPTECHK      ; ADDRESS OF SPTE THAT MAPS ADAPTER
      54 A4 63 DO 0293 1163      MOVL  (R3),ADP$L_UBASPTA(R4)      ; SAVE CONTENTS OF SPTE MAPPING ADAPTER
58 A4 20 A3 DO 0297 1164      MOVL  <8*4>(R3),-      ; CONTENTS OF SPTE MAPPING I/O SPACE
      52 54 DO 029C 1165      ADP$L_UBASPTA+4(R4)
      00000000'GF DE 029F 1167      MOVL  R4,R2      ; COPY ADP ADDRESS BACK TO R2
      53          02A5      MOVAL  GAUBA$UNEXINT,R3      ; GET ADDR OF UNEXP INT SERVICE(IN EXEC)
54 0000'CF DE 02A6 1168      MOVAL  WAUBA$INT0,R4      ; GET ADDR OF SPECIAL VECTOR 0 ROUTINE
      02AB 1169
      02AB 1170 ;
      02AB 1171 ; INIT UB VECTORS TO UNEXPECTED INTERRUPT SERVICE
      02AB 1172 ;
      50 10 A2 DO 02AB 1173      MOVL  ADP$L_VECTOR(R2),R0      ; GET ADDRESS OF VECTORS
      80 54 DO 02AF 1174      MOVL  R4,(R0)+      ; SPECIAL CASE FOR VECTOR 0
51 7F 8F 9A 02B2 1175      MOVZBL #<NUMUBAVEC-1>,R1      ; REST OF VECTORS
      80 53 DO 02B6 1176      MOVL  R3,(R0)+      ; FILL VECTOR WITH UNEXP INT
      FA 51 F5 02B9 1177      SOBGR R1,10$      ; FILL ALL VECTORS
      02BC 1178
      02BC 1180
      02BC 1240
      02BC 1241
      02BC 1269
      02BC 1270
      02BC 1291

```

;RLRSCORPIO  
;RLRSCORPIO  
;RLRSCORPIO

```

02BC .1
02BC .43
02BC .44
02BC 1340 ;
02BC 1341 ; Now check for any UNIBUS memory that may be on the adapter. First we must
02BC 1342 ; disable all the UNIBUS Map Registers so that there is no conflict in
02BC 1343 ; which memory will respond. Then we check all 248Kb of potential memory in
02BC 1344 ; 8Kb chunks, since each disable bit on the 780 UBA represents 16 UMR's or
02BC 1345 ; 8Kb of memory. The number of registers is stored in the ADP and the
02BC 1346 ; corresponding number withdrawn from the UMR map in the ADP.
02BC 1347 ;
02BC 1348
56 62 D0 02BC 1350 MOVL ADP$L_CSR(R2),R6 ; Pick up adapter pointer
51 51 D4 02BF 1351 CLRL R1 ; Zero out number of UMR to disable
00000200 8F C3 02C1 1353 SUBL3 #512,8(SP),R7 ; R7 = VA of last page of UNIBUS
57 08 AE 02C7
0C AE 04 C3 02CA 1354 SUBL3 #4,12(SP),R8 ; R8 = VA of SPTe mapping (R7)
58 02CE
00000200 8F C3 02CF 1355 SUBL3 #512,32(SP),R4 ; R4 = PFN of first page of UNIBUS
54 20 AE 02D5
68 DD 02D8 1356 PUSHL (R8) ; Save contents of SPTe
53 54 D0 02DA 1357 MOVL R4,R3 ; Copy starting PFN
55 1F D0 02DD 1358 MOVL #31,R5 ; 31 8Kb chunks to test
02E0 1359 50$: INVALID R7 ; Invalidate TB
90000000 8F C9 02E3 1360 BISL3 #<PTE$M VALID!PTE$C_KW>,-
68 54 02E9 1361 R4,(R8) ; Map each page of UNIBUS
50 57 D0 02EB 1362 MOVL R7,R0 ; Address to check
FDOF 30 02EE 1363 BSBW EXE$TEST_CSR ; Validate it
0D 50 E9 02F1 1364 BLBC R0,70$ ; Not there
54 53 D1 02F4 1365 CMPL R3,R4 ; First time in?
04 13 02F7 1366 BEQL 60$ ; Yes, skip next test
51 D5 02F9 1367 TSTL R1 ; Any registers already?
3A 13 02FB 1368 BEQL 80$ ; No, memory not start at 0
51 10 A1 9E 02FD 1369 60$: MOVAB 16(R1),R1 ; Yes, up the count
54 10 A4 9E 0301 1370 70$: MOVAB 16(R4),R4 ; Map Next 8Kb (16*512)
D8 55 F5 0305 1371 SOBGTR R5,50$ ; Loop until done
68 8ED0 0308 1372 POPL (R8) ; Restore old contents of SPTe
030B 1373 INVALID R7 ; Invalidate TB
0256 C2 51 B0 030E 1375 MOVW R1,ADP$W_UMR_DIS(R2) ; Record number disabled
0313 1377 ;
0313 1378 ; Initialize fields for new UBA map register allocation. Make it appear
0313 1379 ; that we have one contiguous array of 496 available map registers.
0313 1380 ; To do this we set ADP$L_MRACTMDRS to one (the number of active
0313 1381 ; map register descriptors for distinct contiguous areas),
0313 1382 ; ADP$W_MRNREGARY(0) to 496 (i.e the number of registers in this
0313 1383 ; contiguous range) and ADP$FREGARY(0) to 0 (i.e. the first register
0313 1384 ; in the range is register 0).
0313 1385 ;
5C A2 01 D0 0313 1386 MOVL #1,ADP$L_MRACTMDRS(R2) ; 1 active map descriptor
01F0 8F 51 A3 0317 1387 SUBW3 R1,#496,ADP$W_MRNREGARY(R2); for a range of 496 registers
64 A2 031C
015E C2 51 B0 031E 1399 MOVW R1,ADP$W_MRFREGARY(R2) ; starting at register zero.
62 A2 01 AE 0323 1400 MNEGW #1,ADP$W_MRNFFENCE(R2) ; Also init "fences" which precede
015C C2 01 AE 0327 1401 MNEGW #1,ADP$W_MRFFENCE(R2) ; the two descriptor arrays.
032C 1402 ;
032C 1403 ; Initialize adapter hardware.
032C 1404 ;

```



INIADP780  
V03-021

- ADAPTER INITIALIZATION FOR VAX 11/780 3-JUN-1984 14:53:36 VAX-11 Macro V03-01 Page 25  
INI\$UBADP - BUILD ADP AND INITIALIZE UBA 19-OCT-1983 11:50:19 DISK\$VMSMASTER:[SYSLOA.SRC]INIADP(14)

```
54 62 D0 032C 1405      MOVL   ADP$_CSR(R2),R4      ; Get CSR address to init
      FCCE' 30 032F 1406      BSBW   UBA$INITIAL          ; And initialize adapter
01FF 8F BA 0332 1407      POPR   #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; Restore registers
      05 0336 1408      RSB                                ; Return
      0337 1409
      0337 1411 ;
      0337 1412 ; Error if UNIBUS memory not start at location 0
      0337 1413 ;
51 030D'CF 9E 0337 1414 80$:  MOVAB   W^BADUMR,R1          ; Set error message
      FE5F 31 033C 1415      BRW     ERROR_HALT_1        ; Put it out
      033F 1417
```

;RLRSCORPIO  
 ;RLRSCORPIO  
 ;RLRSCORPIO  
 -2

```

033F 1420 ;
033F 1421 ; UBA INTERRUPT SERVICE ROUTINES. ONE COPY OF THESE ROUTINES IS
033F 1422 ; MOVED INTO NONPAGED POOL AND RELOCATED FOR EACH UBA.
033F 1423 ;
033F 1424 ; **** NOTE **** THE CODING SEQUENCE FOR DISPATCHING ON UBA INTERRUPTS
033F 1425 ; IS ASSUMED IN THE MODULE MCHECK780.MAR. THE ASSUMPTIONS ARE MADE SO
033F 1426 ; THE MACHINE CHECK HANDLER CAN IDENTIFY A CPU TIMEOUT WHEN THE
033F 1427 ; BICL3 INSTRUCTION IS READING THE UBA'S BRRVR REGISTER.
033F 1428 ; THE ASSUMPTIONS MADE ARE THAT THE VALUE OF THE VIRTUAL ADDRESS OF THE BRRVR
033F .1 ; REGISTER IS AT AN OFFSET OF 10. BYTES PAST THE INTERRUPT VECTOR ENTRY POINT,
033F .2 ; THAT THE PC OF THE INSTRUCTION ACCESSING BRRVR IS 3 BYTES PAST THE INTERRUPT
033F .3 ; VECTOR ENTRY, AND THAT R4 AND R5 ARE SAVED ON THE STACK AT THAT POINT.
033F 1431 ;
033F 1432 ;
033F 1433 .ENABL LSB
033F 1434 .ALIGN QUAD
0340 1435 UBAINTBASE= ; BASE OF UBA INTERRUPT DISPATCHERS
0340 1436 UBAINT4=-UBAINTBASE ; UBA 0 INTERRUPT DISPATCH LEVEL 4
0340 1437 MOVQ R4,-(SP) ; SAVE REGISTERS
7FFF0E03 8F CB 0343 1438 BICL3 #^X7FFFFE03,@#UBA$L_BRRVR,R4 ; READ VECTOR REGISTER AND CLEAR BITS
00000030 9F 0349
54 034E
0000000A 034F 1439 UBAINT4REL=-UBAINTBASE-5 ; OFFSET TO ADD UBACSR VALUE
7E 52 7D 034F 1440 MOVQ R2,-(SP) ; SAVE REGISTERS
55 D4'AF44 9E 0352 1441 MOVAB B^VECTAB[R4],R5 ; GET ADDRESS OF INTERRUPT VECTOR
65 18 0357 1442 BGEQ 10$ ; IF GEQ UBA INTERRUPTS
7E 50 7D 0359 1443 MOVQ R0,-(SP) ; SAVE REGISTERS
95 17 035C 1444 JMP @(R5)+ ; DISPATCH INTERRUPT
035E 1445 .ALIGN QUAD
00000020 0360 1446 UBAINT5=-UBAINTBASE ; UBA 0 INTERRUPT DISPATCH LEVEL 5
7E 54 7D 0360 1447 MOVQ R4,-(SP) ; SAVE REGISTERS
7FFF0E03 8F CB 0363 1448 BICL3 #^X7FFFFE03,@#UBA$L_BRRVR+4,R4 ; READ VECTOR REGISTER AND CLEAR BITS
00000034 9F 0369
54 036E
0000002A 036F 1449 UBAINT5REL=-UBAINTBASE-5 ; OFFSET TO ADD UBACSR VALUE
7E 52 7D 036F 1450 MOVQ R2,-(SP) ; SAVE REGISTERS
55 D4'AF44 9E 0372 1451 MOVAB B^VECTAB[R4],R5 ; GET ADDRESS OF INTERRUPT VECTOR
45 18 0377 1452 BGEQ 10$ ; IF GEQ UBA INTERRUPTS
7E 50 7D 0379 1453 MOVQ R0,-(SP) ; SAVE REGISTERS
95 17 037C 1454 JMP @(R5)+ ; DISPATCH INTERRUPT
037E 1455 .ALIGN QUAD
00000040 0380 1456 UBAINT6=-UBAINTBASE ; UBA 0 INTERRUPT DISPATCH LEVEL 6
7E 54 7D 0380 1457 MOVQ R4,-(SP) ; SAVE REGISTERS
7FFF0E03 8F CB 0383 1458 BICL3 #^X7FFFFE03,@#UBA$L_BRRVR+8,R4 ; READ VECTOR REGISTER AND CLEAR BITS
00000038 9F 0389
54 038E
0000004A 038F 1459 UBAINT6REL=-UBAINTBASE-5 ; OFFSET TO ADD UBACSR VALUE
7E 52 7D 038F 1460 MOVQ R2,-(SP) ; SAVE REGISTERS
55 D4'AF44 9E 0392 1461 MOVAB B^VECTAB[R4],R5 ; GET ADDRESS OF INTERRUPT VECTOR
25 18 0397 1462 BGEQ 10$ ; IF GEQ UBA INTERRUPTS
7E 50 7D 0399 1463 MOVQ R0,-(SP) ; SAVE REGISTERS
95 17 039C 1464 JMP @(R5)+ ; DISPATCH INTERRUPT
039E 1465 .ALIGN QUAD
00000060 03A0 1466 UBAINT7=-UBAINTBASE ; UBA 0 INTERRUPT DISPATCH LEVEL 7
7E 54 7D 03A0 1467 MOVQ R4,-(SP) ; SAVE REGISTERS
7FFF0E03 8F CB 03A3 1468 BICL3 #^X7FFFFE03,@#UBA$L_BRRVR+12,R4 ; READ VECTOR AND CLEAR BITS
0000003C 9F 03A9
  
```

```

54      03AE
0000006A 03AF 1469 UBAIN7REL=-UBAINTBASE-5      ; OFFSET TO ADD UBACSR VALUE
7E      52 7D 03AF 1470      MOVQ   R2,-(SP)      ; SAVE REGISTERS
55  D4'AF44 9E 03B2 1471      MOVAB  BAVECTAB[R4],R5      ; GET ADDRESS OF INTERRUPT VECTOR
      05 18 03B7 1472      BGEQ   10$      ; IF GEQ UBA INTERRUPTS
7E      50 7D 03B9 1473      MOVQ   R0,-(SP)      ; SAVE REGISTERS
      95 17 03BC 1474      JMP    @(R5)+      ; DISPATCH INTERRUPT
00 54 1F E5 03BE 1475 10$:  BBCC   #31,R4,20$      ;CLEAR ADAPTER ERROR INTERRUPT FLAG (MSB)
55  D4'AF44 9E 03C2 1476 20$:  MOVAB  BAVECTAB[R4],R5      ;GET ADDRESS OF INTERRUPT VECTOR
00000000'8F D0 03C7 1477      MOVL   I^#0,R4      ;GET ADDRESS OF ADAPTOR CONTROL BLOCK
      54      03CD
00000089 03CE 1478 UBAIN7ADP=-UBAINTBASE-5      ;OFFSET TO START OF LOADED CODE
00000000 9F 17 03CE 1479      JMP    @#0      ;ERROR ROUTINE IN ADPERR780
00000090 03D4 1480 UBAERRADR=-UBAINTBASE-4
      03D4 1481      .DSABL  LSB
      03D4 1482
      03D4 1483      .ALIGN  LONG      ; LONGWORD ALIGN VECTORS
      03D4 1484 VECTAB:
00000094 03D4 1485 UBINTSZ=-UBAINTBASE      ; END OF INTERRUPT CODE, START OF VECTORS
      03D4 1486      ; SIZE OF UBA INTERRUPT CODE

```

```

03D4 1503      .SBTTL INI$MBADP - BUILD ADP AND INITIALIZE MBA
03D4 1504      .SBTTL INI$DRADP - BUILD ADP AND INITIALIZE DR32
03D4 1505      .SBTTL INI$CIADP - BUILD ADP AND INITIALIZE CI
03D4 1506      ;+
03D4 1507      ; INI$MBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
03D4 1508      ; AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED. A CRB AND IDB ARE
03D4 1509      ; ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
03D4 1510      ; BY CALLING MBA$INITIAL.
03D4 1511      ;
03D4 1512      ; INI$DRADP IS CALLED AFTER MAPPING THE REGISTERS FOR THE DR32
03D4 1513      ; ADAPTER. THE ADAPTER CONTROL BLOCK, CRB, AND IDB ARE ALLOCATED
03D4 1514      ; AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED BY
03D4 1515      ; CALLING DR$INITIAL.
03D4 1516      ;
03D4 1517      ; INI$MBADP AND INI$DRADP SHARE COMMON CODE AFTER THE TABLE OF ADAPTER
03D4 1518      ; SPECIFIC CONSTANTS IS SELECTED AND STORED IN R8.
03D4 1519      ;
03D4 1520      ; INPUT:
03D4 1521      ;      R4 - nexus identification number of this adapter
03D4 1522      ;      R11- offset from beginning of SCB to correct SCB page for this adapter
03D4 1523      ;
03D4 1524      ; OUTPUTS:
03D4 1525      ;      ALL REGISTERS PRESERVED
03D4 1526      ;-
03D4 1527
00000000'GF 17 03D4 1528 ALONPAGD:JMP      G^INI$ALONONPAGED
03DA 1529
03DA 1530      .ENABL  LSB
03DA 1531
03DA 1532 INI$DRADP:                ; INITIALIZE DR32 DATA STRUCTURES
03DA 1533
03DA 1536      PUSHR  #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
58 0011'CF  DE 03DE 1537      MOVAL  W^DRTAB,R8                ; GET DR32 TABLE OF CONSTANTS
59 0000'CF  9E 03E3 1538      MOVAB  W^DR$INT,R9                ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A 0000'CF  9E 03E8 1539      MOVAB  W^DR$INITIAL,R10            ; ADDRESS OF DEVICE INITIALIZATION
      28 11 03ED 1540      BRB    10$                ; JOIN COMMON CODE
03EF 1543
03EF 1544 INI$CIADP:                ; INITIALIZE CI DATA STRUCTURES
03EF 1545
03EF 1548      PUSHR  #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ; SAVE REGISTERS
58 0015'CF  DE 03F3 1549      MOVAL  W^ACITAB,R8                ; GET CI TABLE OF CONSTANTS
59 0000'CF  9E 03F8 1550      MOVAB  W^ACI$INT,R9                ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A 0000'CF  9E 03FD 1551      MOVAB  W^ACI$INITIAL,R10            ; ADDRESS OF DEVICE INITIALIZATION
      13 11 0402 1552      BRB    10$                ; JOIN COMMON CODE
0404 1555
0404 1556 INI$MBADP:                ; INIT MBA DATA STRUCTURES
0404 1557
0404 1560      PUSHR  #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10> ;
58 0000'CF  DE 0408 1561      MOVAL  W^AMBATAB,R8                ; GET MBA TABLE OF CONSTANTS
59 0000'CF  9E 040D 1562      MOVAB  W^AMBA$INT,R9                ; ADDRESS OF INITERRUPT SERVICE ROUTINE
5A 0000'CF  9E 0412 1563      MOVAB  W^AMBA$INITIAL,R10            ; ADDRESS OF DEVICE INITIALIZATION
0417 1564 10$:
0417 1565 ;
0417 1566 ; Allocate and initialize Channel Request Block.
0417 1567 ;
51 0048 8F 3C 0417 1568      MOVZWL #CRB$C_LENGTH,R1            ; SET SIZE OF CRB
      B6 10 041C 1569      BSBB  ALONPAGD                ; ALLOCATE SPACE FOR CRB

```

```

08 A2 51 B0 041E 1570      MOVW   R1,CRB$W_SIZE(R2)      ; SET CORRECT SIZE
0A A2 05 90 0422 1571      MOVB   #DYN$C_CRB,CRB$B_TYPE(R2) ; SET CORRECT TYPE
      62 62 DE 0426 1572      MOVAL  CRB$L_WQFL(R2),CRB$L_WQFL(R2) ; INITIALIZE WAIT QUEUE HEADER
04 A2 62 DE 0429 1573      MOVAL  CRB$L_WQFL(R2),CRB$L_WQBL(R2) ; FLINK AND BLINK
50 24 A2 9E 042D 1574      MOVAB  CRB$L_INTD(R2),R0      ; SET ADDRESS OF INTD AREAD
9F163CBB 8F D0 0431 1575      MOVL   #^X9F163CBB,(R0)+      ; "PUSHR ^M<R2,R3,R4,R5>,JSB @#"
      80      0437
      80 59 D0 0438 1576      MOVL   R9,(R0)+              ; ADDR OF XXX$INT ROUTINE
      80 D4 043B 1577      CLRL  (R0)+                  ; CLEAR OUT UNNEEDED AREA
      60 5A D0 043D 1578      MOVL   R10,(R0)              ; ADDR OF XXX$INITIAL ROUTINE
      5A 52 D0 0440 1579      MOVL   R2,R10                ; SAVE CRB ADDRESS
      0443 1580 ;
      0443 1581 ; Allocate and initialize Interrupt Dispatch Block.
      0443 1582 ;
      51 68 9A 0443 1583      MOVZBL ADPTAB_IDBUNITS(R8),R1 ; GET # OF IDB UNITS
00000038 9F41 DE 0446 1584      MOVAL  @#IDB$C_LENGTH[R1],R1 ; GET TOTAL SIZE OF IDB
      51      044D
      84 10 044E 1585      BSBB  ALONPAGD                ; ALLOCATE SPACE FOR CRB
08 A2 51 B0 0450 1586      MOVW   R1,IDB$W_SIZE(R2)      ; SET STRUCTURE SIZE
0A A2 09 90 0454 1587      MOVB   #DYN$C_IDB,-          ; AND TYPE CODE
      0458 1588      IDB$B_TYPE(R2)
      68 9B 0458 1589      MOVZBW ADPTAB_IDBUNITS(R8),- ; SET COUNT OF UNITS
      0C A2 045A 1590      IDB$W_UNITS(R2)
62 00E4'CF44 D0 045C 1591      MOVL   W^ASBICONF[R4],-      ; SET CSR ADDRESS TO
      0462 1592      IDB$L_CSR(R2)              ; START OF ADAPTER REG SPACE
      2C AA 52 D0 0462 1593      MOVL   R2,-                  ; SET ADDRESS OF IDB INTO CRB
      0466 1594      CRB$L_INTD+VEC$L_IDB(R10)
      59 52 D0 0466 1595      MOVL   R2,R9                ; SAVE ADDRESS OF IDB
      0469 1596 ;
      0469 1597 ; Allocate and initialize Adapter Control Block (ADP).
      0469 1598 ;
      51 01 A8 3C 0469 1599      MOVZWL ADPTAB_ADPLEN(R8),R1 ; GET SIZE OF ADAPTER
      FF64 30 046D 1600      BSBW  ALONPAGD                ; ALLOCATE SPACE FOR CRB
08 A2 51 B0 0470 1601      MOVW   R1,ADP$W_SIZE(R2)      ; SET SIZE OF STRUCTURE
0A A2 01 90 0474 1602      MOVB   #DYN$C_ADP,ADP$B_TYPE(R2); AND TYPE CODE
      62 69 D0 0478 1603      MOVL  IDB$L_CSR(R9),ADP$L_CSR(R2); SET ADDRESS OF CONFIGURATION REGISTER
0C A2 54 B0 047B 1604      MOVW   R4,ADP$W_TR(R2)        ; SET TR/SLOT-16 NUMBER OF ADAPTER
      03 A8 9B 047F 1605      MOVZBW ADPTAB_ATYPE(R8),- ; SET THE ADAPTER TYPE
      0E A2 0482 1606      ADP$W_ADPTYPE(R2)
      10 A2 5A D0 0484 1607      MOVL  R10,ADP$L_CRB(R2)      ; POINT ADP TO CRB
      0488 1608 ;
      0488 1609 ;
      0488 1610 ;
      0488 1611 ; Initialize adapter interrupt vectors in System Control Block.
      0488 1612 ;
00000000'GF D0 0488 1613      MOVL  G^AEXE$GL_SCB,R0      ; GET ADDRESS OF SCB
      50      048E
      55 5B 09 78 048F 1614      ASHL  #9,R11,R5              ; Turn SCB page offset into byte offset.
      50 55 C0 0493 1615      ADDL  R5,R0                  ; set to beginning of correct SCB page.
      54 04 00 EF 0496 1616      EXTZV #0,#4,R4,R4          ; Use low 4 bits of nexus number.
      54      049A
      50 0100 C044 DE 049B 1617      MOVAL  ^X100(R0)[R4],R0      ; COMPUTE ADDR OF 1ST VECTOR
      1C A2 50 D0 04A1 1618      MOVL  R0,ADP$L_AVECTOR(R2) ; SAVE ADDR OF ADAPTER'S SCB VECTORS
      60 25 AA DE 04A5 1619      MOVAL  CRB$L_INTD+1(R10),(R0) ; CONNECT VECTOR TO CRB CODE
      40 A0 25 AA DE 04A9 1620      MOVAL  CRB$L_INTD+1(R10),64(R0); SAME FOR
      25 AA DE 04AE 1621      MOVAL  CRB$L_INTD+1(R10),128(R0); ALL FOUR
      0080 C0 04B1

```

```

    25 AA  DE 04B4 1622      MOVAL  CRB$_INTD+1(R10),192(R0); VECTORS
00C0 C0      04B7
                04BA 1623 ;
                04BA 1624 ; Continue with ADP initialization.
                04BA 1625 ;
14 A2  25 AA  DE 04BA 1626 20$:  MOVAL  CRB$_INTD+1(R10), - ; SAVE SCB VECTOR CONTENTS IN ADP
                04BF 1627      ADP$_MBASCB(R2)
                04BF 1628      PUSHR  #^M<R2,R3> ; SAVE SOME REGISTERS
                55 52  DO 04C1 1629      MOVL  R2,R5 ; COPY ADP ADDRESS
                52 62  DO 04C4 1630      MOVL  ADP$_CSR(R2),R2 ; VIRTUAL ADDRESS OF ADAPTER
00000000 GF 16 04C7 1631      JSB   G^MMG$SVAPTECHK ; ADDRESS OF SPTE THAT MAPS ADAPTER
18 A5  63  DO 04CD 1632      MOVL  (R3),ADP$_MBASPTE(R5) ; SAVE CONTENTS OF SPTE
                0C  BA 04D1 1633      POPR  #^M<R2,R3> ; RESTORE REGISTERS
38 AA  52  DO 04D3 1634      MOVL  R2,CRB$_INTD+VEC$_ADP(R10) ; SET CRB POINTER TO ADP
14 A9  52  DO 04D7 1635      MOVL  R2,IDB$_ADP(R9) ; AND INTO IDB
                FB22 30 04DB 1636      BSBW  ADPLINK ; LINK ADP TO END OF CHAIN
                04DE 1637 ;
                04DE 1638 ; Initialize adapter hardware.
                04DE 1639 ;
55 59  DO 04DE 1640      MOVL  R9,R5 ; ADDRESS OF IDB
54 65  DO 04E1 1641      MOVL  IDB$_CSR(R5),R4 ; ADDRESS OF CONFIGURATION REGISTER 0
30 BA  16 04E4 1642      JSB   @CRB$_INTD+VEC$_INITIAL(R10) ; INIT ADAPTER
07FF 8F  BA 04E7 1643      POPR  #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10>; RESTORE ALL REGISTERS
                05 04EB 1644      RSB   ; RETURN
                04EC 1645
                04EC 1646      .DSABL LSB

```

INIADP780  
V03-021

- ADAPTER INITIALIZATION FOR VAX 11/780 3-JUN-1984 14:53:36 VAX-11 Macro V03-01 Page 31  
INI\$KDZ11 19-OCT-1983 11:50:19 DISK\$VMSMASTER:[SYSLOA.SRC]INIADP(14)

```
;RLRSCORPIO      04EC .36      .SBTTL INI$KDZ11
;RLRSCORPIO      04EC .37 ;++
;RLRSCORPIO      04EC .38 ;
;RLRSCORPIO      04EC .39 ; INPUTS:
;RLRSCORPIO      04EC .40 ;      R2 - VA of next free system page
;RLRSCORPIO      04EC .41 ;      R3 - VA of system page table entry to be used to map VA in R2
;RLRSCORPIO      04EC .42 ;      R4 - nexus identification number of this adapter
;RLRSCORPIO      04EC .43 ;
;RLRSCORPIO      04EC .44 ; OUTPUTS:
;RLRSCORPIO      04EC .45 ;
;RLRSCORPIO      04EC .46 ;--
;RLRSCORPIO      04EC .47
;RLRSCORPIO      04EC .48 INI$KDZ11:
;RLRSCORPIO      04EC .49
;RLRSCORPIO      05 04EC .68      RSB      ; Return to caller.
```

```

04ED 1650          .SBTTL INI$CONSOLE, init data structures for console
04ED 1651 ;++
04ED 1652 ; FUNCTIONAL DESCRIPTION:
04ED 1653 ;
04ED 1654 ;         This routine is executed only once, during system initialization.
04ED 1655 ;         It initializes the CRB and IDB for boot/console device.
04ED 1656 ;
04ED 1657 ;         This routine is called from INIT.
04ED 1658 ;
04ED 1659 ; INPUTS:
04ED 1660 ;
04ED 1661 ;         R3 --> DISK [CLASS] DRIVER DDB
04ED 1662 ;         R4 --> DISK [CLASS] DRIVER DPT
04ED 1663 ;         R5 --> DISK [CLASS] DRIVER UCB
04ED 1664 ;         R6 --> RPB
04ED 1665 ;         R7 --> ADP FOR EITHER A REAL DISK OR A PORT
04ED 1666 ;         R9 --> PORT DRIVER DPT (IF PRESENT)
04ED 1667 ;         R10--> PORT DIRVER UCB (IF PRESENT)
04ED 1668 ;
04ED 1669 ;--
04ED 1670
04ED 1671 INI$CONSOLE::
04ED 1672          .ENABL  LSB
04ED 1673
    66 A6  91 04ED 1675          CMPB   RPB$B_DEV TYP(R6),-      ; BOOTING FROM CONSOLE BLOCK
    40 8F          04F0 1676          #BTD$K_CONSOLE      ; STORAGE DEVICE?
    12          12 04F2 1677          BNEQ   BLD_CRB          ; NO
41534303 8F  D0 04F4 1678          MOVL   #AA/CSA/@8+3,-    ; YES, SET DEVICE NAME
    14 A3          04FA 1679          DDB$_NAME(R3)        ; COUNTED STRING
    04FC 1681
;RLRSCORPIO 00000000'9F DE 04FC      .2          MOVAL  @#OPA$CRB,R8      ; SET ADDRESS OF CRB
    58
;RLRSCORPIO 0068          31 0503      .3          BRW   100$
;RLRSCORPIO
-12          0506      .5
          0506 1694
          0506 1695 ;
          0506 1696 ; NOW BUILD THE AUXILIARY DATA BLOCKS (CRB,IDB)
          0506 1697 ;
          0506 1698 BLD_CRB:
    58 10 A7  D0 0506 1699          MOVL   ADP$_CRB(R7),R8      ; GET ADDRESS OF CRB IF IT EXISTS
    0E A7  01  B1 050A 1700          CMPW   #AT$_UBA,ADP$_W_ADPTYPE(R7); IS THIS A UNIBUS ADAPTER?
    03          13 050E 1701          BEQL   FILL_CRB          ; YES, ALLOCATE CRB
    005B      31 0510 1702          BRW   100$              ; NO, CRB/IDB ALREADY ALLOCATED
    0513 1703
    0513 1704 FILL_CRB:
00000000'9F 16 0513 1705          JSB   @#INI$ALLOC_CRB    ; GO ALLOCATE AND SETUP CRB
9F163FBB 8F  D0 0519 1706          MOVL   #^X9F163FBB,CRB$_INTD(R2) ; SET PUSHR #^M<R0,...R5>
    24 A2          051F
    0521 1707          ; JSB @#0 INTO INTERRUPT DISPATCH
    38 A2  57  D0 0521 1708          MOVL   R7,CRB$_INTD+VEC$_ADP(R2) ; SET POINTER TO ADP
    58 52  D0 0525 1709          MOVL   R2,R8            ; SAVE CRB POINTER
51 0058 8F  3C 0528 1710          MOVZWL #<IDB$_LENGTH+<8*4>>,R1 ; SIZE TO ALLOCATE FOR IDB
00000000'9F 16 052D 1711          JSB   @#INI$ALONONPAGED ; ALLOCATE IDB
    08 A2  51  B0 0533 1712          MOVW   R1,IDB$_W_SIZE(R2) ; SET SIZE OF IDB
    0A A2  09  90 0537 1713          MOVB   #DYN$_IDB,IDB$_B_TYPE(R2); AND STRUCTURE TYPE CODE
    2C A8  52  D0 053B 1714          MOVL   R2,CRB$_INTD+VEC$_IDB(R8) ; SET IDB INTO CRB
    053F      .1

```

;TCM0009



-1

```

        053F 1736
    62 58 A6 D0 053F 1737 10$:   MOVL   RPB$L_CSRVIR(R6), -      ; SAVE BOOT DEVICE CSR ADDRESS
        0543 1738                IDB$L_CSR(R2)                ; IN INTERRUPT DISPATCH BLOCK
        11 91 0543 1739         CMPB   #BTD$K_UDA, -          ; LOW ORDER BYTE OF ORIGINAL R0 TELLS
    66 A6 08 12 0545 1740         RPB$B_DEVTYP(R6)           ; BOOT DEVICE TYPE.
        0547 1741         BNEQ   20$                          ; IF NOT BOOTING FROM A UDA BRANCH
        0549 1742                ; AROUND.
    58 A6 D0 0549 1743         MOVL   RPB$L_CSRVIR(R6), -      ; COPY VIRTUAL ADDRESS OF UDA PORT CSR
00000000'9F 054C
        0551 1744                @#BOO$GB_SYSTEMID          ; TO LOW ORDER LONGWORD OF SYSTEMID
        0551 1745 20$:
    14 A2 57 D0 0551 1746         MOVL   R7, IDB$L_AD(P(R2))    ; POINT IDB TO ADP
    50 1E A6 3C 0555 1747         MOVZWL RPB$W_ROUBVEC(R6), R0    ; GET USER SPECIFIED VECTOR
        0A 12 0559 1748         BNEQ   30$                          ; BRANCH IF VECTOR SPECIFIED
    50 66 A6 9A 055B 1749         MOVZBL RPB$B_DEVTYP(R6), R0    ; ELSE GET DEVICE TYPE CODE
    50 FFFE'CF40 3C 055F 1750         MOVZWL W^BOOTVECTOR-2[R0], R0    ; GET DEFAULT INTERRUPT VECTOR
    50 10 B740 9E 0565 1751 30$:   MOVAB  @ADP$L_VECTOR(R7)[R0], R0    ; COMPUTE ADDRESS OF VECTOR
    60 26 A8 9E 056A 1752         MOVAB  CRB$L_INTD+2(R8), (R0)    ; SET ADDR OF INTERRUPT VECTOR
        056E 1753                ;
        056E 1757                ;
        056E 1758 100$:
    05 056E 1759         RSB                                ; RETURN
        056F 1760         .DISABLE LSB

```

```

056F 1762      .SBTTL  EXE$INI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
056F 1763 ;++
056F 1764 ; FUNCTIONAL DESCRIPTION:
056F 1765 ;
056F 1766 ; EXE$INI_TIMWAIT initializes EXE$GL_TENUSEC and EXE$GL_UBDELAY, cells used
056F 1767 ; in the time-wait macros. The first data cell, EXE$GL_TENUSEC, is the number
056F 1768 ; of times the following loop will be executed in ten u-seconds. This is
056F 1769 ; done once here to calibrate the loop instead of reading the processor clock.
056F 1770 ; The resulting number is used in the system macros TIMEWAIT and TIMEDWAIT.
056F 1771 ;
056F 1772 ; The first step is to initialize EXE$GL_UBDELAY. If the bit test instruction
056F 1773 ; in the TIMEWAIT macro is executed too rapidly in a loop, it can saturate the
056F 1774 ; Unibus. EXE$GL_UBDELAY is used to introduce a 3 microsecond delay loop into
056F 1775 ; the TIMEWAIT bit test loop.
056F 1776 ;
056F 1777 ; This routine is called only once, from INIT.
056F 1778 ;
056F 1779 ; INPUT PARAMETERS:
056F 1780 ;
056F 1781 ;     NONE
056F 1782 ;
056F 1783 ; IMPLICIT INPUTS:
056F 1784 ;
056F 1785 ;     Time-of-day processor clock.
056F 1786 ;     Interval timers.
056F 1787 ;
056F 1788 ; OUTPUT PARAMETERS:
056F 1789 ;
056F 1790 ;     R0 - Destroyed.
056F 1791 ;
056F 1792 ; IMPLICIT OUTPUTS:
056F 1793 ;
056F 1794 ;     EXE$GL_TENUSEC - set to appropriate value to make TIMEWAIT and TIMEDWAIT
056F 1795 ;                   macros loop for 10 micro-seconds.
056F 1796 ;
056F 1797 ;     EXE$GL_UBDELAY - set to appropriate value to make TIMEWAIT and TIMEDWAIT
056F 1798 ;                   macros loop for 3 micro-seconds in the unibus delay
056F 1799 ;                   loop.
056F 1800 ;
056F 1801 ;--
056F 1802
056F 1803 EXE$INI_TIMWAIT::                                ; Initialize time-wait data cells
056F 1804     .ENABLE LSB
056F 1805
19 00 DA 056F 1808     MTPR    #0,#PR780$_NICR                ; Initialize next interval count register.
0572 1810
0572 1814
0572 1818
00004E20 8F D0 0572 1822     MOVL    #20000,-(SP)                ; # of times to execute timed loop.
0578
18 11 DA 0579 1824     MTPR    #^X11,#PR$_ICCS                ; Start clock, no interrupts.
057C 1825
057C 1826 ; * * * start of loop to time * * *
FD 6E F5 057C 1827 10$:  SOBGTR  (SP),10$                ; Delay loop.
057F 1828 ; * * * end of loop to time * * *
057F 1829

```

```

50 1A DB 057F 1831 MFPR #PR780$_ICR,R0 ; Read total time to execute loop.
      0582 1833
      0582 1837
      0582 1841
      0582 1845
18 00 DA 0582 1846 MTPR #0,#PR$_ICCS ; Shut off clock.
      50 C7 0585 1847 DIVL3 R0,#60000,G^EXE$GL_UBDELAY; Calculate number of times through
0000EA60 8F 0587
00000000'GF 058C
00000000'GF D6 0591 1848 INCL G^EXE$GL_UBDELAY ; loop to delay 3 microseconds.
      0597 1849
19 00 DA 0597 1851 MTPR #0,#PR780$_NICR ; Initialize next interval count register.
      059A 1853
      059A 1857
      059A 1861
      059A 1865
00004E20 8F D0 059A 1866 MOVL #20000,R0 ; Number of times to execute test loop
      50 05A0
00000000'GF D0 05A1 1867 MOVL G^EXE$GL_UBDELAY,(SP) ; Get delay loop iteration count.
      6E 05A7
18 11 DA 05A8 1868 MTPR #^X11,#PR$_ICCS ; Start clock, no interrupts
      05AB 1869
      05AB 1870 ; **** Start of loop to time
8000 8F B3 05AB 1871 20$: BITW #^X8000,40$ ; Random BITx instruction to time
000005B9'EF 05AF
      03 12 05B4 1872 BNEQ 40$ ; Random conditional branch instruction
      FD 6E F5 05B6 1873 30$: SOBGTR (SP),30$ ; Delay 3 microseconds.
      EF 50 F5 05B9 1874 40$: SOBGTR R0,20$ ; Loop
      05BC 1875 ; **** End of loop to time
      05BC 1876
50 1A DB 05BC 1878 MFPR #PR780$_ICR,R0 ; Read total time to execute loop.
      05BF 1880
      05BF 1884
      05BF 1888
18 00 DA 05BF 1892 MTPR #0,#PR$_ICCS ; Shut clock off
      8E D5 05C2 1893 TSTL (SP)+ ; Pop delay loop index off stack.
      50 C7 05C4 1894 DIVL3 R0,#200000,G^EXE$GL_TENUSEC ; Calculate number of times to
00030D40 8F 05C6
00000000'GF 05CB
00000000'GF D6 05D0 1895 INCL G^EXE$GL_TENUSEC ; execute the loop to kill 10 u-secs.
      05 05D6 1896 RSB ; Return
      05D7 1897 .DISABLE LSB

```

```

05D7 1905          .SBTTL EXE$INIT_TODR - SET SYSTEM TIME TO CORRECT VALUE AT STARTUP
05D7 1906 ;++
05D7 1907 ; FUNCTIONAL DESCRIPTION:
05D7 1908 ;
05D7 1909 ;         EXE$INIT_TODR SOLICITS THE CORRECT TIME FROM THE OPERATOR IF NECESSARY,
05D7 1910 ;         CONVERTS THE ASCII RESPONSE TO BINARY FORMAT AND CALLS AN INTERNAL
05D7 1911 ;         ENTRY POINT OF THE $SETIME SYSTEM SERVICE TO SET THE NEW SYSTEM TIME
05D7 1912 ;         IN MEMORY WITHOUT MODIFYING THE CONTENTS OF THE SYSTEM DISK.
05D7 1913 ;
05D7 1914 ;         IF THE TIME WOULD NORMALLY BE SOLICITED FROM AN OPERATOR, BECAUSE
05D7 1915 ;         THE HARDWARE TIME OF YEAR CLOCK IS ZERO, THEN THE SYSGEN PARAMETER
05D7 1916 ;         "TPWAIT" IS CHECKED. IF IT IS ZERO, THEN IT IS ASSUMED THAT NO
05D7 1917 ;         OPERATOR IS PRESENT AND THE SYSTEM IS BOOTED USING THE LAST TIME
05D7 1918 ;         RECORDED IN THE SYSTEM IMAGE. IF THE PARAMETER IS NON ZERO THEN
05D7 1919 ;         THAT TIME IS USED AS THE MAXIMUM TIME TO WAIT BEFOR ASSUMING THAT
05D7 1920 ;         THERE IS NO OPERATOR AND BOOTING ANY WAY. IF THE PARAMETER IS
05D7 1921 ;         NEGATIVE, THE SYSTEM WILL WAIT FOREVER.
05D7 1922 ;
05D7 1923 ;         THIS ROUTINE IS CALLED ONLY ONCE, FROM SYSINIT OR STASYSGEN.
05D7 1924 ;
05D7 1925 ; INPUT PARAMETERS:
05D7 1926 ;
05D7 1927 ;         NONE
05D7 1928 ;
05D7 1929 ; IMPLICIT INPUTS:
05D7 1930 ;
05D7 1931 ;         TIME-OF-DAY PROCESSOR CLOCK.
05D7 1932 ;
05D7 1933 ; OUTPUT PARAMETERS:
05D7 1934 ;
05D7 1935 ;         R0,R1 - DESTROYED
05D7 1936 ;
05D7 1937 ; IMPLICIT OUTPUTS:
05D7 1938 ;
05D7 1939 ;         EXE$GQ_SYSTIME - SET TO CURRENT TIME IN 100 NANOSECOND UNITS SINCE
05D7 1940 ;         17-NOV-1858 00:00:00.
05D7 1941 ;
05D7 1942 ;--
05D7 1943 ;
05D7 1944 ;
05D7 1945 ; Stack storage offsets:
05D7 1946 ;
00000000 05D7 1947 TTCHAN = ^X00          ; CHANNEL FOR TERMINAL (LONGWORD)
00000004 05D7 1948 TTNAME = ^X04          ; STRING DESCRIPTOR FOR OPERATOR'S TERM
0000000C 05D7 1949 TMPDESC = ^X0C         ; TEMPORY STRING DESCRIPTOR (QUADWORD)
00000014 05D7 1950 INTIME = ^X14         ; INPUT TIME VALUE (QUADWORD)
0000001C 05D7 1951 LINBUF = ^X1C         ; INPUT LINE BUFFER (5 LONGWORDS)
00000014 05D7 1952 LINBUFSIZ = ^X14     ; (LENGTH OF LINE BUFFER IN BYTES)
05D7 1953 ;
05D7 1954 ;
05D7 1955 ; PURE DATA
05D7 1956 ;
05D7 1957 TERM_NAMADR:
30 41 50 4F 05D7 1958 .ASCII \OPAO\          ; DEVICE NAME FOR OPERATOR'S TERMINAL
00000004 05DB 1959 TERM_NAMSIZ = . - TERM_NAMADR
6C 61 76 6E 69 00 05DB 1960 TIMERR: .ASCII \invalid date/time\ ;
74 61 64 20 64 69 05E1

```

```

65 6D 69 74 2F 65 05E7
      11 05DB
      05ED 1961 TIMEPROMPT:
      33' 05ED 1962 .BYTE NPROMPT
41 45 4C 50 0A 0D 05EE 1963 .ASCII <13><10>/PLEASE ENTER DATE AND TIME (DD-MMM-YYYY HH:MM) /
54 4E 45 20 45 53 05F4
54 41 44 20 52 45 05FA
20 44 4E 41 20 45 0600
28 20 45 4D 49 54 0606
4D 4D 4D 2D 44 44 060C
20 59 59 59 59 2D 0612
4D 4D 3A 48 48 20 0618
      20 20 29 061E
      00000033 0621 1964 NPROMPT=.-TIMEPROMPT-1
      0621 1965
      0621 1966
      0621 1967 EXE$INIT_TODR:: ; SET CORRECT TIME
      0621 1968 .ENABLE LSB
      077C 8F BB 0621 1969 PUSHR #AM<R2,R3,R4,R5,R6,R8,R9,R10> ; SAVE REGISTERS
      5E 30 C2 0625 1970 SUBL #4*12,SP ; SCRATCH STORAGE
      56 5E D0 0628 1971 MOVL SP,R6 ; SAVE ADDRESS OF SCRATCH STORAGE
      04 A6 04 9A 062B 1972 MOVZBL #TERM_NAMSIZ,TTNAME(R6) ; SET SIZE OF OPERATOR'S TERM NAME AND
      FFA4 CF 9E 062F 1973 MOVAB WATERM_NAMADR,TTNAME+4(R6) ; PIC ADDRESS INTO TERM NAME DESC
      08 A6 0633
      00' E0 0635 1974 BBS S^EXE$V_SETTIME,G^EXE$GL_FLAGS,READTIME ; BR TO SOLICIT TIME
1C 00000000'GF 0637
      063D 1975
      063D 1976
      50 1B DB 063D 1978 MFPR #PR780$_TODR,R0 ; GET TIME OF DAY CLOCK VALUE
      0640 1980
      0640 1984
      0640 1988
      0640 1992
      50 C3 0640 1994 SUBL3 R0,G^EXE$GL_TODR,R9 ; GET TOD DELTA TIME (10 MS UNITS)
00000000'GF 0642
      59 0647
      09 1B 0648 1995 BLEQU 5$ ; BRANCH IF TIME IS LATER
      59 D1 064A 1996 CMPL R9,#24*60*60*100 ; CHECK FOR SETBACK OF ONE DAY
0083D600 8F 064C
      06 1E 0651 1997 BGEQU READTIME ; MORE, MUST SOLICIT TIME
      14 A6 7C 0653 2002 5$: CLRQ INTIME(R6) ; NULL ARGUMENT FOR EXE$SETIME_INT
      00C7 31 0656 2003 BRW 200$ ; RETURN TO CALLER
      0659 2004
      0659 2005 READTIME: ; SOLICIT TIME
      59 D4 0659 2006 CLRL R9 ; CLEAR A FLAG
00000000'GF 32 065B 2007 CVTWL G^SGN$GW_TPWAIT,R8 ; PICK UP TIMEOUT WAIT INTERVAL
      58 0661
      14 14 0662 2008 BGTR 8$ ; POSITIVE, WAIT THAT PERIOD ONCE
      0D 19 0664 2009 BLSS 7$ ; NEGATIVE IS WAIT FOREVER
      0666 2010 6$:
      01 C1 0666 2012 ADDL3 #1,G^EXE$GL_TODR,R0 ; ZERO, SET TIME-OF-DAY CLOCK TO
00000000'GF 0668
      50 066D
      066E 2014
      1B 50 DA 066E 2016 MTPR R0,#PR780$_TODR ; KNOWN VALUE + 10 MSEC AND FINISH UP
      0671 2018
      0671 2022

```

;KDM0099  
;KDM0099  
-1

```

0671 2026
0671 2030
;KDM0099      E0 11 0671 .2      BRB      5$      ;
;KDM0099      0673 .4
-1            0673 2032
            58 14 D0 0673 2033 7$:      MOVL      #20,R8      ; STARTING WAIT
            59 D6 0676 2034      INCL      R9      ; NEGATIVE - WAIT FOREVER
            DD 50 E9 0678 2035 8$:      $ASSIGN_S      TTNAME(R6),TTCHAN(R6) ; AND ASSIGN TO INPUT DEVICE
52      FF60 CF 9E 0686 2036      BLBC      R0,6$      ; ERROR - FALL BACK TO STORED TIME
            53 82 9A 0689 2037 10$:      MOVAB     W^TIMEPROMPT,R2 ; GET ADDRESS OF PROMPT STRING
            068E 2038      MOVZBL    (R2)+,R3      ; AND LENGTH
            0691 2039      $QIOW_S   #0,W^TTCHAN(R6),- ; PROMPT AND READ TIME
            0691 2040      #<IO$ READPROMPT!IO$M_PURGE!IO$M_TIMED!IO$M_CVTLW>,-
            0691 2041      TMPDESC(R6),,,- ; I/O STATUS BLOCK, NO AST OR PARAM
            0691 2042      LINBUF(R6),#LINBUFSIZ,- ; BUFFER ADDRESS AND SIZE
            0691 2043      R8,#0,- ; TIME OUT
            0691 2044      R2,R3      ; PROMPT ADDRESS AND SIZE
            AD 50 E9 06B6 2045      BLBC      R0,6$      ; ERROR - FALL BACK TO STORED TIME
54      OC A6 7D 06B9 2046      MOVQ     TMPDESC(R6),R4 ; GET COMPLETION STATUS
            OD 54 E8 06BD 2047      BLBS     R4,20$      ; CONTINUE IF SUCCESSFUL READ
            A3 59 E9 06C0 2048      BLBC      R9,6$      ; FAILED ON ONE-TIME READ, RETURN
58      01 A848 9E 06C3 2049      MOVAB     1(R8)[R8],R8 ; (2 * TIMEOUT) + 1
            58 58 3C 06C8 2050      MOVZWL   R8,R8      ; BOUND TIMEOUT
            BC 11 06CB 2051      BRB      10$      ; TRY AGAIN FOR TIME
            OC A6 OE A6 3C 06CD 2052 20$:      MOVZWL   TMPDESC+2(R6),TMPDESC(R6) ; FORM DESCRIPTOR FOR BUFFER
10 A6 1C A6 9E 06D2 2054      MOVAB     LINBUF(R6),TMPDESC+4(R6) ; SET DESCRIPTOR ADDRESS
            06D7 2055      $BINTIM_S TMPDESC(R6),INTIME(R6) ; CONVERT TO BINARY TIME
            05 50 E9 06E4 2056      BLBC      R0,B9$      ; INVALID TIME
            18 A6 D5 06E7 2057      TSTL     INTIME+4(R6) ; CHECK FOR DELTA TIME
            2A 14 06EA 2058      BGTR     100$      ; BRANCH IF NOT - OK
52      FEEB CF 9E 06EC 2059 89$:      MOVAB     W^TIMERR,R2 ; ADDRESS OF ERROR MESSAGE
            53 82 9A 06F1 2060      MOVZBL    (R2)+,R3      ; GET STRING LENGTH
            06F4 2062      $QIOW_S   #0,TTCHAN(R6),- ; GIVE ERROR MESSAGE
            06F4 2063      #IO$_WRITEVBLK,- ;
            06F4 2064      ,,,- ; NO I/O STATUS,AST OR AST PARAM
            06F4 2065      (R2),R3 ,- ; BUFFER ADDRESS, LENGTH
            06F4 2066      #0,#32 ; SET CARRIAGE CONTROL TO CR/LF
            FF73 31 0713 2067      BRW      10$      ; AND TRY AGAIN
            0716 2068 100$:      ; EXIT
            0716 2069      $DASSGN_S TTCHAN(R6) ; DE-ASSIGN TERMINAL CHANNEL
            14 A6 7F 0720 2070 200$:      PUSHAQ   INTIME(R6) ; SET NEW SYSTEM TIME
            01 FB 0723 2071      CALLS    #1,G^EXE$SETIME_INT ; USE TODR CLOCK TO SET SYSTEM TIME
00000000'GF 0725
00000000'GF 7D 072A 2072      MOVQ     G^EXE$GQ_TODCBASE,G^EXE$GQ_BOOTTIME ; SAVE BOOT TIME
00000000'GF 0730
            5E 30 CO 0735 2073      ADDL     #12*4,SP      ; CLEAN OFF SCRATCH STORAGE
            077C 8F BA 0738 2074      POPR     #^M<R2,R3,R4,R5,R6,R8,R9,R10> ; RESTORE REGISTERS
            073C 2075
            073C 2076 ;
            073C 2077 ; Fall through into the deallocate logic.
            073C 2078 ;
            073C 2079 ; RSB ; *** This goes in if another piece of
            073C 2080 ; *** initialization code is added that
            073C 2081 ; *** is executed after EXE$INI_TIMWAIT.
            073C 2082      .DISABLE LSB

```

INIADP780  
V03-021

- ADAPTER INITIALIZATION FOR VAX 11/780 3-JUN-1984 14:53:36 VAX-11 Macro V03-01 Page 39  
EXE\$INIT\_TODR - SET SYSTEM TIME TO COR 19-OCT-1983 11:50:19 DISK\$VMMASTER:[SYSLOA.SRC]INIADP(16)

073C 2083

```

073C 2085 DEAL_INIT_CODE: ; DEALLOCATE THE INITIALIZATION CODE
073C 2086 ;
073C 2087 ; It is the duty of the last-executed, loadable initialization
073C 2088 ; routine to make itself and all other such routines disappear, i.e.,
073C 2089 ; release the space they occupy to non-paged pool. Each routine's vector
073C 2090 ; must be disconnected, e.g., be made to point to the symbol, EXE$LOAD_ERROR.
073C 2091 ;
073C 2092 ; NOTE: This means that new initialization routines should be added
073C 2093 ; to this module in a particular order, not necessarily at the
073C 2094 ; end of the module!
073C 2095 ;
073C 2096 .ENABLE LSB
7E 52 7D 073C 2097 MOVQ R2,-(SP) ; Save some registers
073F 2098
073F 2099 ;
073F 2100 ; First find the vectors that point to these initialization routines
073F 2101 ; and reset them to point to EXE$LOAD_ERROR.
073F 2102 ;
50 0000'CF 9E 073F 2103 MOVAB W^SYSL$BEGIN,R0 ; Compute bounds of releasable piece:
00000000'8F C1 0744 2104 ADDL3 #<STAY_HEADER-SYSL$BEGIN>,R0,R1 ; starting and ending addresses.
51 50 074A
00000000'GF 9E 074C 2105 MOVAB G^EXE$AL_LOAVEC,R2 ; Get starting address of vectors.
52 0752
00000000'GF 9E 0753 2106 MOVAB G^EXE$LOAD_ERROR,R3 ; Get end of vectors.
53 0759
9F17 8F 62 B1 075A 2107 10$: CMPW (R2),#^X9F17 ; Is this JMP @# ?
1B 13 075F 2108 BEQL 30$ ; Br if yes, skip past it.
80 8F 03 A2 91 0761 2109 CMPB 3(R2),#^X80 ; Is this a system space address
16 12 0766 2110 BNEQ 40$ ; Br if no, assume it's a HALT instr.
50 62 D1 0768 2111 CMPL (R2),R0 ; Is address before the releasable
0C 1F 076B 2112 BLSSU 20$ ; piece of memory? Br on yes.
51 62 D1 076D 2113 CMPL (R2),R1 ; Is address after the releasable
07 1A 0770 2114 BGTRU 20$ ; piece of memory? Br on yes.
00000000'GF 9E 0772 2115 MOVAB G^EXE$LOAD_ERROR,(R2) ; Reset this vector.
62 0778
52 02 C0 0779 2116 20$: ADDL #2,R2 ; Point past this vector.
52 D6 077C 2117 30$: INCL R2 ; Come here to point past JMP @#.
52 D6 077E 2118 40$: INCL R2 ; Come here to point past HALT.
53 52 D1 0780 2119 CMPL R2,R3 ; Past the end of the vectors?
D5 1F 0783 2120 BLSSU 10$ ; Keep searching vectors.
0785 2121 ;
0785 2122 ; Now release the memory to non-paged pool.
0785 2123 ;
50 0000'CF 9E 0785 2124 MOVAB W^SYSL$BEGIN,R0 ; Point to start of module
51 0000'8F 3C 078A 2125 MOVZWL #<STAY_HEADER-SYSL$BEGIN>,R1 ; Length to vaporize
F87A 31 078F 2126 BRW 50$ ; Br to code that is not released.
0792 2127
00000000 2128 PSECT $$$INIT__END,PAGE ; 'PAGE' SINCE 16-BYTE ALIGN IS NOT
0000 2129
0000 2130 STAY_HEADER:
00000000 00000000 0000 2131 .LONG 0,0
0000 0008 2132 .WORD <SYSL$END-STAY_HEADER>
62 000A 2133 .BYTE DYN$C_LOADCODE
00 000B 2134 .BYTE 0
000C 2135
00000000'9F 16 000C 2136 50$: JSB @#EXE$DEANONPGDSIZ ; Just the smile on the Chesire cat
52 8E 7D 0012 2137 MOVQ (SP)+,R2 ; Restore

```



INIADP780  
V03-021

- ADAPTER INITIALIZATION FOR VAX 11/780 3-JUN-1984 14:53:36 VAX-11 Macro V03-01 Page 41  
EXE\$INIT\_TODR - SET SYSTEM TIME TO COR 19-OCT-1983 11:50:19 DISK\$VMSMASTER:[SYSLOA.SRC]INIADP(17)

```
05 0015 2138      RSB                      ; Return.
    0016 2139
    0016 2140      .DISABLE LSB
    0016 2141      .END
```

\$\$\$VMSDEFINED	=	00000001		CONFREGL		000001E4	R	08
\$\$T1	=	00000001		CPU_ADPSIZE		00000019	R	08
ADAPTERS		00000000	R 02	CPU_TYPE	=	00000001		
ADP\$B_TYPE	=	0000000A		CR	=	0000000D		
ADP\$C_CIAADPLEN	=	00000030		CRB\$B_TYPE	=	0000000A		
ADP\$C_DRADPLEN	=	00000030		CRB\$C_LENGTH	=	00000048		
ADP\$C_MBAADPLEN	=	00000030		CRB\$L_INTD	=	00000024		
ADP\$C_UBAADPLEN	=	00000258		CRB\$L_WQBL	=	00000004		
ADP\$L_AVECTOR	=	0000001C		CRB\$L_WQFL	=	00000000		
ADP\$L_CRB	=	00000010		CRB\$W_SIZE	=	00000008		
ADP\$L_CSR	=	00000000		CREATE_ARRAYS	=	0000011A	R	09
ADP\$L_DPQFL	=	00000014		CSR_LEN_OFFSET	=	FFFFFFFFB		
ADP\$L_LINK	=	00000004		DDB\$T_NAME	=	00000014		
ADP\$L_MBASCB	=	00000014		DEAL_INIT_CODE	=	0000073C	R	09
ADP\$L_MBASPT	=	00000018		DIRECT_VEC_NODE_CNT	=	00000009	R	08
ADP\$L_MRACTMDRS	=	0000005C		DR\$INITIAL	=	*****	X	09
ADP\$L_MRQFL	=	00000030		DR\$INT	=	*****	X	09
ADP\$L_UBASCB	=	00000044		DRTAB	=	00000011	R	08
ADP\$L_UBASPT	=	00000054		DYN\$C_ADP	=	00000001		
ADP\$L_VECTOR	=	00000010		DYN\$C_CONF	=	00000007		
ADP\$W_ADPTYPE	=	0000000E		DYN\$C_CRB	=	00000005		
ADP\$W_DPBITMAP	=	00000060		DYN\$C_IDB	=	00000009		
ADP\$W_MRFFENCE	=	0000015C		DYN\$C_INIT	=	00000063		
ADP\$W_MRFREGARY	=	0000015E		DYN\$C_LOADCODE	=	00000062		
ADP\$W_MRNFFENCE	=	00000062		END_NEXUS	=	00000115	R	09
ADP\$W_MRNREGARY	=	00000064		ERROR_HALT	=	00000199	R	09
ADP\$W_SIZE	=	00000008		ERROR_HALT_1	=	0000019E	R	09
ADP\$W_TR	=	0000000C		EXE\$AL_LOAVEC	=	*****	X	09
ADP\$W_UMR_DIS	=	00000256		EXE\$DEANONPGDSIZ	=	*****	X	0A
ADPLINK		*****	X 09	EXE\$GL_CONFREG	=	*****	X	09
ADPTAB_ADPLEN	=	00000001		EXE\$GL_CONFREGL	=	*****	X	09
ADPTAB_ATYPE	=	00000003		EXE\$GL_FLAGS	=	*****	X	09
ADPTAB_IDBUNITS	=	00000000		EXE\$GL_NUMNEXUS	=	*****	X	09
ALONPAGD	=	000003D4	R 09	EXE\$GL_RPB	=	*****	X	09
AT\$CI	=	00000004		EXE\$GL_SCB	=	*****	X	09
AT\$DR	=	00000002		EXE\$GL_TENUSEC	=	*****	X	09
AT\$MBA	=	00000000		EXE\$GL_TODR	=	*****	X	09
AT\$UBA	=	00000001		EXE\$GL_UBDELAY	=	*****	X	09
BADUMR	=	000003D0	R 08	EXE\$GQ_BOOTTIME	=	*****	X	09
BI_BUS_CODE	=	80000000		EXE\$GQ_TODCBASE	=	*****	X	09
BI_CPU	=	00000000		EXE\$INIT_TODR	=	00000621	RG	09
BI_CSR_LEN	=	00000002		EXE\$INI_TIMWAIT	=	0000056F	RG	09
BI_LIKE	=	00000000		EXE\$LOAD_ERROR	=	*****	X	09
BLD_CRB	=	00000506	R 09	EXE\$MCHK_PRTCT	=	*****	X	09
BOO\$GB_SYSTEMID	=	*****	X 09	EXE\$OUTZSTRING	=	*****	X	09
BOO\$GL_SPTFREH	=	*****	X 09	EXE\$SETIME_INT	=	*****	X	09
BOO\$GL_SPTFREL	=	*****	X 09	EXE\$TEST_CSR	=	*****	X	09
BOOTVECTOR	=	00000000	R 08	EXE\$UBAERR_INT	=	*****	X	09
BTD\$K_CONSOLE	=	00000040		EXE\$V_SETTIME	=	*****	X	09
BTD\$K_UDA	=	00000011		FILL_CRB	=	00000513	R	09
BUS_CODE_OFFSET	=	FFFFFFFFC		GET_GEN_TYPE	=	000000B9	R	09
BUS_CSR_LEN	=	00000004	R 08	GET_TYPE	=	000000A0	R	09
CI\$INITIAL	=	*****	X 09	IDB\$B_TYPE	=	0000000A		
CI\$INT	=	*****	X 09	IDB\$C_LENGTH	=	00000038		
CITAB	=	00000015	R 08	IDB\$L_ADP	=	00000014		
CONFIG_IOSPACE	=	0000005F	R 09	IDB\$L_CSR	=	00000000		
CONFREG	=	000000A4	R 08	IDB\$W_SIZE	=	00000008		

IDB\$W_UNITS	=	0000000C		NDT\$_UB0	=	00000028	
INI\$ALLOC_CRB		*****	X 09	NDT\$_UB1	=	00000029	
INI\$ALONONPAGED		*****	X 09	NDT\$_UB2	=	0000002A	
INI\$CIADP		000003EF	R 09	NDT\$_UB3	=	0000002B	
INI\$CONSOLE		000004ED	RG 09	NEXUSDESC		00000020	R 08
INI\$DRADP		000003DA	R 09	NOSPT		000002E4	R 08
INI\$IOMAP		00000000	RG 09	NPROMPT	=	00000033	
INI\$KDZ11		000004EC	R 09	NUMUBAVEC	=	00000080	
INI\$MBADP		00000404	R 09	NUM_PAGES		00000000	R 04
INI\$MPMADP		*****	X 06	NXT_NEXUS		0000006B	R 09
INI\$UBADP		000001C3	R 09	OPA\$CRB		*****	X 09
INI\$UBSPACE		000001A7	R 09	PA	=	20020000	
INIT_ROUTINES		00000000	R 06	PR\$_ICCS	=	00000018	
INTIME	=	00000014		PR\$_SID_TYP730	=	00000003	
IO\$M_CVTLOW		*****	X 09	PR\$_SID_TYP750	=	00000002	
IO\$M_PURGE		*****	X 09	PR\$_SID_TYP780	=	00000001	
IO\$M_TIMED		*****	X 09	PR\$_SID_TYP790	=	00000004	
IO\$_READPROMPT		*****	X 09	PR\$_SID_TYP8NN	=	00000006	
IO\$_WRITEVBLK		*****	X 09	PR\$_SID_TYP8SS	=	00000005	
IO780\$AL_IOBASE	=	20000000		PR\$_SID_TYPUV1	=	00000007	
IO780\$AL_NNEX	=	00000010		PR\$_TBIS	=	0000003A	
IO780\$AL_PERNEX	=	00002000		PR780\$_ICR	=	0000001A	
IO780\$AL_UBOSP	=	20100000		PR780\$_NICR	=	00000019	
LF	=	0000000A		PR780\$_TODR	=	0000001B	
LINBUF	=	0000001C		PTE\$C_KW	=	10000000	
LINBUFSIZ	=	00000014		PTE\$M_VALID	=	80000000	
MAP_NEXUS		000000F9	R 09	READTIME		00000659	R 09
MAP_PAGES		00000173	R 09	RPB\$B_DEVTYP	=	00000066	
MAXNEXUS	=	00000040		RPB\$L_ADPPHY	=	0000005C	
MBA\$INITIAL		*****	X 09	RPB\$L_ADPVIR	=	00000060	
MBA\$INT		*****	X 09	RPB\$L_BOOTR1	=	00000020	
MBATAB		0000000D	R 08	RPB\$L_CSRPHY	=	00000054	
MCHK\$M_LOG	=	00000001		RPB\$L_CSRVIR	=	00000058	
MCHK\$M_NEXM	=	00000004		RPB\$W_ROUBVEC	=	0000001E	
MMG\$GL_SBICONF		*****	X 09	SBICONF		000000E4	R 08
MMG\$GL_SPTBASE		*****	X 09	SBI_BUS_CODE	=	00000000	
MMG\$SVAPTECHK		*****	X 09	SBI_CPU	=	00000001	
NDT\$_BUA	=	80000102		SBI_CSR_LEN	=	00000001	
NDT\$_CI	=	00000038		SBI_LIKE	=	00000001	
NDT\$_DR32	=	00000030		SGN\$GW_TPWAIT		*****	X 09
NDT\$_KDZ11	=	80000105		STAY_HEADER		00000000	R 0A
NDT\$_MB	=	00000020		SW_BUS_CODE		00000005	R 08
NDT\$_MEM1664NI	=	00000012		SYS\$ASSIGN		*****	GX 09
NDT\$_MEM16I	=	00000011		SYS\$BINTIM		*****	GX 09
NDT\$_MEM16NI	=	00000010		SYS\$DASSGN		*****	GX 09
NDT\$_MEM4I	=	00000009		SYS\$QIOW		*****	GX 09
NDT\$_MEM4NI	=	00000008		SYSL\$BEGIN		*****	X 09
NDT\$_MEM64EIL	=	00000069		SYSL\$END		*****	X 0A
NDT\$_MEM64EIU	=	0000006B		TERM_NAMADR		000005D7	R 09
NDT\$_MEM64I	=	0000006C		TERM_NAMSIZ	=	00000004	
NDT\$_MEM64NIL	=	00000068		TEST_NEXUS		00000071	R 09
NDT\$_MEM64NIU	=	0000006A		TIMEPROMPT		000005ED	R 09
NDT\$_MPM0	=	00000040		TIMERR		000005DB	R 09
NDT\$_MPM1	=	00000041		TMPDESC	=	0000000C	
NDT\$_MPM2	=	00000042		TTCHAN	=	00000000	
NDT\$_MPM3	=	00000043		TTNAME	=	00000004	
NDT\$_SCORMEM	=	80000001		UBA\$INITIAL		*****	X 09

```

UBA$INT0          ***** X 09
UBA$L_BRRVR      = 00000030
UBA$L_CR         = 00000004
UBA$UNEXINT     ***** X 09
UBAERRADR       = 00000090
UBAINT4         = 00000000
UBAINT4REL      = 0000000A
UBAINT5         = 00000020
UBAINT5REL     = 0000002A
UBAINT6         = 00000040
UBAINT6REL     = 0000004A
UBAINT7         = 00000060
UBAINT7REL     = 0000006A
UBAINTADP       = 00000089
UBAINTBASE     = 00000340 R 09
UBINTSZ        = 00000094
VASM_SYSTEM    = 80000000
VEC$L_ADP      = 00000014
VEC$L_IDB      = 00000008
VEC$L_INITIAL  = 0000000C
VECTAB         = 000003D4 R 09
    
```

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

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	00000004 ( 4.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$INIT\$DATA0	00000060 ( 96.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$INIT\$DATA1	00000000 ( 0.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$INIT\$DATA2	00000030 ( 48.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$INIT\$DATA3	00000000 ( 0.)	05 ( 5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$INIT\$DATA4	00000060 ( 96.)	06 ( 6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$INIT\$DATA5	00000000 ( 0.)	07 ( 7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$INIT\$DATA	0000033F ( 831.)	08 ( 8.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$\$\$INIT\$CODE	00000792 ( 1938.)	09 ( 9.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD
\$\$\$INIT__END	00000016 ( 22.)	0A ( 10.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.07	00:00:00.34
Command processing	124	00:00:00.50	00:00:01.88
Pass 1	530	00:00:16.59	00:01:02.39
Symbol table sort	0	00:00:01.67	00:00:04.91
Pass 2	313	00:00:06.56	00:00:33.79
Symbol table output	32	00:00:00.16	00:00:00.66
Psect synopsis output	7	00:00:00.06	00:00:00.21
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1040	00:00:25.63	00:01:44.18

The working set limit was 1950 pages.

134197 bytes (263 pages) of virtual memory were used to buffer the intermediate code.  
There were 90 pages of symbol table space allocated to hold 1618 non-local and 36 local symbols.  
2489 source lines were read in Pass 1, producing 39 object records in Pass 2.  
46 pages of virtual memory were used to define 44 macros.

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

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	21
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	15
TOTALS (all libraries)	36

1755 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:INIADP780/OBJ=OBJ\$:INIADP780 MSRC\$:CPUSW780/UPDATE=(ENH\$:CPUSW780)+MSRC\$:INIADP/UPDATE=(ENH\$:INIADP)+EXECML\$/LIB



**LIOSUB780**

(3) 137

PURGE DATAPATH



```
0000 1 .NOSHOW CONDITIONALS
0000 3 .TITLE LIOSUB780 - LOADABLE I/O SUBROUTINES
0000 5
0000 9
0000 13
0000 17
0000 21
0000 22 .IDENT 'V03-012'
0000 23
0000 24 ;
0000 25 ;*****
0000 26 ;*
0000 27 ;* COPYRIGHT (c) 1978, 1980, 1982, 1983, 1984 BY *
0000 28 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
0000 29 ;* ALL RIGHTS RESERVED. *
0000 30 ;*
0000 31 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
0000 32 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
0000 33 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
0000 34 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
0000 35 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
0000 36 ;* TRANSFERRED. *
0000 37 ;*
0000 38 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
0000 39 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
0000 40 ;* CORPORATION. *
0000 41 ;*
0000 42 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
0000 43 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
0000 44 ;*
0000 45 ;*
0000 46 ;*****
0000 47 ;
0000 48 ;++
0000 49 ;
0000 50 ; FACILITY:
0000 51 ;
0000 52 ; EXECUTIVE, I/O CONTROL ROUTINES
0000 53 ;
0000 54 ; ABSTRACT:
0000 55 ;
0000 56 ; I/O SUBROUTINES WHICH CONTAIN PROCESSOR DEPENDENCIES.
0000 57 ;
0000 58 ; AUTHOR:
0000 59 ;
0000 60 ; N. KRONENBERG, JANUARY 12, 1979.
0000 61 ;
0000 62 ; MODIFIED BY:
0000 63 ;
0000 64 ; V03-012 KDM0096 Kathleen D. Morse 27-Mar-1984
0000 65 ; Add memory CSR scanning to IOC$PURGDATAP for MicroVAX I.
0000 66 ; (All DMA MicroVAX I drivers should call this routine, just
0000 67 ; before calling IOC$REQCOM.)
0000 68 ;
0000 69 ; V03-011 KDM0081 Kathleen D. Morse 13-Sep-1983
0000 70 ; Create a version for Micro-VAX I.
0000 71 ;
```

0000	72 ;	V03-010	TCM0004	Trudy C. Matthews	4-Jan-1982
0000	73 ;			Added 11/790-specific path to IOC\$PURGDATAP.	
0000	74 ;				
0000	75 ;	V09	TCM0003	Trudy C. Matthews	9-Nov-1982
0000	76 ;			Added a .TITLE statement for LIOSUB790.	
0000	77 ;				
0000	78 ;	V08	TCM0002	Trudy C. Mathews	29-Jul-1981
0000	79 ;			Changed all "7ZZ"s to "730"s.	
0000	80 ;				
0000	81 ;	V07	TCM0001	Trudy C Matthews	28-Feb-1980
0000	82 ;			Changed IOC\$PURGDATAP for NEBULA so that it logs	
0000	83 ;			the Unibus Error Summary register itself when there	
0000	84 ;			are Unibus errors reported.	
0000	85 ;				
0000	86 ;	V06	NPK0002	N. KRONENBERG	4-DEC-1979
0000	87 ;			REPLACED IOC\$PURGDATAP FOR NEBULA	
0000	88 ;				
0000	89 ;	V05	NPK0001	N. KRONENBERG	23-AUG-1979
0000	90 ;			CORRECTED 11/750 CHECK FOR PURGE DONE.	
0000	91 ;				
0000	92 ;	V04	TCM0001	Trudy C. Matthews	3-Jul-1979
0000	93 ;			Modified IOC\$PURGDATAP for NEBULA.	
0000	94 ;				
0000	95 ;--				

```
0000 97 ;  
0000 98 ; MACRO LIBRARY CALLS:  
0000 99 ;  
0000 100 $ADPDEF ; Define ADP offsets  
0000 101 $CRBDEF ; Define CRB offsets  
0000 102 $EMBETDEF ; Define error types.  
0000 103 $EMBUDEF ; Define Unibus Error buffer.  
0000 104 $IDBDEF ; Define IDB offsets  
0000 105 $PRDEF ; Define IPR'S  
0000 106 $UBADEF ; Define UBA offsets  
0000 107 $UBIDEF ; Define UBI offsets  
0000 108 $UCBDEF ; Define UCB offsets  
0000 109 $VECDEF ; Define CRB/VEC offsets  
0000 110  
00000001 0000 112 C780_LIKE = 1  
00000000 0000 113 C750_LIKE = 0  
0000 115  
0000 120  
0000 125  
0000 130  
0000 135
```

```

0000 137          .SBTTL  PURGE DATAPATH
0000 138 ;+
0000 139 ; IOC$PURGDATAP - PURGE DATAPATH
0000 140 ;
0000 141 ; This routine purges the caller's buffered datapath, and clears any
0000 142 ; datapath errors.  if there was a datapath error, this fact is
0000 143 ; returned to the caller.
0000 144 ;
0000 145 ; INPUTS:
0000 146 ;
0000 147 ;          R5 = UCB address
0000 148 ;
0000 149 ; OUTPUTS:
0000 150 ;
0000 151 ;          R0-R3 altered
0000 152 ;          Other registers preserved
0000 153 ;          R0 = low bit clear/set if transmission error/success
0000 154 ;          R1 = DPR contents after purge (for register dump by caller)
0000 155 ;          R2 = address of start of adapter map registers (for reg dump by caller)
0000 156 ;          R3 = CRB address
0000 157 ;-
0000 158
00000000 159          .PSECT  WIONONPAGED
0000 160
0000 161          .ENABL  LSB
0000 162
0000 163 IOC$PURGDATAP::
53      24 A5  BB 0000 165          PUSHR  #AM<R4>          ; Save register
52      38 B3  D0 0002 166          MOVL   UCB$L_CRB(R5),R3      ; Get CRB address
000A    168          MOVL   @CRB$L_INTD+VEC$L_ADP(R3),R2 ; Get start of adapter register space
0000 169          EXTZV  #VEC$V_DATAPATH,-          ; Extract datapath #
0005    05          MOVL   #VEC$S_DATAPATH,-          ; from CRB
51      37 A3  D0 000D 171          MOVL   CRB$L_INTD+VEC$B_DATAPATH(R3),R1
54      40 A241 DE 0010 172          MOVAL  UBA$L_DPR(R2)[R1],R4      ; Get address of DPR
64      01 1F 78 0015 173          ASHL  #UBA$V_DPR_BNE,#1,(R4) ; Purge datapath
0051    64 D0 0019 174          MOVL  (R4),R1          ; Get DPR contents
08      51 1E E1 001C 175          BBC   #UBA$V_DPR_XMTER,R1,20$ ; Branch if no error
64      01 1E 78 0020 176          ASHL  #UBA$V_DPR_XMTER,#1,(R4) ; Clear error in DPR
0050    50 D4 0024 177          CLRL  R0          ; Set to return transfer error
0003    03 11 0026 178          BRB   30$          ; Join common code
0050    01 9A 0028 179 20$:      MOVZBL #1,R0          ; Set to return transfer success
52      0800 C2 DE 002B 180 30$:  MOVAL  UBA$L_MAP(R2),R2      ; Return addr of 1st map register
0030    181
0010    10 BA 0030 182          POPR  #AM<R4>          ; Restore register
0005    05 0032 183          RSB          ; Return
0033    185
0033    186
0033    214
0033    263
0033    297
0033    298          .DSABL  LSB
0033    299
0033    300          .END

```

LIOSUB780  
Symbol table

- LOADABLE I/O SUBROUTINES

3-JUN-1984 14:57:17 VAX-11 Macro V03-01 Page 5  
6-APR-1984 16:16:35 DISK\$VMSMASTER:[SYSLOA.SRC]LIOSUB.(3)

```
C750_LIKE          = 00000000
C780_LIKE          = 00000001
CPU_TYPE          = 00000001
CRB$L_INTD        = 00000024
IOC$PURGDATAP     00000000 RG    02
PR$_SID_TYP730    = 00000003
PR$_SID_TYP750    = 00000002
PR$_SID_TYP780    = 00000001
PR$_SID_TYP790    = 00000004
PR$_SID_TYPUV1    = 00000007
UBA$L_DPR         = 00000040
UBA$L_MAP         = 00000800
UBA$V_DPR_BNE     = 0000001F
UBA$V_DPR_XMTER   = 0000001E
UCB$L_CRB         = 00000024
VEC$B_DATAPATH    = 00000013
VEC$L_ADP         = 00000014
VEC$$S_DATAPATH   = 00000005
VEC$V_DATAPATH    = 00000000
```

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

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WIONONPAGED	00000033 ( 51.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.04	00:00:00.24
Command processing	121	00:00:00.49	00:00:02.65
Pass 1	273	00:00:04.90	00:00:10.47
Symbol table sort	0	00:00:00.77	00:00:01.25
Pass 2	31	00:00:00.97	00:00:02.58
Symbol table output	3	00:00:00.03	00:00:00.03
Psect synopsis output	3	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	459	00:00:07.21	00:00:17.23

The working set limit was 1350 pages.  
43453 bytes (85 pages) of virtual memory were used to buffer the intermediate code.  
There were 40 pages of symbol table space allocated to hold 754 non-local and 2 local symbols.  
304 source lines were read in Pass 1, producing 13 object records in Pass 2.  
17 pages of virtual memory were used to define 16 macros.

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

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	9
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	13

845 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:LIOSUB780/OBJ=OBJ\$:LIOSUB780 MSRC\$:CPUSW780/UPDATE=(ENH\$:CPUSW780)+MSRC\$:LIOSUB/UPDATE=(ENH\$:LIOSUB)+EXECML\$/LIB

**LOADMREG**

(1)	74	LOAD MASSBUS ADAPTER MAP REGISTERS
(1)	120	LOAD UNIBUS ADAPTER MAP REGISTERS
(1)	197	GET PFN FROM INVALID PTE
(1)	230	Load UBA map registers
(1)	276	LOAD UNIBUS ADAPTER MAP REGISTERS FOR UDA PORT



```
LOADMREG - LOAD MBA AND UBA MAP REGISTERS
;RLRPDTADP -1
0000 1 .TITLE LOADMREG - LOAD MBA AND UBA MAP REGISTERS
0000 1 .IDENT 'V03-002'
0000 3
0000 4 ;
0000 5 ;*****
0000 6 ;*
0000 7 ;* COPYRIGHT (c) 1978, 1980, 1982 BY
0000 8 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 ;* ALL RIGHTS RESERVED.
0000 10 ;*
0000 11 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 ;* TRANSFERRED.
0000 17 ;*
0000 18 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 ;* CORPORATION.
0000 21 ;*
0000 22 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 ;*
0000 25 ;*
0000 26 ;*****
0000 27 ;
0000 28 ; D. N. CUTLER 1-NOV-77
0000 29 ;
0000 30 ; LOAD MBA MAP REGISTERS
0000 31 ;
0000 32 ; MODIFIED BY:
0000 33 ;
;RLRPDTADP 0000 .1 ; V03-002 RLRPDTADP Robert L. Rappaport 9-Apr-1984
;RLRPDTADP 0000 .2 ; Modify IOC$LUBAUDAMAP so as to pickup the pointer to the
;RLRPDTADP 0000 .3 ; ADP from PDT$L_ADP(R4).
;RLRPDTADP 0000 .4 ;
;KDM0002 0000 .5 ; V03-001 KDM0002 Kathleen D. Morse 28-Jun-1982
;KDM0002 0000 .6 ; Add $VADEF.
-24 0000 58 ;
```

	0000	60 ;		
	0000	61 ;	MACRO LIBRARY CALLS	
	0000	62 ;		
	0000	63		
	0000	64	\$ADPDEF	;DEFINE ADP OFFSETS
	0000	65	\$CDRPDEF	;DEFINE CDRP OFFSETS
	0000	66	\$CRBDEF	;DEFINE CRB OFFSETS
	0000	67	\$MBADEF	;DEFINE MBA REGISTER OFFSET DEFINITIONS
;RLRPDTADP	0000	.1	\$PDTDEF	;DEFINE PDT OFFSETS
;RLRPDTADP	0000	.2	\$PTEDEF	;DEFINE PAGE TABLE ENTRY FIELDS
;RLRPDTADP	0000	.3	\$UBADEF	;DEFINE UCB OFFSETS
;RLRPDTADP	0000	.4	\$UBMDEF	;DEFINE UBMD OFFSETS
;RLRPDTADP	0000	.5	\$UCBDEF	;DEFINE UCB OFFSETS
;KDM0002	0000	.6	\$VADEF	;DEFINE VIRTUAL ADDRESS FIELDS
-4	0000	72	\$VECDDEF	;DEFINE CRB TRANSFER VECTOR OFFSETS

```

0000 74          .SBTTL  LOAD MASSBUS ADAPTER MAP REGISTERS
0000 75 ;+
0000 76 ; IOC$LOADMBAMAP - LOAD MASSBUS ADAPTER MAP REGISTERS
0000 77 ;
0000 78 ; THIS ROUTINE IS CALLED TO LOAD THE MASSBUS ADAPTER MAP REGISTERS, THE
0000 79 ; BYTE COUNT REGISTER, AND THE VIRTUAL ADDRESS REGISTER.
0000 80 ;
0000 81 ; INPUTS:
0000 82 ;
0000 83 ;      R4 = ADDRESS OF MBA CONFIGURATION STATUS REGISTER.
0000 84 ;      R5 = UCB ADDRESS OF UNIT TRANSFER IS TO OCCUR ON.
0000 85 ;
0000 86 ; OUTPUTS:
0000 87 ;
0000 88 ;      THE TRANSFER BYTE COUNT, STARTING PAGE OFFSET, AND ADDRESS OF THE
0000 89 ;      PAGE TABLE ENTIRES THAT DESCRIBE THE TRANSFER ARE RETRIEVED FROM
0000 90 ;      THE SPECIFED UCB AND USED TO LOAD THE MBA BYTE COUNT, VIRTUAL ADDRESS,
0000 91 ;      AND MAP REGISTERS. ONE ADDITIONAL MAP REGISTER IS LOADED AS INVALID
0000 92 ;      TO STOP THE TRANSFER IF A HARDWARE FAILURE SHOULD OCCUR.
0000 93 ;
0000 94 ;      R3 IS PRESERVED ACROSS CALL.
0000 95 ;-
0000 96
00000000 97          .PSECT  WIONONPAGED
0000 98 IOC$LOADMBAMAP:
52 7E A5 53 DD 0000 99          PUSHL  R3          ;LOAD MASSBUS ADAPTER MAP REGISTERS
10 A4 52 CE 0006 100         MOVZWL  UCB$W_BCNT(R5),R2      ;SAVE REGISTERS
51 7C A5 3C 000A 101         MNEGL  R2,MBA$B_BCR(R4)      ;GET TRANSFER BYTE COUNT
0C A4 51 D0 000E 102         MOVZWL  UCB$W_BOFF(R5),R1      ;LOAD BYTE COUNT REGISTER
0C A4 51 D0 0012 103         MOVL   R1,MBA$L_VAR(R4)      ;GET BYTE OFFSET IN PAGE
52 01FF C241 9E 0016 104        MOVL   R1,MBA$L_VAR(R4)      ;LOAD STARTING VIRTUAL ADDRESS
52 F7 8F 78 001C 105        MOVAB  ^X1FF(R2)[R1],R2      ;*****TEMP UNTIL MBA ECO *****
52 52 0020 106        ASHL   #-9,R2,R2          ;CALCULATE HIGHEST RELATIVE BYTE AND ROUND
51 0800 C4 DE 0021 107        MOVAL  MBA$L_MAP(R4),R1      ;CALCULATE NUMBER OF MAP REGISTERS TO LOAD
50 78 A5 D0 0026 108        MOVL   UCB$L_SVAPTE(R5),R0    ;GET ADDRESS OF MBA MAP REGISTERS
81 80 D0 002A 109 10$:      MOVL   (R0)+,(R1)+          ;GET ADDRESS OF PAGE TABLE
09 18 002D 110        BGEQ  30$                    ;LOAD MAP REGISTER
F8 52 F5 002F 111 20$:      SOBGTR  R2,10$                    ;IF GEQ PTE INVALID
61 D4 0032 112        CLRL  (R1)                    ;ANY MORE TO LOAD?
53 8E D0 0034 113        MOVL   (SP)+,R3              ;LOAD INVALID MAP ENTRY
05 0037 114        RSB                                     ;RESTORE REGISTER
53 FC A0 D0 0038 115 30$:      MOVL   -4(R0),R3              ;
0095 30 003C 116        BSBW  IOC$PTETOPFN          ;GET THE PTE (NOT FROM MAP REGISTER!)
80000000 8F C9 003F 117        BISL3  ^#X80000000,R3,-4(R1) ;GET PFN FROM INVALID PTE
FC A1 53 0045 118        BRB    20$                    ;AND LOAD THE MAP REGISTER
E5 11 0048

```

```

004A 120      .SBTTL  LOAD UNIBUS ADAPTER MAP REGISTERS
004A 121 ;+
004A 122 ; IOC$LOADUBAMAP - LOAD UNIBUS ADAPTER MAP REGISTERS
004A 123 ; IOC$LOADUBAMAPA - LOAD UNIBUS ADAPTER MAP REGISTERS ALTERNATE ENTRY FOR
004A 124 ;      BYTE ALIGNED UNIBUS DMA DEVICES WHICH NEVER WISH TO SET THE BYTE
004A 125 ;      OFFSET BIT IN MAP REGISTERS.  IN ALL OTHER RESPECTS THESE TWO
004A 126 ;      ENTRYPOINTS PRODUCE IDENTICAL RESULTS.
004A 127 ;
004A 128 ; THIS ROUTINE IS CALLED TO LOAD THE UNIBUS ADAPTER MAP REGISTERS.
004A 129 ;
004A 130 ; INPUTS:
004A 131 ;
004A 132 ;      R5 = UCB ADDRESS OF UNIT TRANSFER IS TO OCCUR ON.
004A 133 ;
004A 134 ;      IT IS ASSUMED THAT THE DATAPATH AND MAP REGISTERS HAVE BEEN PREVIOUSLY
004A 135 ;      ASSIGNED.
004A 136 ;
004A 137 ; OUTPUTS:
004A 138 ;
004A 139 ;      EACH MAP REGISTER IS LOADED WITH THE APPROPRIATE PAGE FRAME NUMBER
004A 140 ;      MERGED WITH THE DATAPATH DESIGNATOR AND BYTE OFFSET BIT.  ONE ADDITIONAL
004A 141 ;      MAP REGISTER IS LOADED AS INVALID TO STOP THE TRANSFER IF A HARDWARE
004A 142 ;      FAILURE SHOULD OCCUR.
004A 143 ;
004A 144 ;      R3 IS PRESERVED ACROSS CALL.
004A 145 ; -
004A 146
004A 147      .ENABL  LSB
004A 148 IOC$LOADUBAMAPA::      ;LOAD UNIBUS ADAPTER MAP REGISTERS - ALTERNATE
004A 149      ; HERE WE DUPLICATE THE CODE IN THE OTHER ENTRY
004A 150      ; EXCEPT THAT WE DO NOT CHECK WHETHER THE BYTE
004A 151      ; OFFSET IS ODD.  INSTEAD WE BRANCH DIRECTLY
004A 152      ; PAST THE SETTING OF THE BYTE OFFSET BIT.
      7E 53 7D 004A 153      MOVQ      R3, -(SP)      ; SAVE REGISTERS
51 7C A5 3C 004D 154      MOVZWL   UCB$W_BOFF(R5),R1      ; GET BYTE OFFSET IN PAGE
52 7E A5 3C 0051 155      MOVZWL   UCB$W_BCNT(R5),R2      ; GET TRANSFER BYTE COUNT
53 24 A5 D0 0055 156      MOVL     UCB$L_CRB(R5),R3      ; GET ADDRESS OF CRB
      00  EF 0059 157      EXTZV   #VEC$V_DATAPATH,-      ; GET DATAPATH
      05 005B 158      #VEC$S_DATAPATH,-      ; NUMBER
54 37 A3 005C 159      CRB$L_INTD+VEC$B_DATAPATH(R3),R4
      1B 11 005F 160      BRB      10$      ; BRANCH AROUND TO JOIN COMMON CODE
      0061 161 IOC$LOADUBAMAP::      ;LOAD UNIBUS ADAPTER MAP REGISTERS
      7E 53 7D 0061 162      MOVQ      R3, -(SP)      ; SAVE REGISTERS
51 7C A5 3C 0064 163      MOVZWL   UCB$W_BOFF(R5),R1      ; GET BYTE OFFSET IN PAGE
52 7E A5 3C 0068 164      MOVZWL   UCB$W_BCNT(R5),R2      ; GET TRANSFER BYTE COUNT
53 24 A5 D0 006C 165      MOVL     UCB$L_CRB(R5),R3      ; GET ADDRESS OF CRB
      00  EF 0070 166      EXTZV   #VEC$V_DATAPATH,-      ; GET DATAPATH
      05 0072 167      #VEC$S_DATAPATH,-      ; NUMBER
54 37 A3 0073 168      CRB$L_INTD+VEC$B_DATAPATH(R3),R4
      03 51 E9 0076 169      BLBC     R1,10$      ; IF LBC WORD ALIGNED TRANSFER
      54 10 88 0079 170      BISB     #^X10,R4      ; SET BYTE OFFSET BIT
54 0400 8F A8 007C 171 10$:  BISW     #^X400,R4      ; MERGE VALID WITH BYTE OFFSET AND DATAPATH
      05  E1 0081 172      BBC      #VEC$V_LWAE,-      ; BRANCH IF LONGWORD ACCESS NOT ENABLED
      03 37 A3 0083 173      CRB$L_INTD+VEC$B_DATAPATH(R3),15$
      54 20 88 0086 174      BISB     #^X20,R4      ; ELSE SET LWAE FOR MAP REG
52 01FF C241 9E 0089 175 15$:  MOVAB    ^X1FF(R2)|R1|,R2      ; CALCULATE HIGHEST RELATIVE BYTE AND ROUND
      52 F7 8F 78 008F 176      ASHL     #-9,R2,R2      ; CALCULATE NUMBER OF MAP REGISTERS TO LOAD

```

```

52          0093
36 A3 52 91 0094 177      CMPB   R2,CRB$L_INTD+VEC$B_NUMREG(R3) ;ENOUGH MAP REGISTERS ASSIGNED?
          2C 1E 0098 178      BGEQU  40$ ;IF GEQU NO
51  38 B3 D0 009A 179      MOVL   @CRB$L_INTD+VEC$L_ADP(R3),R1 ;GET ADDRESS OF CONFIGURATION REGISTER
          00 EF 009E 180      EXTZV  #VEC$V_MAPREG,- ;GET STARTING REGISTER
          0F 00A0 181      #VEC$S_MAPREG,-
50  34 A3 00A1 182      CRB$L_INTD+VEC$W_MAPREG(R3),R0
51 0800 C140 DE 00A4 183      MOVAL  UBA$L_MAP(R1)[R0],R1 ;GET ADDRESS OF FIRST MAP REGISTER TO LOAD
50  78 A5 D0 00AA 184      MOVL   UCB$L_SVAPTE(R5),R0 ;GET ADDRESS OF PAGE TABLE
          53 80 D0 00AE 185 20$: MOVL   (R0)+,R3 ;GET NEXT PAGE TABLE ENTRY
          02 19 00B1 186      BLSS   30$ ;IF LSS VALID PAGE TABLE ENTRY
          1F 10 00B3 187      BSBB   IOC$PTETOPFN ;GET PFN FROM INVALID PTE
OB  15 54 F0 00B5 188 30$: INSV   R4,#21,#11,R3 ;INSERT VALID, BYTE OFFSET, AND DATAPATH
          53 00B9
          81 53 D0 00BA 189      MOVL   R3,(R1)+ ;LOAD UBA MAP REGISTER
          EE 52 F5 00BD 190      SOBGTR R2,20$ ;ANY MORE TO LOAD?
          61 D4 00C0 191      CLRL   (R1) ;LOAD INVALID MAP ENTRY
          53 8E 7D 00C2 192      MOVQ   (SP)+,R3 ;RESTORE REGISTERS
          05 00C5 193      RSB ;
          00C6 194 40$: BUG_CHECK UBAPEXCED,FATAL ;UNIBUS MAP REGISTER ALLOCATION EXCEEDED
          00CA 195      .DSABL LSB

```

```

00CA 197          .SBTTL  GET PFN FROM INVALID PTE
00CA 198 ;+
00CA 199 ; IOC$PTETOPFN - GET PFN FROM INVALID PTE
00CA 200 ;
00CA 201 ; THIS ROUTINE IS CALLED TO RETURN THE PAGE FRAME NUMBER FROM A
00CA 202 ; PAGE TABLE ENTRY WHICH HAS ALREADY BEEN DETERMINED TO BE NOT VALID.
00CA 203 ;
00CA 204 ; INPUTS:
00CA 205 ;
00CA 206 ;          R3 = PAGE TABLE ENTRY
00CA 207 ;
00CA 208 ; OUTPUTS:
00CA 209 ;
00CA 210 ;          R3 = PAGE FRAME NUMBER AND MAY INCLUDE THE FOLLOWING FIELDS
00CA 211 ;                   VALID BIT, MODIFY BIT, PROTECTION FIELD, OWNER FIELD
00CA 212 ;
00CA 213 ;          ALL OTHER REGISTERS PRESERVED
00CA 214 ; -
00CA 215
00CA 216          .ENABL  LSB
00CA 217 GLOBAL:
00000000'FF43 D0 00CA 218          MOVL   @MMG$GL_GPTBASE[R3],R3  ;GLOBAL PAGE TABLE ENTRY
          53 00D1
          0F 19 00D2 219          BLSS   10$                      ;BRANCH IF VALID
          FB800000 8F CA 00D4 220 IOC$PTETOPFN::
          53 00D4 221          BICL   #^C<PTE$M_TYP1 ! PTE$M_TYPO !- ;PTE TYPE BITS
          00DB 222
          05 53 1A E0 00DB 223          BBS   #PTE$V_TYP1,R3,20$      ;AND GPTX/PFN
          E7 53 16 E4 00DF 224          BBSC  #PTE$V_TYPO,R3,GLOBAL  ;BRANCH IF BAD PTE FOR I/O
          05 00E3 225 10$:          RSB
          00E4 226 20$:          BUG_CHECK INVPTEFMT,FATAL      ;INVALID PAGE TABLE ENTRY FORMAT
          00E8 227          .DSABL  LSB
          00E8 228

```

```

00E8 230      .SBTTL  Load UBA map registers
00E8 231 ;++
00E8 232 ; IOC$LOADUBAMAPN - Load UBA map registers
00E8 233 ;
00E8 234 ;   Functional description:
00E8 235 ;
00E8 236 ;   This routine is called to load the UNIBUS adapter map registers.
00E8 237 ;   It differs from IOC$LOADUBAMAPN in that it does not obtain its
00E8 238 ;   variant inputs from the UCB and CRB (which are normally synchronized
00E8 239 ;   at fork IPL). Also, the byte offset and longword aligned
00E8 240 ;   capabilities are not supported.
00E8 241 ;
00E8 242 ; Inputs:
00E8 243 ;
00E8 244 ;   R1 = buffer address
00E8 245 ;   R2 = number of map registers allocated
00E8 246 ;   (last one should be extra one for wild transfer stopper)
00E8 247 ;   R3 = starting map register allocated
00E8 248 ;   R4 = datapath number
00E8 249 ;   R5 = UCB address
00E8 250 ;
00E8 251 ; Outputs:
00E8 252 ;
00E8 253 ;   R0-R4 destroyed. R5 preserved.
00E8 254 ;--
00E8 255 IOC$LOADUBAMAPN: ;
51 15 54 DD 00E8 256 PUSHL R4 ; Save datapath number
00000000'GF 51 09 EF 00EA 257 EXTZV S^#VA$V_VPN,S^#VA$S_VPN,R1,R1 ; Get buffer virtual page number
54 00EE
00000000'GF 54 D0 00EF 258 MOVL G^MMG$GL_SPTBASE,R4 ; Get system page table address
54 00F5
50 6441 DE 00F6 259 MOVAL (R4)[R1],R0 ; Get first PTE address
54 24 A5 D0 00FA 260 MOVL UCB$L_CRB(R5),R4 ; Get CRB address
54 38 B4 D0 00FE 261 MOVL @CRB$L_INTD+VEC$L_ADP(R4),R4 ; Get first UBA register address
51 0800 C443 DE 0102 262 MOVAL UBA$L_MAP(R4)[R3],R1 ; Get address of first map register
00000400 8F C9 0108 263 BISL3 #1@<UBA$V_MAP_VALID-UBA$V_MAP_DPD>,- ; Get datapath number
54 8E 010E 264 (SP)+,R4 ; and set map register valid bit
52 D7 0110 265 DECL R2 ; Subtract last register from count
53 80 D0 0112 266 20$: MOVL (R0)+,R3 ; Get next page table entry
02 19 0115 267 BLSS 30$ ; Br if valid page
BB 10 0117 268 BSBB IOC$PTETOPFN ; Get PFN from invalid (global) page
15 54 F0 0119 269 30$: INSV R4,#UBA$V_MAP_DPD,- ; Insert datapath and valid bit,
53 0B 011C 270 #32-UBA$V_MAP_DPD,R3 ; clearing other PTE flags
81 53 D0 011E 271 MOVL R3,(R1)+ ; Load the map register
EE 52 F5 0121 272 SOBGTR R2,20$ ; Loop through all registers
61 D4 0124 273 CLRL (R1) ; Invalidate last one to stop wild xfer
05 0126 274 RSB ;

```

```

0127 276          .SBTTL  LOAD UNIBUS ADAPTER MAP REGISTERS FOR UDA PORT
0127 277 ;+
0127 278 ; IOC$LUBAUDAMAP - LOAD UNIBUS ADAPTER MAP REGISTERS FOR UDA PORT
0127 279 ;
0127 280 ; INPUTS:
0127 281 ;
;RLRPDTADP      0127 .1 ;          R4 => PDT.
0127 282 ;          R5 => CDRP OF I/O REQUEST.
0127 283 ;
0127 284 ;          IT IS ASSUMED THAT THE DATAPATH AND MAP REGISTERS HAVE BEEN PREVIOUSLY
0127 285 ;          ASSIGNED.
0127 286 ;
0127 287 ; OUTPUTS:
0127 288 ;
0127 289 ;          EACH MAP REGISTER IS LOADED WITH THE APPROPRIATE PAGE FRAME NUMBER
0127 290 ;          MERGED WITH THE DATAPATH DESIGNATOR AND BYTE OFFSET BIT. ONE ADDITIONAL
0127 291 ;          MAP REGISTER IS LOADED AS INVALID TO STOP THE TRANSFER IF A HARDWARE
0127 292 ;          FAILURE SHOULD OCCUR.
0127 293 ;
0127 294 ;          R3 IS PRESERVED ACROSS CALL.
0127 295 ; -
0127 296
0127 297 IOC$LUBAUDAMAP::          ;LOAD UNIBUS ADAPTER MAP REGISTERS
0127 298
;RLRPDTADP      7E 53 7D 0127 299          MOVQ  R3,-(SP)          ;SAVE REGISTERS
;RLRPDTADP      012A .1
;RLRPDTADP      00E0 D4 DD 012A .2          ASSUME  ADP$L_CSR          EQ 0
;RLRPDTADP      012E .4          PUSHL  @PDT$L_ADP(R4)          ; Push UBA CSR address on stack.
;RLRPDTADP      51 D0 A5 3C 012E 300          MOVZWL CDRP$W_BOFF(R5),R1          ; R1=BYTE OFFSET IN PAGE
52 D2 A5 3C 0132 301          MOVZWL CDRP$W_BCNT(R5),R2          ; R2=TRANSFER BYTE COUNT
50 3C A5 9E 0136 302          MOVAB  CDRP$L_UBARSRC(R5),R0          ; R0 => MAPPING RESOURCE DESCRIPTOR
;RLRPDTADP      00 EF 013A 303          EXTZV  #VEC$V_DATAPATH,-          ; GET DATAPATH
;RLRPDTADP      05 013C 304          #VEC$S_DATAPATH,-          ; NUMBER
54 03 A0          013D 305          UBMD$B_DATAPATH(R0),R4
;RLRPDTADP      03 51 E9 0140 306          ;
;RLRPDTADP      54 10 88 0143 307          BLBC  R1,10$          ; IF LBC WORD ALIGNED TRANSFER
;RLRPDTADP      54 0400 8F A8 0146 308          BISB  #^X10,R4          ; SET BYTE OFFSET BIT
52 01FF C241 9E 014B 310          BISW  #^X400,R4          ; MERGE VALID WITH BYTE OFFSET AND DATAPATH
;RLRPDTADP      52 F7 8F 78 0151 311          MOVAB  ^X1FF(R2)[R1],R2          ; CALCULATE HIGHEST RELATIVE BYTE AND ROUND
;RLRPDTADP      52          ASHL  #-9,R2,R2          ; CALCULATE NUMBER OF MAP REGISTERS TO LOAD
;RLRPDTADP      0155 312
;RLRPDTADP      51 8ED0 0156 .1          POPL  R1          ; R1 => UBA CSR.
;RLRPDTADP      02 A0 52 91 0159 .2          CMPB  R2,UBMD$B_NUMREG(R0)          ; ENOUGH MAP REGISTERS ASSIGNED?
;RLRPDTADP      28 1E 015D .3          BGEQU 40$          ; IF GEQU NO
;RLRPDTADP      015F .4
;RLRPDTADP      00 EF 015F .5          EXTZV  #VEC$V_MAPREG,-          ; GET STARTING REGISTER
;RLRPDTADP      0F 0161 322          #VEC$S_MAPREG,-
-9          50 60          0162 323          UBMD$W_MAPREG(R0),R0
;RLRPDTADP      0164 324
51 0800 C140 DE 0164 325          MOVAL  UBA$L_MAP(R1)[R0],R1          ; GET ADDRESS OF FIRST MAP REGISTER TO LOAD
;RLRPDTADP      50 CC A5 D0 016A 326          MOVL  CDRP$L_SVAPTE(R5),R0          ; GET ADDRESS OF PAGE TABLE
;RLRPDTADP      53 80 D0 016E 327 20$:          MOVL  (R0)+,R3          ; GET NEXT PAGE TABLE ENTRY
;RLRPDTADP      03 19 0171 328          BLSS  30$          ; IF LSS VALID PAGE TABLE ENTRY
;RLRPDTADP      FF5E 30 0173 329          BSBW  IOC$PTETOPFN          ; GET PFN FROM INVALID PTE
;RLRPDTADP      0B 15 54 F0 0176 330 30$:          INSV  R4,#21,#11,R3          ; INSERT VALID, BYTE OFFSET, AND DATAPATH

```



LOADMREG  
V03-002

- LOAD MBA AND UBA MAP REGISTERS 3-JUN-1984 11:45:48 VAX-11 Macro V03-01 Page 9  
LOAD UNIBUS ADAPTER MAP REGISTERS FOR UD 12-MAR-1982 17:12:47 DISK\$VMSMASTER:[SYS.SRC]LOADMREG.M(1)

```
      53      017A
81  53  D0  017B  331      MOVL   R3,(R1)+      ;LOAD UBA MAP REGISTER
      ED 52  F5  017E  332      SOBGTR R2,20$      ;ANY MORE TO LOAD?
      61  D4  0181  333      CLRL   (R1)      ;LOAD INVALID MAP ENTRY
53  8E  7D  0183  334      MOVQ   (SP)+,R3      ;RESTORE REGISTERS
      05  0186  335      RSB      ;
      0187  336 40$:  BUG_CHECK UBMAPEXCED,FATAL ;UNIBUS MAP REGISTER ALLOCATION EXCEEDED
      018B  337      .END
```

```
ADP$L_CSR = 00000000
BUG$_INVPTEFMT ***** X 02
BUG$_UBMAPEXCED ***** X 02
CDRP$L_SVAPTE = FFFFFFFC
CDRP$L_UBARSRCE = 0000003C
CDRP$W_BCNT = FFFFFFFD2
CDRP$W_BOFF = FFFFFFFD0
CRB$L_INTD = 00000024
GLOBAL 000000CA R 02
IOC$LOADMBAMAP 00000000 RG 02
IOC$LOADUBAMAP 00000061 RG 02
IOC$LOADUBAMAPA 0000004A RG 02
IOC$LOADUBAMAPN 000000E8 RG 02
IOC$LUBAUDAMAP 00000127 RG 02
IOC$PTETOPFN 000000D4 RG 02
MBA$L_BCR = 00000010
MBA$L_MAP = 00000800
MBA$L_VAR = 0000000C
MMG$GL_GPTBASE ***** X 02
MMG$GL_SPTBASE ***** X 02
PDT$L_ADP = 000000E0
PTE$M_GPTX = 003FFFFFF
PTE$M_TYPO = 00400000
PTE$M_TYP1 = 04000000
PTE$V_TYPO = 00000016
PTE$V_TYP1 = 0000001A
UBA$L_MAP = 00000800
UBA$V_MAP_DPD = 00000015
UBA$V_MAP_VALID = 0000001F
UBMD$B_DATAPATH = 00000003
UBMD$B_NUMREG = 00000002
UBMD$W_MAPREG = 00000000
UCB$L_CRB = 00000024
UCB$L_SVAPTE = 00000078
UCB$W_BCNT = 0000007E
UCB$W_BOFF = 0000007C
VA$S_VPN = 00000015
VA$V_VPN = 00000009
VEC$B_DATAPATH = 00000013
VEC$B_NUMREG = 00000012
VEC$L_ADP = 00000014
VEC$S_DATAPATH = 00000005
VEC$S_MAPREG = 0000000F
VEC$V_DATAPATH = 00000000
VEC$V_LWAE = 00000005
VEC$V_MAPREG = 00000000
VEC$W_MAPREG = 00000010
```

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

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WIONONPAGED	0000018B ( 395.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.08	00:00:00.77
Command processing	75	00:00:00.61	00:00:06.93
Pass 1	285	00:00:09.21	00:00:38.62
Symbol table sort	0	00:00:01.35	00:00:06.01
Pass 2	67	00:00:01.95	00:00:07.61
Symbol table output	5	00:00:00.06	00:00:00.06
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	446	00:00:13.29	00:01:00.03

The working set limit was 1050 pages.  
48842 bytes (96 pages) of virtual memory were used to buffer the intermediate code.  
There were 50 pages of symbol table space allocated to hold 892 non-local and 16 local symbols.  
322 source lines were read in Pass 1, producing 14 object records in Pass 2.  
20 pages of virtual memory were used to define 19 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	12
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	16

991 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:LOADMREG/OBJ=OBJ\$:LOADMREG MSRC\$:LOADMREG/UPDATE=(ENH\$:LOADMREG)+EXECMLS/LIB



**IOCIOPST**

(1)	43	HISTORY ; DETAILED
(2)	79	DECLARATIONS
(3)	115	I/O COMPLETION POSTING
(4)	315	PAGIO - PAGE I/O COMPLETION
(5)	644.2	VIRTUAL (OR LOGICAL) I/O COMPLETION
(6)	734	QUEUE NEXT SEGMENT
(7)	856	BUFFERED READ COMPLETION AST ROUTINE
(8)	948	DIRECT I/O COMPLETION AST ROUTINE
(9)	1036.1	ERASE I/O HELPER ROUTINES
(10)	1037	MOVE DATA TO USER BUFFER
(11)	1054	UNLOCK AREAS IN IRPE'S

;ACG0422  
-1

```
0000 1 .TITLE IOCIOPST - I/O COMPLETION POSTING
0000 .1 .IDENT 'V03-024'
0000 3
0000 4 ;
0000 5 ;*****
0000 6 ;*
0000 7 ;* COPYRIGHT (c) 1978, 1980, 1982 BY
0000 8 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 ;* ALL RIGHTS RESERVED.
000Q 10 ;*
0000 11 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 ;* TRANSFERRED.
0000 17 ;*
0000 18 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 ;* CORPORATION.
0000 21 ;*
0000 22 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 ;*
0000 25 ;*
0000 26 ;*****
0000 27
0000 28 ;++
0000 29 ; FACILITY: EXECUTIVE, I/O SYSTEM
0000 30 ;
0000 31 ; ABSTRACT:
0000 32 ; IOCIOPST IMPLEMENTS THE DEVICE INDEPENDENT COMPLETION PROCESSING FOR
0000 33 ; I/O PACKETS. IT IS INVOKED BY QUEUEING THE PACKET ON THE I/O POST QUEUE
0000 34 ; AND TRIGGERING THE IPL$ IOPOST SOFTWARE INTERRUPT. SOME OF THE IOPOST
0000 35 ; OPERATIONS SUCH AS SETTING EVENT FLAGS, UNLOCKING BUFFER PAGES,
0000 36 ; RELEASING BUFFERS AND PAGING I/O COMPLETION ARE PERFORMED IN THE IOPOST
0000 37 ; INTERRUPT SERVICE ROUTINE, WHILE OTHER OPERATIONS THAT REQUIRE ACCESS
0000 38 ; TO PROCESS ADDRESS SPACE ARE PERFORMED BY SENDING A SPECIAL KERNEL AST.
0000 39 ;
0000 40 ; ENVIRONMENT: MODE = KERNEL, RESIDENT
0000 41 ;
0000 42 ;--
0000 43 .SBTTL HISTORY ; DETAILED
0000 44 ;
0000 45 ; AUTHOR: R. HUSTVEDT, CREATION DATE: 26-AUG-76
0000 46 ;
0000 47 ; MODIFIED BY:
0000 48 ;
0000 .1 ; V03-024 ACG0422 Andrew C. Goldstein, 1-May-1984 19:35
0000 .2 ; Fix use of R0 in ACG0421
0000 .3 ;
0000 .4 ; V03-023 ACG0421 Andrew C. Goldstein, 20-Apr-1984 14:19
0000 .5 ; Fix segment byte count limiting in erase QIO's
0000 .6 ;
0000 .7 ; V03-022 EMD0076 Ellen M. Dusseault 05-Apr-1984
0000 .8 ; Modify IOPOST to check for a valid status bit for
0000 .9 ; encryption. If valid, deallocate nonpaged pool buffer
```

;ACG0422  
;ACG0422  
;ACG0422  
;ACG0421  
;ACG0421  
;ACG0421  
;EMD0076  
;EMD0076  
;EMD0076

;EMD0076	0000	.10 ;		which contains the encryption key.	
;EMD0076	0000	.11 ;			
;SSA0021	0000	.12 ;	V03-021	SSA0021	Stan Amway 22-Mar-1984
;SSA0021	0000	.13 ;		Decrement device queue length in UCB.	
;SSA0021	0000	.14 ;			
;WMC0020	0000	.15 ;	V03-020	WMC0020	Wayne Cardoza 07-Mar-1984
;WMC0020	0000	.16 ;		Move POSTEF to fork context to regain optimization which	
;WMC0020	0000	.17 ;		avoids reexecution of WAITFR.	
;WMC0020	0000	.18 ;			
;WMC0019	0000	.19 ;	V03-019	WMC0019	Wayne Cardoza 28-Dec-1983
;WMC0019	0000	.20 ;		Erase QIOs can be physical, logical, or virtual.	
;WMC0019	0000	.21 ;			
;CDS0003	0000	.22 ;	V03-018	CDS0003	Christian D. Saether 14-Dec-1983
;CDS0003	0000	.23 ;		Add IOC\$BUFFPOST entry point. This is used to perform	
;CDS0003	0000	.24 ;		the iopost level part of i/o posting to be executed as	
;CDS0003	0000	.25 ;		a subroutine call directly and avoid the iopost software	
;CDS0003	0000	.26 ;		interrupt entirely. The F11BXQP is the initial user	
;CDS0003	0000	.27 ;		of this feature.	
;CDS0003	0000	.28 ;			
;ROW49597C	0000	.29 ;	V03-017	ROW49597C	Ralph O. Weber 21-SEP-1983
;ROW49597C	0000	.30 ;		Change PAGEIO_OR_SWAPIO patch (from ROW49597B and ROW49597) to	
;ROW49597C	0000	.31 ;		zero bytes transferred count in the IOSB when status is not	
;ROW49597C	0000	.32 ;		successful and bytes transferred is greater than or equal to	
;ROW49597C	0000	.33 ;		bytes requested.	
;ROW49597C	0000	.34 ;			
;ROW0218	0000	.35 ;	V03-016	ROW0218	Ralph O. Weber 7-SEP-1983
;ROW0218	0000	.36 ;		Change maximum byte count, UCB\$L_MAXBCNT, tests to be	
;ROW0218	0000	.37 ;		unsigned.	
;ROW0218	0000	.38 ;			
;ADE9005	0000	.39 ;	V03-015	ADE9005	Alan D. Eldridge 30-May-1983
;ADE9005	0000	.40 ;		Changed BSBW to JSB for calls to IOC\$MAPVBLK and IOC\$CVTLOGPHY.	
;ADE9005	0000	.41 ;			
;STJ3100	0000	.42 ;	V03-014	STJ3100	Steven T. Jeffreys, 03-May-1983
;STJ3100	0000	.43 ;		-Added local subroutine CHECK_ERASE.	
;STJ3100	0000	.44 ;		-Do not update IRP\$L_SVAPTE for ALL erase I/O's. This	
;STJ3100	0000	.45 ;		is an extension of STJ3085.	
;STJ3100	0000	.46 ;			
;STJ3085	0000	.47 ;	V03-013	STJ3085	Steven T. Jeffreys, 13-Apr-1983
;STJ3085	0000	.48 ;		-Do not update IRP\$L_SVAPTE for erase I/O segmented	
;STJ3085	0000	.49 ;		requests if using the special erase PPT.	
;STJ3085	0000	.50 ;		-After segmentation complete, restore original SVAPTE	
;STJ3085	0000	.51 ;		address to IRP\$L_SVAPTE.	
;STJ3085	0000	.52 ;			
;ROW49597B	0000	.53 ;	V03-012	ROW49597B	Ralph O. Weber 9-APR-1983
;ROW49597B	0000	.54 ;		Change PAGEIO_OR_SWAPIO from ROW49597 to zero bytes	
;ROW49597B	0000	.55 ;		transferred count when status is not successful and bytes	
;ROW49597B	0000	.56 ;		transferred is greater than or equal to bytes requested.	
;ROW49597B	0000	.57 ;			
;RLRMXBCNTC	0000	.58 ;	V03-011	RLRMXBCNTc	Robert L. Rappaport 28-Mar-1983
;RLRMXBCNTC	0000	.59 ;		Verify IRP\$L_DIAGBUF is non-zero before assuming that it	
;RLRMXBCNTC	0000	.60 ;		contains the original value of IRP\$L_SVAPTE in VIRTUAL_LOGIO.	
;RLRMXBCNTC	0000	.61 ;			
;RLRMXBCNTC	0000	.62 ;	V03-010	RLRMXBCNTb	Robert L. Rappaport 22-Mar-1983
;RLRMXBCNTC	0000	.63 ;		Check for file oriented device before going to VIRTUAL_LOGIO.	
;RLRMXBCNTC	0000	.64 ;			
;RLRMXBCNTA	0000	.65 ;	V03-009	RLRMXBCNTa	Robert L. Rappaport 22-Mar-1983
;RLRMXBCNTA	0000	.66 ;		CLRL the byte count in the I/O status before queuing	



```
;RLRMXBCNTA      0000 .67 ;
;RLRMXBCNTA      0000 .68 ;
;RLRMXBCNT       0000 .69 ;
;RLRMXBCNT       0000 .70 ;
;RLRMXBCNT       0000 .71 ;
;RLRMXBCNT       0000 .72 ;
;ROW49597        0000 .73 ;
;ROW49597        0000 .74 ;
;ROW49597        0000 .75 ;
;ROW49597        0000 .76 ;
;ROW49597        0000 .77 ;
;ROW49597        0000 .78 ;
;ROW49597        0000 .79 ;
;ROW49597        0000 .80 ;
;ROW49597        0000 .81 ;
;ROW49597        0000 .82 ;
;ROW49597        0000 .83 ;
;STJ3049         0000 .84 ;
;STJ3049         0000 .85 ;
;STJ3049         0000 .86 ;
;CDS0002         0000 .87 ;
;CDS0002         0000 .88 ;
;CDS0002         0000 .89 ;
;CDS0002         0000 .90 ;
;CDS0001         0000 .91 ;
;CDS0001         0000 .92 ;
;CDS0001         0000 .93 ;
;KDM0002         0000 .94 ;
;KDM0002         0000 .95 ;
;KDM0002         0000 .96 ;
;LJK45299        0000 .97 ;
;LJK45299        0000 .98 ;
;LJK45299        0000 .99 ;
-27              0000 76 ;
                0000 77 ;**
```

```
an IRP back to the ACP in VIRTUAL_LOGIO.
V03-008 RLRMXBCNT      Robert L. Rappaport      11-Mar-1983
Allow for segmentation of Logical I/O (and Virtual)
based on the UCB$_MAXBCNT field.
V03-007 ROW49597      Ralph O. Weber          26-JAN-1983
Change both VIRTUAL and PAGEIO_OR_SWAPIO to guarantee an error
status in IRP$_IOST1 whenever the bytes transferred is less
than the bytes requested. For V3.x, the error will be
SS$_CTRLERR. After that, it will be SS$_INCSEGTRA. The check
and error status are required to detect and gracefully
recovered from the instance where a driver returns success
status but bytes transferred is less than bytes requested.
The segmented transfer logic goes berserk when this happens
and eventually crashes the system.
V03-006 STJ3049      Steven T. Jeffreys      06-Jan-1983
Add support for the erase qio.
V03-005 CDS0002      C Saether          12-Oct-1982
Fix bug where R5 was not preserved when queuing
packet to xqp.
V03-004 CDS0001      C Saether          18-Jul-1982
Changes to accomodate XQP mechanism.
V03-003 KDM0002      Kathleen D. Morse      28-Jun-1982
Added $DEVDEF and $$SDEF.
V03-002 LJK45299      Lawrence J. Kenah      2-Jun-1982
Fix deaccess-pending-on-spoiled-device logic.
```



```

0004 115 .SBTTL I/O COMPLETION POSTING
0004 116 ;++
0004 117 ; FUNCTIONAL DESCRIPTION:
0004 118 ;
0004 119 ; IOC$IOPST IS INITIATED BY TRIGGERING AN IPL$ IOPOST SOFTWARE
0004 120 ; INTERRUPT AFTER PLACING A COMPLETED I/O PACKET IN THE IOPOST
0004 121 ; QUEUE. IOC$IOPST PERFORMS ALL APPROPRIATE COMPLETION ACTIVITY
0004 122 ; REQUIRED FOR THE PACKET EITHER DIRECTLY OR BY QUEUEING KERNEL
0004 123 ; ASTS TO CONCLUDE PROCESSING IN THE CONTEXT OF THE PROCESS
0004 124 ; WHEN REQUIRED.
0004 125 ;
0004 126 ; CALLING SEQUENCE:
0004 127 ;
0004 128 ; SOFTINT #IPL$_IOPOST
0004 129 ;
0004 130 ; INPUT PARAMETERS:
0004 131 ;
0004 132 ; NONE
0004 133 ;
0004 134 ; IMPLICIT INPUTS:
0004 135 ;
0004 136 ; IOC$GL_PSFL - IOPOSTING QUEUE
0004 137 ;
0004 138 ; OUTPUT PARAMETERS:
0004 139 ;
0004 140 ; NONE
0004 141 ;
0004 142 ;--
0004 143
0004 144 .ENABL LSB
0004 145 IOC$IOPST::
55 7E 54 7D 0004 146 MOVQ R4,-(SP) ; I/O POSTING INTERRUPT
7E 52 7D 0007 147 MOVQ R2,-(SP) ; SAVE
7E 50 7D 000A 148 MOVQ R0,-(SP) ; NORMAL
; SSA0021 000D 149 IOPOST: REMQUE @W^IOC$GL_PSFL,R5 ; REGISTERS
; SSA0021 16 1C 0012 150 BVC 10$ ; GET HEAD OF POST QUEUE
; SSA0021 50 8E 7D 0014 151 MOVQ (SP)+,R0 ; QUEUE NOT YET EMPTY
; SSA0021 52 8E 7D 0017 152 MOVQ (SP)+,R2 ; RESTORE
; SSA0021 54 8E 7D 001A 153 MOVQ (SP)+,R4 ; REGISTERS
; SSA0021 02 001D 154 REI ; AND EXIT
; SSA0021 001E 155 ; IF QUEUE EMPTY
; RLRMXBCNT 038B 31 001E .1 5$: BRW VIRTUAL_LOGIO ; PROCESS VIRTUAL (OR LOGICAL) I/O COMPLETIO
; SSA0021 0021 .2
; SSA0021 61 16 0021 .3 7$: JSB (R1) ; CALL END ACTION ROUTINE
; SSA0021 E8 11 0023 .4 BRB IOPOST ;
; SSA0021 0025 .5
; SSA0021 6A A0 B4 0025 .6 8$: CLRW UCB$W_QLEN(R0) ; Device queue length went negative
; SSA0021 18 11 0028 .7 BRB 11$ ; Reset queue length and continue
-4 002A 160
51 0C A5 D0 002A 161 10$: MOVL IRP$L_PID(R5),R1 ; GET PID/END ACTION ADDRESS
F1 19 002E 162 BLSS 7$ ; BR IF END ACTION ADDRESS
; SSA0021 0030 163 ; (SYSTEM SPACE ADDRESSES ARE NEGATIVE)
; SSA0021 51 51 3C 0030 164 MOVZWL R1,R1 ; GET PROCESS INDEX
54 0000'DF41 D0 0033 165 MOVL @W^SCH$GL_PCBVEC[R1],R4 ; AND TRANSLATE TO PCB ADDRESS
; EMD0076 50 1C A5 D0 0039 .1 MOVL IRP$L_UCB(R5),R0 ; R0 => UCB. (Presets UCB for DIO path)
; EMD0076 6A A0 B7 003D .2 DECW UCB$W_QLEN(R0) ; Decrement device queue length
; EMD0076 E3 19 0040 .3 BLSS 8$ ; Length went negative, so go adjust

```

```

;EMD0076      07 2A A5 0F E1 0042 .4 11$: BBC #IRP$V_KEY,IRP$W_STS(R5),12$ ; set, buffer alloc for encryption
;EMD0076      50 5C A5 D0 0047 .5      MOVL IRP$L_KEYDESC(R5), R0 ; r0 contains address of alloc buffer
;EMD0076      FFB2 30 004B .6      BSBW EXE$DEANONPAGED ; deallocate this buffer
;EMD0076      03 2A A5 00 E1 004E .7 12$: BBC #IRP$V_BUFIO,IRP$W_STS(R5),13$ ; IF CLEAR, DIRECT I/O
;EMD0076      00C6 31 0053 .8      BRW BUFIO ; BUFFERED I/O
;EMD0076      3E A4 B6 0056 .9 13$: INCW PCB$W_DIOCNT(R4) ; UPDATE DIRECT I/O COUNT
-3           53 2C A5 D0 0059 169 MOVL IRP$L_SVAPTE(R5),R3 ; GET ADDRESS OF FIRST PTE
           005D 170
           005D 171 ASSUME IRP$V_PAGIO LE 7
           005D 172 ASSUME IRP$V_SWAPIO LE 7
           2A A5 44 8F 93 005D 173 BITB #<IRP$M_PAGIO ! IRP$M_SWAPIO>,IRP$W_STS(R5) ; PAGIO OR SWAPIO?
           5D 12 0062 174 BNEQ PAGIO_OR_SWAPIO
           0064 175
           0064 176 ;
           0064 177 ; DIRECT I/O COMPLETION
           0064 178 ;
           0064 179
           53 D5 0064 180 DIRIO: TSTL R3 ; PTE ADDRESS VALID?
           42 13 0066 181 BEQL 18$ ; IF EQL NO PAGES TO UNLOCK
51 32 A5 D0 0068 182 MOVL IRP$L_BCNT(R5),R1 ; GET REQUESTED TRANSFER BYTE COUNT
52 30 A5 3C 006C 183 MOVZWL IRP$W_BOFF(R5),R2 ; GET BYTE OFFSET IN PAGE
;SSA0021      08 E0 0070 .1      BBS #IRP$V_PHYSIO,-
;SSA0021      1C 2A A5 0072 .2      IRP$W_STS(R5),UNLOCK ; BRANCH IF PHYSICAL I/O
;SSA0021      0E E1 0075 .3      BBC #DEV$V_FOD,- ; If NOT file oriented, go to UNLOCK.
;SSA0021      17 38 A0 0077 .4      UCB$L_DEVCHAR(R0),UNLOCK; (R0 preloaded in common DIO/BIO path)
;SSA0021      A0 38 A5 E9 007A .5      BLBC IRP$L_IOST1(R5),5$ ; BRANCH IF ERROR IN VIRT. OR LOG. REQUEST
-2           46 A5 B5 007E 186 TSTW IRP$L_OBCNT+2(R5) ; WAS ORIGINAL COUNT > 64K?
           07 13 0081 187 BEQL 14$ ; EQL IMPLIES NO
           3A A5 D1 0083 188 CMLP IRP$L_IOST1+2(R5),- ; LONGWORD COMPARE FOR > 64K OBCNT
           44 A5 0086 189 IRP$L_OBCNT(R5) ; IF COMPLETED ORIGINAL BYTE COUNT
           0088 190 ; THEN NO SPECIAL VIRTUAL PROCESSING
           05 11 0088 191 BRB 16$ ; BRANCH AROUND TO COMMON 'BNEQ'
           008A 192 14$:
           3A A5 B1 008A 193 CMPW IRP$L_IOST1+2(R5),- ; *NOTE 'CMPW' DUE TO CODE PATH FOR <64K BCN
           44 A5 008D 194 IRP$L_OBCNT(R5) ; IF COMPLETED ORIGINAL BYTE COUNT
           008F 195 ; THEN NO SPECIAL VIRTUAL PROCESSING
           008F 196 16$:
           8D 12 008F 197 BNEQ 5$ ; OTHERWISE DO THE SEGMENTED COMPLETION
51 01FF C142 9E 0091 198 UNLOCK: MOVAB 511(R1)[R2],R1 ; COMBINE OFFSET AND COUNT AND ROUND
51 F7 8F 78 0097 199 ASHL #-VA$$_BYTE,R1,R1 ; CONVERT TO NUMBER OF PAGES
           51 009B
;STJ3100      0A E1 009C .1      BBC #IO$V_ERASE,- ; BRANCH IF DEFINITELY NOT AN ERASE
;STJ3100      06 20 A5 009E .2      IRP$W_FUNC(R5),17$ ;
;STJ3100      066C 30 00A1 .3      BSBW CHECK_ERASE ; IS THIS AN ERASE FUNCTION?
;STJ3100      0B 50 E8 00A4 .4      BLBS R0,19$ ; BRANCH IF IT IS AN ERASE
;STJ3100      FF56 30 00A7 .5 17$: BSBW MMG$UNLOCK ; UNLOCK PAGES
-1           0B E1 00AA 201 18$: BBC #IRP$V_EXTEND,-
           03 2A A5 00AC 202 IRP$W_STS(R5),19$ ; BRANCH IF NO IRPE'S ATTACHED
           06B3 30 00AF 203 BSBW UNLOCK_MORE ; UNLOCK AREAS DESCRIBED IN IRPE'S
           00B2 204 19$: ; REFERENCE LABEL
           00B2 205
           00B2 206 .IF DF CA$_MEASURE_IOT
           00B2 207
           FF4B' 30 00B2 208 BSBW PMS$END_RQ ; INSERT END OF I/O REQUEST MESSAGE
           00B5 209
           00B5 210 ENDC
           00B5 211

```

```

;CDS0003      0648'CF  9E  00B5  .1      MOVAB  W^DIRPOST,ACB$L_KAST(R5) ; SET SPECIAL KERNEL AST ADDRESS
              18 A5  00B9
;CDS0003      009A  31  00BB  .2      BRW    40$
;CDS0003      00BE  .3
;WMC0020      00BE  .4 BRW_QNXTSEG:
;WMC0020      034A  31  00BE  .5      BRW    QNXTSEG          ; GO DO THE NEXT VIRTUAL SEGMENT
;WMC0020      00C1  .6
-2            00C1  214 ;
              00C1  215 ;          PAGE I/O OR SWAP I/O COMPLETION
              00C1  216 ;
              00C1  217
              00C1  218 PAGIO_OR_SWAPIO:          ; HERE WE ASSUME DISK I/O FOR PAGING
              00C1  219          ; AND SWAPPING AND WE FURTHER RELY
              00C1  220          ; ON THE FACT THAT ALL DISK DRIVERS
              00C1  221          ; TRADITIONALLY RETURN ZERO IN THE 2ND
              00C1  222          ; LONGWORD OF THE I/O STATUS BLOCK FOR
              00C1  223          ; DATA TRANSFER OPERATIONS. THEREFORE
              00C1  224          ; THIS IS COMPATIBLE WITH DISK CLASS
              00C1  225          ; DRIVER WHICH GROWS THE # OF BYTES
              00C1  226          ; TRANSFERRED FIELD IN THE IOSB TO A
              00C1  227          ; FULL LONGWORD.
;ROW49597     51   3A A5  D0  00C1  .1      MOVL   IRP$L_IOST1+2(R5), R1 ; Get bytes transferred.
;ROW49597B    12  38 A5  E9  00C5  .2      BLBC   IRP$L_IOST1(R5), 21$ ; Branch if transfer not successful.
;ROW49597B    44  A5  51  D1  00C9  .3      CMPL   R1, IRP$L_OBCNT(R5) ; If completed whole transfer, skip
;ROW49597B    3E   3E  13  00CD  .4      BEQL   26$ ; all this segmenting junk.
;ROW49597B    32  A5  51  D1  00CF  .5      CMPL   R1, IRP$L_BCNT(R5) ; Bytes transferred = bytes requested?
;ROW49597B    11   11  13  00D3  .6      BEQL   23$ ; Branch if equal.
;ROW49597B    2234 8F  B0  00D5  .7      MOVW   #$$$_INCSEGTRA, - ; Else, change success status
              38 A5  00D9
;ROW49597B    00DB  .8      IRP$L_IOST1(R5) ; to error status.
;ROW49597B    00DB  .9      ; For the error cases:
;ROW49597B    32  A5  51  D1  00DB  .10 21$: CMPL   R1, IRP$L_BCNT(R5) ; Bytes transferred < bytes requested?
;ROW49597B    05   05  1F  00DF  .11      BLSSU  23$ ; Branch if less than.
;ROW49597B    51   51  D4  00E1  .12      CLRL   R1 ; Else, assume no bytes transferred.
;ROW49597C    3A A5  D4  00E3  .13      CLRL   IRP$L_IOST1+2(R5) ; Clear bytes transferred in IRP too.
;ROW49597     40  A5  51  C0  00E6  .14 23$: ADDL   R1, IRP$L_ABCNT(R5) ; Update accumulated byte count.
;ROW49597     51  17  09  EF  00EA  .15      EXTZV  #VA$V_VPN, - ; Convert bytes transferred to
              51  00EE
;ROW49597     00EF  .16      #<32-VA$V_VPN>, R1, R1 ; pages transferred.
-18           48  A5  51  C0  00EF  246      ADDL   R1,IRP$L_SEGVBN(R5) ; NEXT STARTING VBN (OR ERROR VBN)
              09  38  A5  E9  00F3  247      BLBC   IRP$L_IOST1(R5),24$ ; BRANCH IF ERROR
              40  A5  C3  00F7  248      SUBL3  IRP$L_ABCNT(R5),-
              44  A5  00FA  249      IRP$L_OBCNT(R5),- ; CALCULATE REMAINING BYTE
              32  A5  00FC  250      IRP$L_BCNT(R5) ; COUNT TO BE TRANSFERRED
              BE   12  00FE  251      BNEQ   BRW_QNXTSEG ; BRANCH IF ANOTHER SEGMENT TO DO
              0100  252 ;
              0100  253 ; LAST SEGMENT COMPLETED OR ERROR
              0100  254 ;
              0100  255 24$:
              40  A5  D0  0100  256      MOVL   IRP$L_ABCNT(R5),-
              3A  A5  0103  257      IRP$L_IOST1+2(R5) ; SET BYTES TRANSFERRED
              53  4C  A5  D0  0105  258      MOVL   IRP$L_DIAGBUF(R5),R3 ; GET SAVED SVAPTE
              2C  A5  53  D0  0109  259      MOVL   R3,IRP$L_SVAPTE(R5) ; AND PUT IT BACK
              010D  260 26$:
              010D  261          IF DF CA$_MEASURE_IOT
              010D  262
              FEFO' 30  010D  263      BSBW   PMS$END_RQ          ; INSERT END OF I/O REQUEST MESSAGE

```

```

0110 264
0110 265 .ENDC
0110 266
76 2A A5 02 E0 0110 267 BBS #IRP$V_PAGIO,IRP$W_STS(R5),PAGIO ; BRANCH IF PAGE I/O
0115 268
0115 269 ;
0115 270 ; SWAP I/O COMPLETION
0115 271 ;
0115 272
18 A5 14 A5 D0 0115 273 MOVL IRP$L_ASTPRM(R5),ACB$L_KAST(R5) ; SET KERNEL AST ADDRESS
3C 11 011A 274 BRB 40$ ; AND ENQUEUE AST
011C 275
011C 276 ;
011C 277 ; BUFFERED I/O COMPLETION
011C 278 ;
011C 279
;CDS0003 00000158'EF 9F 011C .1 BUFIO: PUSHAB 40$ ; 'INLINE' SUBROUTINE CALL.
;CDS0003 0122 .2
;CDS0003 0122 .3 ;
;CDS0003 0122 .4 ; THE FOLLOWING PIECE OF CODE MAY BE CALLED AS A SUBROUTINE DIRECTLY
;CDS0003 0122 .5 ; TO DO THE PART OF BUFFERED I/O COMPLETION THAT NORMALLY EXECUTES
;CDS0003 0122 .6 ; AS A RESULT OF AN IOPOST SOFTWARE INTERRUPT.
;CDS0003 0122 .7 ;
;CDS0003 0122 .8 ; THE F11BXQP, FOR EXAMPLE, EXECUTES VIRTUAL FILE SYSTEM FUNCTIONS
;CDS0003 0122 .9 ; IN PROCESS CONTEXT. THERE IS NO NEED FOR THE IOPOST INTERRUPT
;CDS0003 0122 .10 ; AND SPECIAL KERNEL AST TO POST I/O COMPLETION. AFTER RETURNING
;CDS0003 0122 .11 ; FROM THIS SUBROUTINE, THE F11BXQP WILL DO A
;CDS0003 0122 .12 ;
;CDS0003 0122 .13 ; JSB @ACB$L_KAST (R5)
;CDS0003 0122 .14 ;
;CDS0003 0122 .15 ; TO COMPLETE POSTING THE I/O COMPLETION.
;CDS0003 0122 .16 ; BOTH THE IOPOST SOFTWARE INTERRUPT AND THE SPECIAL KERNEL COMPLETION
;CDS0003 0122 .17 ; AST ARE AVOIDED.
;CDS0003 0122 .18 ;
;CDS0003 0122 .19 ; THE CALLER SHOULD TEST IRP$L_PID AND POST A NORMAL IOPOST INTERRUPT
;CDS0003 0122 .20 ; IF IT IS NEGATIVE, AS THAT CASE IS NOT HANDLED HERE.
;CDS0003 0122 .21 ;
;CDS0003 0122 .22 ; THE F11BXQP CODE THAT USES THIS ROUTINE IS IN [F11X.SRC]IODONE.MAR.
;CDS0003 0122 .23 ;
;CDS0003 0122 .24 ; IPL = IPL$ ASTDEL TO BLOCK PROCESS DELETION (PREVENT LOSS OF IRP).
;CDS0003 0122 .25 ; R4 = PCB ADDRESS
;CDS0003 0122 .26 ; R5 = IRP ADDRESS
;CDS0003 0122 .27 ;
;CDS0003 0122 .28
;CDS0003 0122 .29 IOC$BUFPOST::
-1 3A A4 B6 0122 .30 INCW PCB$W_BIOCNT(R4) ; UPDATE BUFFERED I/O COUNT
03 2A A5 0C E1 0125 281 BBC #IRP$V_FILACP,IRP$W_STS(R5),NOTACP ; BR IF NOT ACP I/O
3E A4 B6 012A 282 INCW PCB$W_DIOCNT(R4) ; RESTORE DIRECT I/O COUNT
012D 283 NOTACP:
012D 284
012D 285 .IF DF CA$_MEASURE_IOT
012D 286
FED0' 30 012D 287 BSBW PMS$END_RQ ; INSERT END OF I/O REQUEST MESSAGE
0130 288
0130 289 .ENDC
0130 290
50 0080 C4 D0 0130 291 MOVL PCB$L_JIB(R4),R0 ; GET JIB ADDRESS

```

	51	30	A5	3C	0135	292	MOVZWL	IRP\$W_BOFF(R5),R1	; Convert I/O byte count to a longword.
	20	A0	51	C0	0139	293	ADDL	R1,JIB\$L_BYTCNT(R0)	; Update Byte Count Quota.
	50	2C	A5	D0	013D	294	MOVL	IRP\$L_SVAPTE(R5),R0	; ANY BUFFER SPECIFIED?
			0E	13	0141	295	BEQL	30\$	; IF EQL NO
		0566	'CF	9E	0143	296	MOVAB	W^BUFPOST,ACB\$L_KAST(R5)	; ASSUME READ FUNCTION
			18	A5	0147				
;CDS0003	09	2A	A5	01	E0	0149	.1	BBS	#IRP\$V_FUNC,IRP\$W_STS(R5),35\$ ; IF SET, READ FUNCTION
;CDS0003			FEAF	'	30	014E	.2	BSBW	EXE\$DEANONPAGED ; DEALLOCATE WRITE BUFFER
;CDS0003		0648	'CF	9E	0151	.3	30\$:	MOVAB	W^DIRPOST,ACB\$L_KAST(R5) ; SET SPECIAL KERNEL AST ADDRESS
			18	A5	0155				
;CDS0003				05	0157	.4	35\$:	RSB	; RETURN TO PROCESS CONTEXT IOPOSTING
;CDS0003					0158	.5			; PROCESS, OR CONTINUE INLINE IF THIS
;CDS0003					0158	.6			; IS NORMAL IOPOST SOFTWARE INTERRUPT.
-3		02	00	EF	0158	300	40\$:	EXTZV	#IRP\$V_BUFIO,#2,IRP\$W_STS(R5),R0 ; GET PACKET TYPE
	50	2A	A5		015B				
	03	2A	A5	09	E0	015E	301	BBS	#IRP\$V_TERMIO,IRP\$W_STS(R5),50\$ ; BR IF TERMINAL I/O
		50	01	AA	0163	302		BICW	#1,R0 ; ELSE TREAT AS NORMAL I/O COMPLETION
					0166	303	50\$:		; FOR PRIORITY INCREMENT SELECTION
	51	0C	A5	D0	0166	304		MOVL	IRP\$L_PID(R5),R1 ; PROCESS IDENTIFICATION
;WMC0020	52	FE91	CF40	9A	016A	305		MOVZBL	PRITBL[R0],R2 ; SET PRIORITY INCREMENT CLASS
;WMC0020		53	22	A5	9A	0170	.1	MOVZBL	IRP\$B_EFN(R5),R3 ; GET EVENT FLAG NUMBER
;WMC0020					0174	.2		DSBINT	#IPL\$_SYNCH ; PREVENT INTERRUPT FROM MP SECONDARY
;WMC0020		FE83	'	30	017A	.3		BSBW	SCH\$P\$OSTEF ; AND POST IT
;WMC0020	0B	A5	80	8F	88	017D	.4	BISB	#^X80,ACB\$B_RMOD(R5) ; SET INTERNAL AST FLAG
;WMC0020			FE7B	'	30	0182	.5	BSBW	SCH\$QAST ; NOW QUEUE THE KERNEL AST
;WMC0020					0185	.6		ENBINT	
;WMC0020		FE82		31	0188	.7		BRW	IOPOST ; GET NEXT PACKET TO POST
;WMC0020					018B	.8		.DSABL	LSB

```

018B 315          SBTTL  PAGIO - PAGE I/O COMPLETION
018B 316 ;
018B 317 ; PAGING I/O COMPLETION
018B 318 ;
018B 319 ; INPUTS:
018B 320 ;
018B 321 ;      R3 = SYSTEM VIRTUAL ADDRESS OF PAGE TABLE ENTRY
018B 322 ;      R4 = PROCESS CONTROL BLOCK ADDRESS
018B 323 ;      R5 = I/O REQUEST PACKET ADDRESS
018B 324 ;
018B 325 ;          FOR PAGE READ COMPLETION, THE FOLLOWING LOCATIONS IN THE
018B 326 ; I/O REQUEST PACKET HAVE SPECIAL SIGNIFICANCE.
018B 327 ;
018B 328 ;      IRP$L_ASTPRM    = ORIGINAL PROCESS PAGE TABLE ENTRY BACKING STORE
018B 329 ;                      ADDRESS IF PAGE WAS A COPY ON REFERENCE PAGE.
018B 330 ;                      PFN$V_GBLBAK SET IF IT WAS GLOBAL CRF
018B 331 ;                      = 0 IF NOT A COPY ON REFERENCE PAGE
018B 332 ;      IRP$L_AST      = MASTER PTE CONTENTS IF GLOBAL CRF (>0)
018B 333 ;                      = SLAVE PTE ADDRESS IF GLOBAL NOT CRF (<0)
018B 334 ;                      = 0 IF NOT GLOBAL
018B 335 ;
018B 336 ;          FOR PAGE WRITE COMPLETION, THE FOLLOWING LOCATIONS IN
018B 337 ; THE I/O REQUEST PACKET HAVE SIGNIFICANCE.
018B 338 ;
018B 339 ;      IRP$B_RMOD      = REQUEST MODE ! ACB$V_QUOTA.  IF ACB$V QUOTA IS SET,
018B 340 ;                      PROCESS REQUESTED AN_AST ON PAGE WRITE COMPLETION
018B 341 ;      IRP$L_AST      = AST ADDRESS IF REQUESTED
018B 342 ;      IRP$L_ASTPRM   = AST PARAMETER IF SPECIFIED
018B 343 ;      IRP$L_IOSB     = ADDRESS OF I/O STATUS BLOCK IF SPECIFIED.  IF
018B 344 ;                      NON-ZERO, THEN PROCESS EXPECTS I/O STATUS RETURNED.
018B 345 ;
018B 346 ;      PAGIO:  MOVQ    R6,-(SP)          ; SAVE SOME MORE REGISTERS
018B 347 ;                      MOVVL   R5,R6          ; USE R6 FOR IRP ADDRESS
0191 348 ;
0191 349 ;      SETIPL  #IPL$_SYNCH          ; SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
0191 350 ;                      MOVVL   PCB$_PHD(R4),R5      ; USE R5 FOR PROCESS HEADER ADR
0191 351 ;      EXTZV   #VA$_VPN,-          ;
0191 352 ;                      #<32-VA$_VPN>,-          ; FORM PAGE COUNT
019B 353 ;      IRP$L_IOST1+2(R6),R7      ; OF THE DATA TRANSFERRED
019E 354 ;      BBS     #IRP$_FUNC,IRP$_STS(R6),PAGRD_DONE ; BRANCH IF PAGE READ
01A3 355 ;
01A3 356 ;      PAGE WRITE COMPLETE - R7 = NUMBER OF PAGES
01A3 357 ;      CONDITION CODES SET FROM LOAD OF R7
01A3 358 ;
01A3 359 ;      BEQL    60$          ; BRANCH IF NO PAGES SUCCESSFULLY TRANSFERRE
01A5 360 ;      EXTZV   #PTE$_PFN,#PTE$_PFN,(R3),R0 ;GET PFN FROM PTE
01A9 361 ;
01AA 361 ;      CMPL    R0,MMG$_GL_MAXPFN      ;IS THIS PAGE IN SHARED MEMORY?
01AC 362 ;
01B1 362 ;      BGTRU   60$          ;BR IF PAGE IN SH MEM, NO PFN DATABASE
01B3 363 ;      20$:   PUSHL   R3          ; SAVE SVAPTE
01B5 364 ;      BSBW    PFN_IO_DONE        ; SET PFN DATA BASE
01B8 365 ;
01B8 366 ;      CONDITION CODES SET FROM DECREF
01B8 367 ;
01B8 368 ;      BGTR    40$          ; BRANCH IF REFCNT NOT 0
01BA 369 ;      BSBW    MMG$_RELPFN        ; RELEASE THE PAGE

```



```

53  8E  04  C1  01BD  370 40$:  ADDL3  #4,(SP)+,R3          ; GET NEXT PTE ADDRESS
      EF  57  F5  01C1  371          SOBGTR  R7,20$          ; DO THE NEXT PAGE IF ANY
      00C4 8F  A3  01C4  372 60$:  SUBW3  #IRP$C_LENGTH,IRP$W_SIZE(R6),R7 ; IF EXTENDED I/O PACKET
57  08  A6          01C8          ;
      01CB  373          ; THEN COMPLETION IS DONE BY
      01CB  374          ; SPECIAL UPDATE SECTION KERNEL AST
      04 38  A6  E9  01CB  375          BLBC  IRP$L_IOST1(R6),PAGWRT_ERR ; BRANCH IF PAGE WRITE ERROR
      01CF  376 ;
      01CF  377 ; CONDITION CODES SET FROM LOAD OF R7
      01CF  378 ;
      01CF  379 PAGWRT_ERR_DONE:
      68  13  01CF  380          BEQL  PAGIO_DONE1          ; BRANCH IF NOT, COMPLETE THE I/O HERE
      6D  11  01D-1 381          BRB  PAGIO_DONE2          ; COMPLETE I/O IN PROCESS CONTEXT
      01D3  382 PAGWRT_ERR:
      0150 31  01D3  383          BRW  PAGWRT_ERR1
      01D6  384 ;
      01D6  385 ; PAGE READ COMPLETE - R7 = NUMBER OF PAGES
      01D6  386 ; CONDITION CODES SET FROM LOAD OF R7
      01D6  387 ;
      01D6  388 PAGRD_DONE:
      3C  13  01D6  389          BEQL  100$          ; BRANCH IF NO PAGES SUCCESSFULLY TRANSFERRE
      01A0 30  01D8  390 20$:  BSBW  PFN_IO_DONE          ; RECORD PAGE READ DONE
      01DB  391 ;
      01DB  392 ; CONDITION CODES SET FROM DECREF
      01DB  393 ;
      11  14  01DB  394          BGTR  30$          ; BRANCH IF REFCNT NOT ZERO
      01DD  395 ;
      01DD  396 ; NO MORE REFERENCES FOR THIS PAGE, DON'T MAKE IT VALID, RELEASE IT
      01DD  397 ;
      04  A3  DF  01DD  398          PUSHAL 4(R3)          ; SAVE PTE ADR FOR NEXT PTE
      FE1D' 30  01E0  399          BSBW  MMG$RELPFN          ; RELEASE THE PFN
      08  BA  01E3  400          POPR  #^M<R3>          ; RECOVER PTE FOR NEXT PAGE IN CLUSTER
      10  A6  04  C1  01E5  401          ADDL3  #4,IRP$L_AST(R6),R1 ; GLOBAL PAGE?
      51          01E9
      25  18  01EA  402          BGEQ  80$          ; BRANCH IF IT ISN'T
      1F  11  01EC  403          BRB  60$          ; YES, SET CONTEXT FOR NEXT PAGE IN CLUSTER
0000'DF40 07  88  01EE  404 30$:  BISB  #PFN$C_ACTIVE,@W^PFN$AB_STATE[R0] ; PAGE IS NOW ACTIVE
      00  50  1F  E2  01F4  405          BBSS  #PTE$V_VALID,R0,40$ ; TURN VALID ON WITH PFN
      83  50  C8  01F8  406 40$:  BISL  R0,(R3)+          ; SET VALID IN PTE
      01FB  407          ; NEXT PTE ADDRESS IN R3
      51  10  A6  D0  01FB  408          MOVL  IRP$L_AST(R6),R1 ; GLOBAL PAGE?
      10  18  01FF  409          BGEQ  80$          ; BRANCH IF NOT
      0201  410 ;
      0201  411 ; PAGE IS A GLOBAL PAGE, R1 = PROCESS PTE, MUST MAKE IT VALID TOO
      0201  412 ;
      867FFFFF 8F  CB  0201  413          BICL3  #^C<PTE$M_PROT ! PTE$M_OWN>,(R1),R2 ; PROTECTION AND OWNER FIELDS
      52  61          0207
      81  52  50  C9  0209  414          BISL3  R0,R2,(R1)+          ; MAKE PROCESS PTE VALID
      10  A6  51  D0  020D  415 60$:  MOVL  R1,IRP$L_AST(R6) ; SET UP FOR NEXT PAGE IN CLUSTER
      C4  57  F5  0211  416 80$:  SOBGTR R7,20$          ; DO THE NEXT PAGE IF ANY
      7F 38  A6  E9  0214  417 100$: BLBC  IRP$L_IOST1(R6),PAGRD_ERR ; BRANCH IF PAGE READ ERROR
      0218  418 ;
      0218  419 ; LAST PAGE IN CLUSTER HAS BEEN PROCESSED, COMPLETE THE PROCESSING
      0218  420 ; ASSOCIATED WITH THE TRANSFER AS A WHOLE.
      0218  421 ;
      0218  422 PAGIO_DONE:
      51  14  A6  D0  0218  423          MOVL  IRP$L_ASTPRM(R6),R1 ; COPY ON REFERENCE SECTION?

```

```

      1B 13 021C 424      BEQL 20$          ; BRANCH IF NOT
OB 51 17 E1 021E 425      BBC #PFN$V_GBLBAK,R1,10$      ; BRANCH IF NOT GBL CRF
00000000'FF DE 0222 426      MOVAL @MMG$G_L_SYSPHD,R5      ; SYSTEM HDR FOR GBL CRF PAGE
      55      0228
      51 10 A6 D0 0229 427      MOVL IRP$L_AST(R6),R1      ; CONTENTS OF GBL PTE FOR GBL CRF
      51 51 32 022D 428 10$: CVTWL R1,R1      ; SECTION INDEX
      09 EF 0230 429      EXTZV #VA$V_VPN,-      ;
      17      0232 430      #<32-VA$V_VPN>,-      ; PAGE COUNT FROM
      50 3A A6      0233 431      IRP$L_IOST1+2(R6),R0      ; BYTE COUNT TRANSFERRED
      FDC7' 30 0236 432      BSBW MMG$SUBSECF      ; SUBTRACT R0 FROM SECTION REFERENC COUNT
      0239 433 ;
      0239 434 ; REPORT THAT PAGE I/O HAS COMPLETED.
      0239 435 ;
      0239 436 ; NORMALLY IT IS ONLY NECESSARY TO REPORT "PAGE FAULT COMPLETE"
      0239 437 ; TO THE PROCESS THAT INITIATED THE I/O, BUT FOR SYSTEM PAGES
      0239 438 ; AND FOR GLOBAL PAGES, MULTIPLE FAULTS CAN OCCUR FOR THE SAME
      0239 439 ; PAGE WHILE IT IS ON ITS WAY INTO MEMORY. ALL PROCESSES WHICH
      0239 440 ; FAULT THE PAGE WHILE ITS STATE IS "READ IN PROGRESS" GET QUEUED
      0239 441 ; ON THE COLLISION PAGE QUEUE, AND THE COLLISION BIT IS SET IN THE
      0239 442 ; TYPE BYTE OF THE PFN DATA BASE. THIS ROUTINE ALSO REPORTS THE
      0239 443 ; COLLISION PAGE AVAILABLE EVENT TO ALL PROCESSES ON THE COLLISION
      0239 444 ; QUEUE, IF THE COLLISION BIT IS SET.
      0239 445 ;
      0239 446 20$:
      0239 447 PAGIO_DONE1:
      52 00 9A 0239 448      MOVZBL #PRI$_NULL,R2      ; SET FOR NULL PRIORITY INCREMENT
      023C 449      RPTEVT PFCOM      ; REPORT PAGE FAULT COMPLETE
      0240 450 ;
      0240 451 ; IRP$W_BOFF WAS INCREMENTED IF ANY OF THE PAGES HAD THE COLLISION BIT SET
      0240 452 ;
      0240 453 ; R7 = NON ZERO IF SUPPOSED TO ISSUE KERNEL AST
      0240 454 ; USED ONLY FOR PAGE WRITE COMPLETION
      0240 455 ; BUT MUST BE ZERO FOR PAGE READ COMPLETION
      0240 456 ;
      0240 457 PAGIO_DONE2:
      30 A6 B5 0240 458      TSTW IRP$W_BOFF(R6)      ; ANY PAGES WITH COLLISION BIT SET?
      15 13 0243 459      BEQL 60$          ; BRANCH IF NOT
      54 DD 0245 460      PUSHL R4      ; SAVE PCB ADDRESS
      0008'CF B5 0247 461 40$: TSTW W^SCH$GQ_COLPGWQ+WQH$W_WQCNT ; ANYONE WAITING?
      0B 15 024B 462      BLEQ 50$          ; BRANCH IF NOT
      54 0000'CF D0 024D 463      MOVL W^SCH$GQ_COLPGWQ,R4      ; GET NEXT PCB
      EF 11 0256 464      RPTEVT COLPGA      ; REPORT "COLLISION PAGE AVAILABLE"
      10 BA 0258 465      BRB 40$          ; REPEAT UNTIL QUEUE IS EMPTY
      57 D5 025D 466 50$: POPR #^M<R4>      ; RESTORE SAVED PCB ADDRESS
      16 12 025F 467 60$: SETIPL #IPL$_IOPOST      ; LOWER TO I/O POST LEVEL
      7E 38 A6 E9 0261 468      TSTL R7      ; EXHAUSTED PAGE COUNT NON-ZERO?
      50 56 D0 0265 469      BNEQ PAGIO_KAST      ; BRANCH IF YES, COMPLETE I/O IN PROCESS
      56 8E 7D 0266 470      BLBC IRP$L_IOST1(R6),PAGIO_ERR ; BRANCH IF MORE ERROR PROCESSING TO DO
      026B 471      MOVL R6,R0      ; GET PACKET ADDRESS FOR RELEASE
      026B 472      MOVQ (SP)+,R6      ; RESTORE SAVED REGISTERS
      026B 473 ;
      026B 474 ; RO = I/O REQUEST PACKET ADDRESS
      026B 475 ;
      026B 476 PAGIO_ERR_DONE:
      FD92' 30 026B 477      BSBW EXE$DEANONPAGED      ; AND RELEASE IT
      50 01 3C 026E 478      MOVZWL #RSN$_ASTWAIT,R0      ; SET AST WAIT RESOURCE WAIT NUMBER
      FD8C' 30 0271 479      BSBW SCH$RAVAIL      ; SET RESOURCE AVAILABLE
```

```

FD96 31 0274 480 BRW IOPOST ; CONTINUE TO PROCESS POST QUEUE
      0277 481 ;
      0277 482 ; COMPLETE THE PAGE WRITE IN THE PROCESS CONTEXT
      0277 483 ;
      0277 484 PAGIO_KAST:
55 56 D0 0277 485 MOVL R6,R5 ; I/O PACKET ADDRESS BACK TO NORMAL REG
56 8E 7D 027A 486 MOVQ (SP)+,R6 ; RESTORE SAVED REGISTERS
51 0C A5 D0 027D 487 MOVL IRP$L_PID(R5),R1 ; PROCESS ID FOR ISSUING KERNEL AST
00000000'EF 9E 0281 488 MOVAB MMG$UPDSECAST,ACB$L_KAST(R5) ; ADDRESS TO START KERNEL AST
      0287
52 01 9A 0289 489 MOVZBL #PRI$I_OCOM,R2 ; PRIORITY INCREMENT
OB A5 80 8F 88 028C .1 BISB #^X80,ACB$B_RMOD(R5) ; SET INTERNAL AST FLAG
      FD6C' 30 0291 .2 BSBW SCH$QAST ; NOW QUEUE THE KERNEL AST
      FD76 31 0294 .3 BRW IOPOST ; GET NEXT PACKET TO POST
      0297 491 ;
      0297 492 ; PAGE READ ERROR - CLEAN UP LOGIC
      0297 493 ;
      0297 494 ; R3 = PTE ADDRESS OF BAD PAGE
      0297 495 ; R4 = PCB ADDRESS
      0297 496 ; R5 = PROCESS HEADER ADDRESS
      0297 497 ; R6 = I/O REQUEST PACKET ADDRESS
      0297 498 ; R7 = 0 AND MUST BE PRESERVED
      0297 499 ; IRP$L_AST(R6) = PROCESS PTE ADR OF BAD PAGE IF GLOBAL PAGE
      0297 500 ; IRP$L_ASTPRM(R6) = GPTX FOR START OF TRANSFER IF GLOBAL CRF
      0297 501 ;
      0297 502 PAGRD_ERR:
00E1 30 0297 503 BSBW PFN_IO_DONE ; COMPLETE THE I/O FOR ERR PAGE
      14 90 029A 504 MOVVB #<PFN$M_DELCON ! PFN$C_RDERR>,- ; SET PAGE TO
0000'DF40 029C 505 @W^PFN$AB_STATE[R0] ; READ ERROR STATE
51 14 A6 D0 02A0 506 MOVL IRP$L_ASTPRM(R6),R1 ; GET BACKING STORE ADR IF CRF
      18 13 02A4 507 BEQL 120$ ; BRANCH IF NOT COPY ON REFERENCE
0E 51 17 E1 02A6 508 BBC #PFN$V_GBLBAK,R1,100$ ; BRANCH IF NOT GLOBAL CRF
      09 EF 02AA 509 EXTZV #VAV$VPN,-
      17 02AC 510 #<32-VAV$VPN>,- ; ADJUST GPTX BY
52 3A A6 02AD 511 IRP$L_IOST1+2(R6),R2 ; TRANSFERRED PAGE COUNT
      51 52 C0 02B0 512 ADDL R2,R1 ; TO GET CORRECT GPTX FOR BAD PAGE
      51 01 C1 02B3 513 ADDL3 #1,R1,IRP$L_ASTPRM(R6) ; SET GPTX FOR START OF NEXT TRANSFER
      14 A6 02B6
0000'DF40 51 D0 02B8 514 100$: MOVL R1,@W^PFN$AL_BAK[R0] ; FIX BACKING STORE ADDRESS
      10 A6 D5 02BE 515 120$: TSTL IRP$L_AST(R6) ; IF GLOBAL PAGE (NOT CRF)
      04 18 02C1 516 BGEQ 140$
10 A6 04 C0 02C3 517 ADDL #4,IRP$L_AST(R6) ; THEN SKIP OVER PROCESS PTE ADR
0000'DF40 B5 02C7 518 140$: TSTW @W^PFN$AW_REFCNT[R0] ; IS THIS THE LAST REFERENCE?
      12 14 02CC 519 BGTR 160$ ; BRANCH IF NOT
0000'DF40 B5 02CE 520 TSTW @W^PFN$AW_SWPVBN[R0] ; IF THIS PROCESS HAS BEEN SWAPPED OUT
      08 13 02D3 521 BEQL 150$
52 02 9A 02D5 522 MOVZBL #PFN$C_BADPAGLST,R2 ; THEN PUT THIS PAGE IN LIMBO
      FD25' 30 02D8 523 BSBW MMG$INSPFNT ; ON THE BAD PAGE LIST
      03 11 02DB 524 BRB 160$
      FD20' 30 02DD 525 150$: BSBW MMG$RELPFN ; OTHERWISE RELEASE THE PAGE
      FF35 31 02E0 526 160$: BRW PAGIO_DONE ; COMPLETE THIS PORTION OF THE PAGE READ
      02E3 527 ;
      02E3 528 ; DO THE REMAINING SEGMENT OF THE I/O FOR A PAGE READ OR WRITE ERROR
      02E3 529 ; SKIP OVER THE PORTION THAT WAS TRANSFERRED SUCCESSFULLY AND SKIP OVER
      02E3 530 ; THE PAGE IN ERROR WHICH WAS DEALT WITH BY EITHER PAGRD_ERR OR
      02E3 531 ; PAGWRT_ERR AND SET UP TO TRANSFER THE REMAINING PAGES IF ANY.
      02E3 532 ; NOTE THAT FOR PAGE WRITE ERRORS THE REST OF THE TRANSFER IS NOT DONE

```

;WMC0020  
;WMC0020  
;WMC0020  
-1

```

02E3 533 ; IF I/O COMPLETION STATUS IS RETURNED TO THE PROCESS.
02E3 534 ;
02E3 535 PAGIO_ERR:
55 56 D0 02E3 536 MOVL R6,R5 ; IRP ADDRESS
56 8E 7D 02E6 537 MOVQ (SP)+,R6 ; RESTORE ADDITIONAL SAVED REGISTERS
09 EF 02E9 538 EXTZV #VA$V_VPN,-
17 02EB 539 #<32-VA$V_VPN>,- ; GET PAGE COUNT TRANSFERRED
51 3A A5 02EC 540 IRP$L_IOST1+2(R5),R1
51 D6 02EF 541 INCL R1 ; COUNT THE ERROR PAGE AS DONE
50 51 09 9C 02F1 542 ROTL #9,R1,R0 ; BYTE COUNT COMPLETED
44 A5 50 C2 02F5 543 SUBL R0,IRP$L_OBCNT(R5) ; BYTE COUNT REMAINING
25 13 02F9 544 BEQL 40$ ; BRANCH IF NOTHING LEFT TO DO
40 A5 D4 02FB 545 CLRL IRP$L_ABCNT(R5) ; ZERO ACCUMULATED BYTE COUNT
02FE 546
30 A5 B4 02FE 547 CLRW IRP$W_BOFF(R5) ; ZERO BOFF AND
44 A5 D0 0301 548 MOVL IRP$L_OBCNT(R5)
32 A5 0304 549 IRP$L_BCNT(R5) ; SET NEW BYTE COUNT
48 A5 D6 0306 550 INCL IRP$L_SEGVBN(R5) ; SEGMENT VBN WAS POINTING AT ERROR VBN
53 4C A5 D0 0309 551 MOVL IRP$L_DIAGBUF(R5),R3 ; STARTING SVAPTE OF ENTIRE TRANSFER
4C A5 6341 DE 030D 552 MOVAL (R3)[R1],IRP$L_DIAGBUF(R5) ; STARTING PTE ADDRESS OF THIS SEGMENT
0312 553
0312 554 .IF DF,CA$_MEASURE_IOT
53 DD 0312 555 PUSHL R3 ;REMEMBER SVAPTE
53 55 DO 0314 556 MOVL R5,R3 ;SET ADR OF IRP
FCE6' 30 0317 557 BSBW PMS$START_RQ ; INSERT START OF I/O REQUEST MESSAGE
53 BED0 031A 558 POPL R3 ;RESTORE SVAPTE
031D 559 .ENDC
031D 560
00EB 31 031D 561 BRW QNXTSEG ; QUEUE THIS SEGMENT AND RETURN TO IOPOST
50 55 D0 0320 562 40$: MOVL R5,R0 ; I/O PACKET ADDRESS
FF45 31 0323 563 BRW PAGIO_ERR_DONE
0326 564 ;
0326 565 ; PAGE WRITE ERROR - CLEAN UP LOGIC
0326 566 ;
0326 567 ; R3 = PTE ADDRESS FOR ERROR PAGE
0326 568 ; R4 = PCB ADDRESS
0326 569 ; R5 = PROCESS HEADER ADDRESS
0326 570 ; R6 = I/O REQUEST PACKET ADDRESS
0326 571 ; R7 = 0 IF ALL COMPLETION LOGIC IS DONE IN IOPOST
0326 572 ; = NON-ZERO IF COMPLETION (AND ERROR REPORT) ARE TO BE
0326 573 ; RETURNED TO THE PROCESS.
0326 574 ;
0326 575 PAGWRT_ERR1:
57 DD 0326 576 PUSHL R7 ; SAVE KERNEL AST FLAG
09 EF 0328 577 EXTZV #VA$V_VPN,-
17 032A 578 #<32-VA$V_VPN>,- ; PAGE COUNT TRANSFERRED
50 3A A6 032B 579 IRP$L_IOST1+2(R6),R0
09 EF 032E 580 #VA$V_VPN,-
17 0330 581 #<32-VA$V_VPN>,- ; ORIGINAL PAGE COUNT
57 44 A6 0331 582 IRP$L_OBCNT(R6),R7
57 50 C2 0334 583 SUBL R0,R7 ; COUNT OF REMAINING PAGES
6E D5 0337 584 TSTL (SP) ; IF NOT REPORTING ERROR TO PROCESS
03 12 0339 585 BNEQ 20$
57 01 D0 033B 586 MOVL #1,R7 ; ONLY CLEAN UP THE ERROR PAGE HERE
033E 587 ; REST OF TRANSFER WILL BE DONE BY PAGIO_ERR
7E D4 033E 588 20$: CLRL -(SP) ; INIT "ERROR PAGE" FLAG
63 15 00 EF 0340 589 EXTZV #PTE$V_PFN,#PTE$$_PFN,(R3),R0 ;GET PFN FROM PTE

```

```

50          0344
50 D1 0345 590          CMPL  R0,MMG$GL_MAXPFN          ;IS THIS PAGE IN SHARED MEMORY?
00000000'EF 0347
25 1A 034C 591          BGTRU 130$                  ;BR IF PAGE IN SH MEM, NO PFN DATABASE
53 DD 034E 592 70$:    PUSHL  R3                      ; SAVE SVAPTE
0028 30 0350 593          BSBW  PFN_IO_DONE                ; COMPLETE I/O FOR THIS PAGE
0000'DF40 07 E2 0353 594          BBSS  #PFN$V_MODIFY,@W^PFN$AB_STATE[R0],80$ ; FORCE MODIFY BIT
00 0359
035A 595 80$:
035A 596 ;
035A 597 ; CONDITION CODES STILL SET FROM DECREF AT END OF PFN_IO_DONE
035A 598 ;
10 14 035A 599          BGTR  120$                  ; BRANCH IF NOT THE LAST REFERENCE
08 04 AE 00 E2 035C 600          BBSS  #0,4(SP),100$                ; BRANCH IF NOT ERROR PAGE
0361 601 ;
0361 602 ; THIS IS THE PAGE THAT HAD THE WRITE ERROR
0361 603 ;
52 02 D0 0361 604          MOVL  #PFN$C_BADPAGLST,R2          ; PUT IT ON THE BAD PAGE LIST
FC99' 30 0364 605          BSBW  MMG$INSPFNT                ; WITH "MODIFY" SET AND "BAD" CLEAR
03 11 0367 606          BRB   120$
FC94' 30 0369 607 100$:    BSBW  MMG$RELPFN                ; NO MORE REFERENCES, RELEASE THE PAGE
53 8E 04 C1 036C 608 120$:  ADDL3  #4,(SP)+,R3          ; NEXT PTE ADDRESS
DB 57 F5 0370 609          SOBGTR R7,70$
03 BA 0373 610 130$:    POPR   #^M<R0,R1>                ; CLEAN OFF BAD PAGE FLAG
0375 611 ;
57 51 D0 0375 612          MOVL  R1,R7                      ; PUT IT IN R7, SET CONDITION CODES
FE54 31 0378 613          BRW   PAGWRT_ERR_DONE
037B 614 ;
037B 615 ; PFN_IO_DONE
037B 616 ;
037B 617 ; INPUTS:
037B 618 ;
037B 619 ; R3 = SVAPTE
037B 620 ; R4 = PROCESS CONTROL BLOCK ADDRESS OF PROCESS THAT REQUESTED THE I/O
037B 621 ; R5 = PROCESS HEADER OF THE PROCESS THAT REQUESTED THIS I/O
037B 622 ; R6 = I/O REQUEST PACKET ADDRESS
037B 623 ;
037B 624 ; OUTPUTS:
037B 625 ;
037B 626 ; R0 = PFN
037B 627 ; R3 PRESERVED
037B 628 ; IRP$W_BOFF(R6) INCREMENTED IF THIS WAS A COLLISION PAGE
037B 629 ; CONDITION CODES SET FROM DECW @W^PFN$AW_REFCNT[R0]
037B 630 ;
037B 631 PFN_IO_DONE:
63 15 00 EF 037B 632          EXTZV  #PTE$V_PFN,#PTE$S_PFN,(R3),R0 ; GET PAGE FRAME NUMBER
50 037F
E8 8F 8B 0380 633          BICB3  #^C<PFN$M_COLLISION ! PFN$M_PAGTYP>,- ; FETCH THESE
51 0000'DF40 0383 634          @W^PFN$AB_TYPE[R0],R1 ; BITS FROM PFN TYPE BYTE
09 51 04 E5 0388 635          BBCC  #PFN$V_COLLISION,R1,20$ ; CLEAR COLLISION BIT, BRANCH IF WAS CLEAR
0000'DF40 10 8A 038C 636          BICB  #PFN$M_COLLISION,@W^PFN$AB_TYPE[R0] ; CLEAR IT IN PFN DATA
30 A6 B6 0392 637          INCW  IRP$W_BOFF(R6) ; MUST EMPTY THE COLLISION QUEUE
04 51 91 0395 638 20$:    CMPB  R1,#PFN$C_PPGTBL ; IF PROCESS PAGE TABLE PAGE
07 12 0398 639          BNEQ  40$
51 42 A5 3C 039A 640          MOVZWL PHD$W_PHVINDEX(R5),R1 ; MUST COUNT ONE LESS
FC5F' 30 039E 641          BSBW  MMG$DECPHDREF1 ; PROCESS HEADER REFERENCE
03A1 642 40$:    DECREF ; ONE LESS REFERENCE FOR THE PAGE

```

IOCIOPST  
V03-024

- I/O COMPLETION POSTING  
PAGIO - PAGE I/O COMPLETION

3-JUN-1984 11:26:47 VAX-11 Macro V03-01 Page 16  
24-APR-1982 15:46:59 DISK\$VMSMASTER:[SYS.SRC]IOCIOPST.(4)

05 03AB 643 RSB  
03AC 644

; RETURN WITH CONDITION CODES SET  
; TO NEW STATE OF THE REFCNT

```

;RLRMXBCNT      03AC      .2      .SBTTL VIRTUAL (OR LOGICAL) I/O COMPLETION
;RLRMXBCNT      03AC      .3      ;
;RLRMXBCNT      03AC      .4      ; VIRTUAL (OR LOGICAL) I/O COMPLETION
-6              03AC      651     ;
              03AC      652     ; CALLING SEQUENCE:
              03AC      653     ;
              03AC      654     ; BRW      VIRTUAL
              03AC      655     ;
              03AC      656     ; INPUTS:
              03AC      657     ;
              03AC      658     ; R1 = REQUESTED BYTE COUNT, POSSIBLY DIFFERENT FROM TRANSFERRED
              03AC      659     ; BYTE COUNT FOR MAGTAPE
              03AC      660     ; R2 = IRP$W_BOFF CONTENTS
              03AC      661     ; R3 = SVAPTE OF START OF TRANSFER
;ROW49597       03AC      .1      ; R4 = PCB ADDRESS ASSOCIATED WITH THE PID IN THE PACKET
;RLRMXBCNT      03AC      .2      ; R5 = IRP ADDRESS
              03AC      662     ;
              03AC      663     ; OUTPUTS:
              03AC      664     ;
              03AC      665     ; BRANCHES TO UNLOCK, PRESERVING R1,R2,R3
              03AC      666     ; OR BRANCHES TO IOPOST
              03AC      667     ;
              03AC      668     ;
              03AC      669     ; .ENABL  LSB
              03AC      670     ;
;RLRMXBCNT      03AC      .1      VIRTUAL_LOGIO: ; VIRTUAL (OR LOGICAL) I/O FUNCTION
;RLRMXBCNT      46 A5  B5  03AC      .2      TSTW  IRP$_OBCNT+2(R5) ; SEE IF BYTE COUNT > 64K
;RLRMXBCNT      OF  13  03AF      .3      BEQL  1$ ; EQL IMPLIES NO, BRANCH TO OLD CODE
;RLRMXBCNT      03B1      .4      ;
;RLRMXBCNT      50  3A  A5  D0  03B1      .5      MOVL  IRP$_IOST1+2(R5), R0 ; Else pickup new, longer count.
;RLRMXBCNT      40 A5  50  C0  03B5      .6      ADDL  R0, IRP$_ABCNT(R5) ; Accumulate total bytes transferred.
;RLRMXBCNT      3A A5  40 A5  D0  03B9      .7      MOVL  IRP$_ABCNT(R5), - ; Set accumulated bytes transferred.
;RLRMXBCNT      03BE      .8      ;
;RLRMXBCNT      OD  11  03BE      .9      BRB   3$ ; Rejoin common code.
;RLRMXBCNT      03C0      .10     ;
;RLRMXBCNT      50  3A  A5  3C  03C0      .11  1$:  MOVZWL IRP$_IOST1+2(R5), R0 ; Get old bytes transferred count.
;RLRMXBCNT      40 A5  50  C0  03C4      .12     ADDL  R0, IRP$_ABCNT(R5) ; Accumulate total bytes transferred.
;RLRMXBCNT      3A A5  40 A5  B0  03C8      .13     MOVW  IRP$_ABCNT(R5), - ; Set accumulated bytes transferred.
;RLRMXBCNT      03CD      .14     ;
;RLRMXBCNT      03CD      .15     ; (Note movw due to code path that
;RLRMXBCNT      03CD      .16     ; insures < 64K byte transfer.)
;RLRMXBCNT      50  50  DD  03CD      .17  3$:  PUSHL R0 ; Save # bytes transferred.
;RLRMXBCNT      51  50  D1  03CF      .18     CMPL  R0, R1 ; Do bytes xfered and requested match?
;RLRMXBCNT      13  13  03D2      .19     BEQL  9$ ; Branch if they match.
;RLRMXBCNT      50  1C  A5  D0  03D4      .20     MOVL  IRP$_UCB(R5), R0 ; R0 => UCB.
;RLRMXBCNT      05  E0  03D8      .21     BBS   S^#DEV$V_SQD, -
;RLRMXBCNT      0A 38  A0  03DA      .22     UCB$_DEVCHAR(R0), 9$ ; If SET, sequential device
;RLRMXBCNT      06 38  A5  E9  03DD      .23     BLBC  IRP$_IOST1(R5), 9$ ; If xfer count wrong, guarantee
;RLRMXBCNT      2234 8F  B0  03E1      .24     MOVW  #SS$_INCSEGTRA, - ; that final status is an error
;RLRMXBCNT      38  A5  03E5      .25     ;
;RLRMXBCNT      03E7      .26     ;
;RLRMXBCNT      8E  F7  8F  78  03E7      .26  9$:  ASHL  #-VA$_$_BYTE, (SP)+, R0 ; Calculate number of blocks transferred.
;RLRMXBCNT      50  03EB      .27     ;
;RLRMXBCNT      48 A5  50  C0  03EC      .27     ADDL  R0, IRP$_SEGVBN(R5) ; Calculate next disk segment address.
-20           4C 38  A5  E9  03F0      691     BLBC  IRP$_IOST1(R5), 20$ ; IF LBC I/O ERROR
              50  1C  A5  D0  03F4      692     MOVL  IRP$_UCB(R5), R0 ; GET ADDRESS OF DEVICE UCB
              2E 38  A0  05  E0  03F8      693     BBS   S^#DEV$V_SQD, UCB$_DEVCHAR(R0), 10$ ; IF SET, SEQUENTIAL DEVICE

```

```

40 A5 C3 03FD 694      SUBL3  IRP$L_ABCNT(R5),-
44 A5      0400 695      IRP$L_OBCNT(R5),-      ; CALCULATE BYTES REMAINING
32 A5      0402 696      IRP$L_BCNT(R5)        ;
25      13 0404 697      BEQL   10$              ; IF EQL NONE
51 F7 8F 78 0406 698      ASHL   #-VASS_BYTE,R1,R1 ; CALCULATE NUMBER OF PAGES REQUESTED
51      51      040A
      040B 699 QNXTSEG:
      040B .1 ;
;STJ3100      040B .2 ; ADVANCE THE SVAPTE TO POINT TO THE PORTION OF THE PAGE TABLES THAT MAP THE
;STJ3100      040B .3 ; BUFFER FOR THIS SEGMENT. IF THIS IS AN ERASE I/O, DO NOT ADVANCE THE
;STJ3100      040B .4 ; SVAPTE, AS THE ENTIRE TRANSFER IS MAPPED BY A SINGLE PAGE TABLE PAGE.
;STJ3100      040B .5 ;
;STJ3100      0000 CF D6 040B .6      INCL   W^PMS$GL_SPLIT      ; COUNT A SPLIT TRANSFER
;STJ3100      0A      E1 040F .7      BBC    #IO$V_ERASE,-      ; BRANCH IF NOT ERASE - UPDATE SVAPTE
;STJ3100      06 20 A5 0411 .8      IRP$W_FUNC(R5),13$    ;
;STJ3100      02F9 30 0414 .9      BSBW   CHECK_ERASE      ; IS THIS AN ERASE I/O REQUEST
;STJ3100      05 50  E8 0417 .10     BLBS   R0,69$          ; BRANCH IF YES - DO NOT ADVANCE SVAPTE
;STJ3100      2C A5 6341 DE 041A .11 13$:  MOVAL  (R3)[R1],IRP$L_SVAPTE(R5) ; SET ADDRESS OF NEXT PTE ENTRY
;STJ3100      53 55  D0 041F .12 69$:  MOVL   R5,R3          ; COPY I/O REQUEST PACKET ADDRESS
;STJ3100      55 1C A3 D0 0422 .13     MOVL  IRP$L_UCB(R3),R5   ; COPY UCB ADDRESS
-4      47      10 0426 704      BSBW   IOC$QNXTSEG      ; QUEUE THE NEXT VIRTUAL SEGMENT
      FBE2 31 0428 705 5$:  BRW    IOPOST
      042B 706 ;
      042B 707 ; ALL SEGMENTS OF THIS TRANSFER ARE COMPLETE
      042B 708 ;
      042B 709 10$:
51 44 A5 D0 042B 710      MOVL  IRP$L_OBCNT(R5),R1      ; GET ORIGINAL BYTE COUNT
53 4C A5 D0 042F 711      MOVL  IRP$L_DIAGBUF(R5),R3    ; GET ORIGINAL PAGE TABLE ADDRESS
;STJ3085      04      12 0433 .1      BNEQ   15$              ; NEQ implies IRP$L_DIAGBUF was valid.
;STJ3085      53 2C A5 D0 0435 .2      MOVL  IRP$L_SVAPTE(R5),R3    ; If not valid, then IRP$L_SVAPTE is.
;STJ3085      2C A5 53 D0 0439 .3 15$:  MOVL  R3,IRP$L_SVAPTE(R5) ; SVAPTE MUST BE CORRECT
;STJ3085      FC51 31 043D .4      BRW    UNLOCK
;STJ3085      0440 .5
;STJ3085      0440 .6 ;
;STJ3085      0440 .7 ; I/O OPERATION ENDED WITH AN UNSUCCESSFUL STATUS
;STJ3085      0440 .8 ;
;STJ3085      0440 .9 ; IF THE REQUEST IS LOGICAL I/O, BRANCH BACK TO UNLOCK. (10$)
-4      0440 716 ;
      0440 717 ; IF THE DEVICE IS A SEQUENTIAL DEVICE, THEN THE I/O PACKET IS
      0440 718 ; MERELY SENT TO THE ACP FOR NOTIFICATION OF THE ERROR.
      0440 719 ;
      0440 720 ; IF THE DEVICE IS A RANDOM DEVICE, THEN THE VIRTUAL BLOCK NUMBER
      0440 721 ; STORED IN IRP$L_SEGVBN IS THE BLOCK THAT HAS AN ERROR.
      0440 722 ;
      0440 723 ;
;RLRMXBCNTA      04      E1 0440 .1 20$:  BBC    #IRP$V_VIRTUAL,-
;RLRMXBCNTA      E6 2A A5 0442 .2      IRP$W_STS(R5),10$    ; Branch IF Logical I/O
;RLRMXBCNTA      46 A5 B5 0445 .3      TSTW   IRP$L_OBCNT+2(R5)      ; SEE IF BYTE COUNT > 64K
;RLRMXBCNTA      05      13 0448 .4      BEQL   30$              ; EQL implies < 64K.
;RLRMXBCNTA      3A A5 D4 044A .5      CLRL  IRP$L_IOST1+2(R5)    ; Zero byte count before recycling IRP
;RLRMXBCNTA      03      11 044D .6      BRB    40$              ; Branch around
;RLRMXBCNTA      044F .7 30$:
;RLRMXBCNTA      3A A5 B4 044F .8      CLRW  IRP$L_IOST1+2(R5)    ; Zero byte count before recycling IRP
;RLRMXBCNTA      0452 .9 40$:
;RLRMXBCNTA      53 55 D0 0452 .10     MOVL  R5,R3          ; COPY IRP ADDRESS
;ACG0421      3E A4 B7 0455 .11     DECW  PCB$W_DIOCNT(R4) ; ADJUST DIRECT I/O COUNT
;ACG0421      2A A3 10 AA 0458 .12     BICW  #IRP$M_VIRTUAL,IRP$W_STS(R3) ; CLEAR VIRTUAL I/O FLAG

```



IOCIOPST  
V03-024

- I/O COMPLETION POSTING  
VIRTUAL (OR LOGICAL) I/O COMPLETION

3-JUN-1984 11:26:47 VAX-11 Macro V03-01 Page 19  
24-APR-1982 15:46:59 DISK\$VMSMASTER:[SYS.SRC]IOCIOPST.(5)

;ACG0421	2C A3	4C A3	D0	045C	.13	MOVL	IRP\$L_DIAGBUF(R3),IRP\$L_SVAPTE(R3)	; RESET PAGE TABLE ADDRESS
;ACG0421	52	44 A3	D0	0461	.14	MOVL	IRP\$L_OBCNT(R3),R2	; GET ORIGINAL BYTE COUNT
;ACG0421		009F	30	0465	.15	BSBW	IOC\$QTOACP	; QUEUE PACKET TO ACP
-6		BE	11	0468	730	BRB	5\$	
				046A	731			
				046A	732	.DSABL	LSB	

```

046A 734 .SBTTL QUEUE NEXT SEGMENT
046A 735 ;
046A 736 ; FUNCTIONAL DESCRIPTION:
046A 737 ;
046A 738 ; IOC$QNXTSEG PERFORMS THE FUNCTION OF QUEUEING THE NEXT
046A 739 ; SEGMENT OF A VIRTUAL I/O REQUEST THAT DID NOT MAP TO A
046A 740 ; SINGLE CONTIGUOUS I/O REQUEST.
046A 741 ;
046A 742 ; CALLING SEQUENCE:
046A 743 ;
046A 744 ; BSBW IOC$QNXTSEG
046A 745 ;
046A 746 ; INPUTS:
046A 747 ;
046A 748 ; R3 = I/O REQUEST PACKET ADDRESS
046A 749 ; R4 = PCB ADDRESS ASSOCIATED WITH THE PID IN THE PACKET
046A 750 ; R5 = UCB ADDRESS OF THE ASSOCIATED DEVICE
046A 751 ;
046A 752 ; OUTPUTS:
046A 753 ;
046A 754 ; R4 NOT PRESERVED
046A 755 ;
;ACG0421 046A .1
;ACG0421 046A .2 .ENABLE LSB
;ACG0421 046A .3 ; Out of line code for Logical I/O.
;ACG0421 046A .4 ; This code mimics results of
;ACG0421 046A .5 ; IOC$MAPVBLK for Logical I/O.
;ACG0421 51 50 D0 046A .6 5$: MOVL R0,R1 ; Namely R1 = LBN.
;ACG0421 24 11 046D .7 BRB 10$ ; Branch back to common code.
;ACG0421 046F .8
046F 756 IOC$QNXTSEG::
52 18 A3 D0 046F 757 MOVL IRP$L_WIND(R3),R2 ; GET ADDRESS OF MAPPING WINDOW
51 32 A3 D0 0473 758 MOVL IRP$L_BCNT(R3),R1 ; GET SIZE OF NEXT SEGMENT
50 48 A3 D0 0477 759 MOVL IRP$L_SEGVBN(R3),R0 ; GET STARTING VIRTUAL BLOCK NUMBER
047B 760 ;
047B 761 ; ALTERNATE ENTRY TO IOC$QNXTSEG:
047B 762 ;
047B 763 ; BSBW IOC$QNXTSEG1
047B 764 ;
047B 765 ; ADDITIONAL INPUTS:
047B 766 ;
047B 767 ; R0 = VIRTUAL BLOCK NUMBER OF START OF NEXT SEGMENT
047B 768 ; R1 = DESIRED BYTE COUNT OF NEXT SEGMENT
047B 769 ; R2 = WINDOW ADDRESS
047B 770 ;
047B 771 IOC$QNXTSEG1::
3E A4 B7 047B 772 DECW PCB$W_DIOCNT(R4) ; ADJUST THE DIRECT I/O COUNT
;ADE9005 04 04 047E .1 BBC #IRP$V_VIRTUAL,- ; Branch to out of line code if this
;ADE9005 E7 2A A3 0480 .2 IRP$W_STS(R3),5$ ; is Logical I/O.
;ADE9005 00000000 GF 16 0483 .3 JSB G^IOC$MAPVBLK ; MAP VIRTUAL TO LOGICAL BLOCK
;ACG0422 1C A3 55 D0 0489 .4 MOVL R5,IRP$U_UCB(R3) ; STORE POSSIBLY MODIFIED UCB ADDRESS
;ACG0422 32 A3 52 C2 048D .5 SUBL R2,IRP$L_BCNT(R3) ; CALCULATE SIZE OF NEXT SEGMENT
;ACG0422 74 13 0491 .6 BEQL 30$ ; IF EQL TOTAL MAP FAILURE
;ACG0422 52 00B4 C5 D0 0493 .7 10$: MOVL UCB$L_MAXBCNT(R5),R2 ; R2 = 0 or Max. permissible BCNT.
;ACG0422 05 12 0498 .8 BNEQ 15$ ; NEQ implies Max. permissible BCNT in R0.
;ACG0422 52 FE00 8F 3C 049A .9 MOVZWL #512*127,R2 ; If 0, use default Max. permissible.
;ACG0422 0A E1 049F .10 15$: BBC #10$V_ERASE,- ; BRANCH IF DEFINITELY NOT AN ERASE

```

```

;ACG0422      20 20 A3      04A1  .11      IRP$_FUNC(R3),17$      ;
;ACG0422      55      DD 04A4  .12      PUSHL  R5      ; SAVE UCB ADDRESS
;ACG0422      55 53 D0 04A6  .13      MOVL   R3,R5      ; COPY IRP ADDRESS
;ACG0422      0264 30 04A9  .14      BSBW  CHECK_ERASE      ; IS THIS AN ERASE FUNCTION?
;ACG0422      55 8ED0 04AC  .15      POPL  R5      ; RESTORE UCB ADDRESS
;ACG0422      12 50 E9 04AF  .16      BLBC  R0,17$      ; BRANCH IF IT IS NOT AN ERASE
;ACG0422      2C A3 D5 04B2  .17      TSTL  IRP$_SVAPTE(R3)      ; ARE WE USING A DUMMY PAGE TABLE?
;ACG0422      0D 13 04B5  .18      BEQL  17$      ; BRANCH IF NOT
;ACG0422      50 FE00 8F 3C 04B7  .19      MOVZWL #512*127,R0      ; GET MAX BYTE COUNT FOR PPT
;ACG0422      50 52 D1 04BC  .20      CMLP  R2,R0      ; CHECK LIMIT AGAINST MAX
;ACG0422      03 1B 04BF  .21      BLEQU 17$      ; BRANCH IF OK
;ACG0422      52 50 D0 04C1  .22      MOVL  R0,R2      ; LIMIT TRANSFER TO PPT SIZE
;ACG0422      04C4  .23
;ACG0422      32 A3 52 D1 04C4  .24 17$: CMLP  R2,IRP$_BCNT(R3)      ; See if BCNT too large.
;ROW0218      04 1E 04C8  .25      BGEQU 20$      ; GEQU implies we are OK.
;RLRMXBCNT    32 A3 52 D0 04CA  .26      MOVL  R2,IRP$_BCNT(R3)      ; Else scale down to maximum allowed.
;RLRMXBCNT    04CE  .27 20$:
-5           52 32 A3 D0 04CE  778      MOVL  IRP$_BCNT(R3),R2      ; GET TRANSFER BYTE COUNT
           52 D7 04D2  779      DECL  R2      ; ROUND DOWN AND...
           52 F7 8F 78 04D4  780      ASHL  #-VA$$_BYTE,R2,R2      ; SHIFT DOWN FOR BLOCK COUNT - 1
           52 04D8
           52 51 C0 04D9  781      ADDL  R1,R2      ; COMPUTE ENDING BLOCK NUMBER
           13 1F 04DC  782      BCS  25$      ; BRANCH ON OVERFLOW
00B0 C5 52 D1 04DE  783      CMLP  R2,UCB$_MAXBLOCK(R5)      ; AND CHECK AGAINST DEVICE SIZE
           0C 1E 04E3  784      BGEQU 25$      ; BRANCH IF NOT LEGAL
;ACG0421      50 51 D0 04E5  .1      MOVL  R1,R0      ; COPY STARTING LOGICAL BLOCK NUMBER
;ADE9005      00000000'GF 16 04E8  .2      JSB  G^IOC$CVTLOGPHY      ; CONVERT LOGICAL TO PHYSICAL BLOCK
-1           FB0F' 31 04EE  786      BRW  EXE$INSIOQ      ; INSERT I/O PACKET IN DEVICE QUEUE
           04F1  787      ; AND RETURN
           04F1  788 ;
           04F1  789 ; TO HERE IF THE VIRTUAL BLOCKS MAP OFF THE END OF THE VOLUME. COMPLETE THE
           04F1  790 ; I/O WITH AN ERROR. WE QUEUE THE PACKET FOR PROCESSING, RATHER THAN WANDERING
           04F1  791 ; OFF INTO THE COMPLETION CODE BECAUSE THIS IS A GENERALLY CALLABLE ROUTINE.
           04F1  792 ;
           00DC 8F 3C 04F1  793 25$: MOVZWL #SS$_ILLBLKNUM,IRP$_IOST1(R3) ; SET ILLEGAL BLOCK NUMBER STATUS
           38 A3 04F5
           3C A3 D4 04F7  794      CLRL  IRP$_IOST2(R3)      ; ZERO 2ND I/O STATUS LONGWORD
           63 0E 04FA  795      INSQUE (R3),@IOC$GL_PSBL      ; INSERT AT TAIL OF I/O POST QUEUE
00000000'FF 04FC
           03 12 0501  796      BNEQ  26$      ; BRANCH IF NOT EMPTY
           0503  797      SOFTINT #IPL$_IOPOST      ; WAKE UP I/O COMPLETION
           0506  798 26$: RSB
           0507  799
           0507 800 30$:
;ACG0421      0507 .1      .DISABLE LSB
           0507 801 ;
           0507 802 ; ALTERNATE ENTRY TO IOC$WAKACP:
           0507 803 ;
           0507 804 ; BSBW IOC$QTOACP
           0507 805 ;
           0507 806 ; INPUTS:
           0507 807 ;
           0507 808 ; R2 = DESIRED BYTE COUNT
           0507 809 ; R3 = IRP ADDRESS
           0507 810 ; PCB$_DIOCNT(R4) ALREADY DECREMENTED
           0507 811 ;
           0507 812 IOC$QTOACP:

```

```

32 A3 52 D0 0507 813      MOVL   R2,IRP$L_BCNT(R3)      ; SET REMAINING BYTES TO TRANSFER
52 18 A3 D0 0508 814      MOVL   IRP$L_WIND(R3),R2      ; GET WINDOW ADDRESS
0B A2 02 E0 050F 815      BBS    #WCB$V_NOTFCP,WCB$B_ACCESS(R2),- ; IF SET THEN
                                0513 816      NOTFCP WCB      ; NOT FCP WINDOW
52 1C A3 D0 0514 817      MOVL   IRP$L_UCB(R3),R2      ; GET ADDRESS OF DEVICE UCB
                                0518 818 ;
                                0518 819 ; FUNCTIONAL DESCRIPTION:
                                0518 820 ;
                                0518 821 ; SUBROUTINE TO QUEUE AN I/O PACKET FOR AN ACP PROCESS AND WAKE
                                0518 822 ; THE PROCESS IF ITS QUEUE WAS PREVIOUSLY EMPTY.
                                0518 823 ;
                                0518 824 ; CALLING SEQUENCE:
                                0518 825 ;
                                0518 826 ; BSBW IOC$WAKACP
                                0518 827 ;
                                0518 828 ; INPUTS:
                                0518 829 ;
                                0518 830 ; R2 = DEVICE UCB ADDRESS
                                0518 831 ; R3 = I/O REQUEST PACKET ADDRESS
                                0518 832 ;
                                0518 833 ; OUTPUTS:
                                0518 834 ;
                                0518 835 ; R4 ALTERED
                                0518 836 ;
;CDS0001 0518 .1 .ENABL LSB
;CDS0001 0518 .2 IOC$WAKACP:: ; QUEUE I/O PACKET AND WAKE ACP PROCESS
;CDS0001 0518 .3 DSIBNT #IPL$ SYNCH ; SYNCHRONIZE ACCESS TO SYSTEM DATA BASE
;CDS0001 52 34 A2 D0 051E .4 MOVL UCB$L_VCB(R2),R2 ; GET ASSOCIATED VCB ADDRESS
;CDS0001 52 10 A2 D0 0522 .5 MOVL VCB$L_AQB(R2),R2 ; GET ACP QUEUE BLOCK ADDRESS
;CDS0001 0C A2 D5 0526 .6 TSTL AQB$L_ACPPID(R2) ; PROCEDURE BASED? NO PID IF SO
;CDS0001 17 13 0529 .7 BEQL XQP ; EQL THEN IS NOT AN ACP
-4 FAD2' 30 052B 841 BSBW EXE$INSERTIRP ; INSERT I/O PACKET IN ACP QUEUE
0E 12 052E 842 BNEQ 10$ ; IF NEQ NOT FIRST IN QUEUE
51 0C A2 D0 0530 843 MOVL AQB$L_ACPPID(R2),R1 ; GET ACP PROCESS ID
FAC9' 30 0534 844 BSBW SCH$WAKE ; WAKE ACP PROCESS
04 50 E8 0537 845 BLBS R0,10$ ; IF LBS ACP STILL PRESENT
053A 846 BUG_CHECK NONEXSTACP ; NONEXISTENT ACP PROCESS
053E 847 10$: ENBINT ; RESTORE SAVED IPL
05 0541 848 RSB ;
0542 849 ;
;CDS0002 0542 .1 ; THIS VOLUME IS BEING HANDLED BY AN XQP INSTEAD OF AN ACP. CALL THE
;CDS0002 0542 .2 ; XQP QUEUEING ROUTINE AS A SPECIAL KERNEL AST TO GET IN THE CONTEXT
;CDS0002 0542 .3 ; OF THE PROCESS THAT INITIATED THIS REQUEST TO HANDLE IT.
;CDS0002 0542 .4 ;
;CDS0002 0542 .5 ;
;CDS0002 0542 .6 XQP:
;CDS0002 55 55 DD 0542 .7 PUSHL R5 ; PRESERVE R5.
;CDS0002 55 60 A3 9E 0544 .8 MOVAB IRP$L_FQFL(R3), R5 ; GET TEMP ACB ADDR INTO R5.
;CDS0002 0B A5 80 8F 90 0548 .9 MOVB #ACB$M_KAST, ACB$B_RMOD(R5) ; NOTE AS SPECIAL KERNEL AST
;CDS0002 0C A5 0C A3 D0 054D .10 MOVL IRP$L_PID(R3), ACB$L_PID(R5) ; COPY PID OF PROCESS.
;CDS0002 0000'CF 9E 0552 .11 MOVAB W^EXE$XQPPKT, ACB$L_KAST(R5) ; ADDR OF QUEUEING ROUTINE.
18 A5 0556
;CDS0002 52 D4 0558 .12 CLRL R2 ; NO PRIORITY INCREMENT.
;CDS0002 FAA3' 30 055A .13 BSBW SCH$QAST ; QUEUE THE AST.
;CDS0002 55 8ED0 055D .14 POPL R5 ; RESTORE R5.
;CDS0002 DC 11 0560 .15 BRB 10$ ; BRANCH TO EXIT.
;CDS0002 0562 .16

```

IOCIOPST  
V03-024

- I/O COMPLETION POSTING  
QUEUE NEXT SEGMENT

3-JUN-1984 11:26:47 VAX-11 Macro V03-01 Page 23  
24-APR-1982 15:46:59 DISK\$VMMASTER:[SYS.SRC]IOCIOPST.(6)

;CDS0002  
;CDS0002

0562 .17 .DSABL LSB  
0562 .18 ;  
0562 850 ; WINDOW IS NOT AN FCP WINDOW, ONLY USED FOR BOOT TIME INITIALIZED WINDOWS  
0562 851 ; FOR CONTIGUOUS FILES. IT IS NOT POSSIBLE TO NEED TO TURN SUCH A WINDOW.  
0562 852 ;  
0562 853 NOTFCPWCBC:  
0562 854 BUG\_CHECK NOTFCPWCBC,FATAL

```

0566 856 .SBTTL BUFFERED READ COMPLETION AST ROUTINE
0566 857 ;++
0566 858 ; FUNCTIONAL DESCRIPTION:
0566 859 ;
0566 860 ; BUFPOST PERFORMS ALL NECESSARY COMPLETION OPERATIONS REQUIRED
0566 861 ; FOR A BUFFERED READ OPERATION IN THE CONTEXT OF THE PROCESS
0566 862 ; ISSUING THE I/O REQUEST.
0566 863 ;
0566 864 ; CALLING SEQUENCE:
0566 865 ;
0566 866 ; JSB BUFPOST
0566 867 ;
0566 868 ; INPUT PARAMETERS:
0566 869 ;
0566 870 ; R4 = CURRENT PROCESS PCB ADDRESS.
0566 871 ; R5 = IRP/AST CONTROL BLOCK.
0566 872 ;
0566 873 ; IMPLICIT INPUTS:
0566 874 ;
0566 875 ; SCH$GL_CURPCB - POINTER TO PCB OF CURRENT PROCESS
0566 876 ;--
0566 877
0566 878 BUFPOST: ; BUFFERED READ COMPLETION
0566 879 PUSHR #^M<R5,R6,R7> ; SAVE REGISTERS
56 2C A5 DO 056A 880 MOVL IRP$S_SVAPTE(R5),R6 ; GET ADDRESS OF I/O BUFFER
57 32 A5 DO 056E 881 MOVL IRP$S_BCNT(R5),R7 ; GET COUNT OF BYTES OR DESCRIPTORS
4B 2A A5 03 E1 0572 882 BBC #IRP$V_COMPLX,IRP$W_STS(R5),40$ ;IF CLR, NOT COMPLEX BUFFER FORMAT
59 2A A5 05 E0 0577 883 BBS #IRP$V_CHAINED,IRP$W_STS(R5),50$ ;IF SET ,CHAINED BUFFERS
56 66 DO 057C 884 MOVL (R6),R6 ; GET ADDRESS OF FIRST BUFFER DESCRIPTOR
50 02 A6 3C 057F 885 10$: MOVZWL 2(R6),R0 ; GET COUNT OF BYTES TO TRANSFER
32 13 0583 886 BEQL 30$ ; IF EQL NONE THIS DESCRIPTOR
51 04 A6 DO 0585 887 MOVL 4(R6),R1 ; GET ADDRESS OF USER BUFFER
50 51 CO 0589 888 ADDL R1,R0 ; CALCULATE ENDING ADDRESS OF BUFFER
51 01FF 8F AA 058C 889 BICW #VA$M_BYTE,R1 ; TRUNCATE ADDRESS TO PAGE BOUNDARY
50 51 C2 0591 890 SUBL R1,R0 ; COMPUTE NUMBER OF BYTES TO PROBE
54 66 3C 0594 891 MOVZWL (R6),R4 ; GET OFFSET TO DATA AREA
53 FE00 8F 32 0597 892 CVTWL #-^X200,R3 ; SET ADDITION CONSTANT
059C 893 20$: IFNOWRT R0,(R1),35$,(R6)[R4] ; CAN BUFFER BE WRITTEN?
51 53 C2 05A3 894 SUBL R3,R1 ; UPDATE ADDRESS OF BUFFER
50 6043 3E 05A6 895 MOVAV (R0)[R3],R0 ; UPDATE REMAINING LENGTH
F0 14 05AA 896 BGTB 20$ ; IF GEQ MORE TO CHECK
02 A6 28 05AC 897 MOVC 2(R6),1(R6)[R4],@4(R6) ; MOVE DATA TO USER BUFFER
01 A644 05AF
04 B6 05B2
55 6E DO 05B4 898 MOVL (SP),R5 ; RESTORE ADDRESS OF I/O PACKET
56 08 CO 05B7 899 30$: ADDL #8,R6 ; ADVANCE TO NEXT BUFFER DESCRIPTOR
C2 57 F5 05BA 900 SOBGTR R7,10$ ; ANY MORE DESCRIPTORS TO PROCESS?
007D 31 05BD 901 BRW 130$ ;
78 11 05C0 902 35$: BRB 120$ ; CONTINUE
017E 30 05C2 903 40$: BSBW MOVBUF ; MOVE BUFFER TO USER
55 6E DO 05C5 904 MOVL (SP),R5 ; RETRIEVE ADDRESS OF I/O PACKET
70 2A A5 0A E1 05C8 905 BBC #IRP$V_MBXIO,IRP$W_STS(R5),130$; BR IF NOT MAILBOX READ
50 02 9A 05CD 906 MOVZBL #RSNS_MAILBOX,R0 ; SET UP RESOURCE RELEASE
FA2D 30 05D0 907 BSBW SCH$RAVAIL ; DECLARE MAILBOX RESOURCE AVAILABLE
68 11 05D3 908 BRB 130$ ;
05D5 909
05D5 910 ;

```

```

05D5 911 ; NB: THE FOLLOWING SECTION OF CODE USES A WORD-SIZE BUFFER LENGTH
05D5 912 ; (ALTHOUGH IRP$L_BCNT WAS EXPANDED TO BE A LONGWORD).
05D5 913 ;
50 57 D0 05D5 914 50$:   MOVL    R7,R0           ; SET LENGTH OF USER BUFFER
51 04 A6 D0 05D8 915     MOVL    4(R6),R1       ; SET ADDRESS OF USER BUFFER
50 51 C0 05DC 916     ADDL    R1,R0           ; CALCULATE END OF USER BUFFER
51 01FF 8F AA 05DF 917     BICW   #VASM_BYTE,R1    ; TRUNCATE TO PAGE BOUNDARY
50 51 C2 05E4 918     SUBL   R1,R0           ; COMPUTE NUMBER OF BYTES TO PROBE
02 00 EF 05E7 919     EXTZV  #0,#2,IRP$B_RMOD(R5),R2 ; GET REQUEST ACCESS MODE
52 0B A5 05EA
53 FE00 8F 32 05ED 920     CVTTL  #-^X200,R3          ; SET ADDITION CONSTANT
05F2 921 60$:   IFNOWRT R0,(R1),90$,R2        ; CAN BUFFER BE WRITTEN?
51 53 C2 05F8 922     SUBL   R3,R1           ; UPDATE ADDRESS OF BUFFER
50 6043 3E 05FB 923     MOVAW  (R0)[R3],R0       ; CALCULATE NEW LENGTH
F1 14 05FF 924     BGTR   60$           ; IF GEQ MORE TO PROBE
53 04 A6 D0 0601 925     MOVL   4(R6),R3          ; GET STARTING ADDRESS OF USER BUFFER
0C A6 57 B1 0605 926 70$:   CMPW   R7,CXB$W_LENGTH(R6) ; REMAINING LENGTH LARGER THAN DATA AREA?
04 04 1E 0609 927     BGEQU  80$           ; IF GEQU YES
0C A6 57 B0 060B 928     MOVW   R7,CXB$W_LENGTH(R6) ; TRUNCATE LENGTH OF DATA AREA
96 0C A6 28 060F 929 80$:   MOVVC  CXB$W_LENGTH(R6),@(R6)+,(R3) ; MOVE DATA TO USER BUFFER
63 0613
55 06 0614 930     MOVL   (SP),R5          ; RETRIEVE ADDRESS OF I/O PACKET
57 08 A6 A2 0617 931     SUBW  CXB$W_LENGTH-4(R6),R7 ; REDUCE REMAINING BYTES TO TRANSFER
0B 13 061B 932     BEQL   100$          ; IF EQL DONE
56 0C A6 D0 061D 933     MOVL  CXB$L_LINK-4(R6),R6 ; GET ADDRESS OF NEXT BUFFER IN CHAIN
E2 12 0621 934     BNEQ  70$           ; IF NEQ MORE TO GO
03 11 0623 935     BRB   100$          ;
0138 30 0625 936 90$:   BSBW  ACCVIO          ; SET ACCESS VIOLATION
50 2C A5 D0 0628 937 100$:  MOVL  IRP$L_SVAPTE(R5),R0 ; GET ADDRESS OF FIRST BUFFER
56 10 A0 D0 062C 938 110$:  MOVL  CXB$L_LINK(R0),R6 ; GET ADDRESS OF NEXT BUFFER
0F 13 0630 939     BEQL   140$          ; IF EQL NONE
F9CB 30 0632 940     BSBW  EXE$DEANONPAGED ; DEALLOCATE BUFFER
50 56 D0 0635 941     MOVL  R6,R0           ; SET ADDRESS OF NEXT BUFFER
F2 11 0638 942     BRB   110$          ;
0123 30 063A 943 120$:  BSBW  ACCVIO          ; SET ACCESS VIOLATION STATUS
50 2C A5 D0 063D 944 130$:  MOVL  IRP$L_SVAPTE(R5),R0 ; GET ADDRESS OF BUFFER TO RELEASE
F9BC 30 0641 945 140$:  BSBW  EXE$DEANONPAGED ; DEALLOCATE BUFFER
00E0 8F BA 0644 946     POPR  #^M<R5,R6,R7> ; RESTORE REGISTERS

```

```

0648 948 .SBTTL DIRECT I/O COMPLETION AST ROUTINE
0648 949 ;++
0648 950 ; FUNCTIONAL DESCRIPTION:
0648 951 ;
0648 952 ; DIRPOST PERFORMS ALL GENERAL I/O COMPLETION ACTIVITIES WHICH
0648 953 ; MUST BE DONE IN THE CONTEXT OF THE PROCESS. THESE INCLUDE
0648 954 ; I/O STATUS POSTING IF AN IOSB WAS SPECIFIED, CHANNEL CONTROL
0648 955 ; BLOCK ACTIVITY COUNT DECREMENTING, QUEUEING OF ANY REQUESTED
0648 956 ; AST OR RELEASE OF THE I/O REQUEST PACKET.
0648 957 ;
0648 958 ; CALLING SEQUENCE:
0648 959 ;
0648 960 ; JSB DIRPOST
0648 961 ;
0648 962 ; INPUT PARAMETERS:
0648 963 ;
0648 964 ; R4 = CURRENT PROCESS PCB ADDRESS.
0648 965 ; R5 = IRP/AST CONTROL BLOCK ADDRESS.
0648 966 ;
0648 967 ; IMPLICIT INPUTS:
0648 968 ;
0648 969 ; SCH$GL_CURPCB - POINTER TO CURRENT PCB
0648 970 ;--
0648 971
0648 972 DIRPOST:
0648 973 EXTZV #IRP$V_BUFIO,#1,IRP$W_STS(R5),R0 ; DIRECT I/O POSTING AST
0648 974 MOVL @#CTL$GL_PHD,R1 ; GET PROCESS HEADER ADDRESS
0648 975 INCL PHD$_DIOCNT(R1)[R0] ; ACCOUNT FOR BUFFERED OR DIRECT I/O
0648 976
0648 977 .IF NE CA$ MEASURE ; CHECK FOR MEASUREMENT ENABLED
0648 978 INCL PMS$GL_DIRIO[R0] ; UPDATE MEASUREMENT I/O COUNTER
0648 979 .ENDC
0648 980
0648 981 BBC #IRP$V_DIAGBUF,IRP$W_STS(R5),10$ ; IF CLR, NO DIAGNOSTIC BUFFER
0648 982 PUSHR #^M<R5,R6,R7> ; SAVE REGISTERS
0648 983 MOVL IRP$_DIAGBUF(R5),R6 ; GET ADDRESS OF DIAGNOSTIC BUFFER
0648 984 MOVZWL IRP$_SIZE(R6),R7 ; GET SIZE OF DIAGNOSTIC BUFFER
0648 985 SUBL #12,R7 ; REDUCE BY SIZE OF BUFFER HEADER
0648 986 BSBW MOVBUF ; MOVE DIAGNOSTIC INFORMATION TO USER
0648 987 POPR #^M<R5,R6,R7> ; RESTORE REGISTERS
0648 988 MOVL IRP$_DIAGBUF(R5),R0 ; RETRIEVE ADDRESS OF DIAGNOSTIC BUFFER
0648 989 BSBW EXE$DEANONPAGED ; DEALLOCATE DIAGNOSTIC BUFFER
0648 990 10$: CVTWL IRP$_CHAN(R5),R0 ; GET CHANNEL NUMBER (NEGATED)
0648 991 MOVAB @CTL$GL_CCBBASE[R0],R1 ; SET CCB BASE ADDRESS
0648 992 DECW CCB$_IOC(R1) ; DECREMENT I/O COUNT FOR CHANNEL
0648 993 BNEQ 30$ ; NOT IDLE YET
0648 994 MOVL CCB$_DIRP(R1),R3 ; GET ADDRESS OF DEACCESS PACKET
0648 995 BEQL 30$ ; IF EQL NONE
0648 996 CLRL CCB$_DIRP(R1) ; CLEAR ADDRESS OF DEACCESS PACKET
0648 997 INCW CCB$_IOC(R1) ; ACCOUNT FOR DEACCESS
0648 998 52 1C A3 D0 069F .1 MOVL IRP$_UCB(R3),R2 ; GET ASSIGNED DEVICE UCB ADDRESS
0648 999 BSBW IOC$WAKACP ; QUEUE I/O PACKET AND WAKE ACP
0648 1000 ; R4 ALTERERED
0648 1001 30$:

```

;LJK45299  
-1



```

06A6 1002 ;
06A6 1003 ; R4 DOES NOT NECESSARILY HAVE CURRENT PCB ADDRESS IN IT AT THIS POINT
06A6 1004 ;
06A6 1005 IOC$DIRPOST1::
50 24 A5 D0 06A6 1006        MOVL  IRP$L_IOSB(R5),R0        ; GET IOSB ADDRESS
           10 13 06AA 1007        BEQL  35$                ; IF EQL NONE SPECIFIED
           02 00 EF 06AC 1008        EXTZV #0,#2,IRP$B_RMOD(R5),R1 ; GET REQUEST ACCESS MODE
51 0B A5           06AF
           06B2 1009        IFNOWRT #8,(R0),35$,R1          ; CAN I/O STATUS BE WRITTEN?
60 38 A5 7D 06B8 1010        MOVQ  IRP$L_IOST1(R5),(R0)      ; MOVE STATUS INTO IOSB
51 0C A5 D0 06BC 1011 35$:      MOVL  IRP$L_PID(R5),R1        ; PROCESS IDENTIFICATION
;WMC0020 17 2A A5 0B E0 06C0 .1      BBS  #IRP$V_EXTEND,IRP$W_STS(R5),50$ ; BRANCH TO DEALLOCATE IRPE'S
;WMC0020 05 0B A5 06 E1 06C5 .2 37$:  BBC  #ACB$V_QUOTA,IRP$B_RMOD(R5),40$ ; IF CLR, NO AST SPECIFIED
;WMC0020           52 D4 06CA .3      CLRL  R2                ; SET NULL PRIORITY INCREMENT
;WMC0020           F931' 31 06CC .4      BRW  SCH$QAST          ; QUEUE AST FOR REQUESTOR
;STJ3049           0A E1 06CF .5 40$:  BBC  #10$V_ERASE,-        ; BRANCH IF NOT AN ERASE REQUEST
;STJ3049           02 20 A5 06D1 .6      IRP$W_FUNC(R5),42$ ;
;STJ3049           50 55 D0 06D4 .7      BSBB CLEANUP_ERASE      ; CLEAN UP AFTER AN ERASE REQUEST
;STJ3049           F924' 31 06D6 .8 42$:  MOVL  R5,R0          ; SETUP ADDRESS FOR DEALLOCATE
;STJ3049           06D9 .9      BRW  EXE$DEANONPAGED        ; AND RELEASE I/O PACKET
-11           06DC .10
           06DC 1023        ;
           06DC 1024        ; DEALLOCATE IRPE'S
           06DC 1025        ;
           06DC 1026
50 54 A5 D0 06DC 1027 50$:      MOVL  IRP$L_EXTEND(R5),R0      ; GET ADDRESS OF FIRST IRPE
           06E0 1028
           54 D4 06E0 1029 60$:      CLRL  R4                ; WILL HOLD ADDRESS OF NEXT IRPE
04 2A A0 0B E1 06E2 1030        BBC  #IRP$V_EXTEND,IRP$W_STS(R0),70$ ; BR. IF NO MORE IRPE'S
54 54 A0 D0 06E7 1031        MOVL  IRP$L_EXTEND(R0),R4        ; SAVE ADDRESS OF NEXT IRPE
           F912' 30 06EB 1032 70$:  BSBW EXE$DEANONPAGED      ; DEALLOCATE IRPE POINTED TO BY R0
           50 54 D0 06EE 1033        MOVL  R4,R0          ; PUT ADDRESS OF NEXT IRPE IN R0
           ED 12 06F1 1034        BNEQ  60$                ; BR. IF THERE IS ANOTHER IRPE
           D0 11 06F3 1035        BRB  37$                ; DONE DEALLOCATING IRPE'S

```

```

;WMC0019      06F5      .1      .SBTTL  ERASE I/O HELPER ROUTINES
;WMC0019      06F5      .2      ;++
;WMC0019      06F5      .3      ; CLEANUP_ERASE
;WMC0019      06F5      .4      ;
;WMC0019      06F5      .5      ; LOCAL SUBROUTINE TO FINISH PROCESSING AN ERASE REQUEST
;WMC0019      06F5      .6      ; THE CLEANUP WILL VARY WITH THE TYPE OF ERASE REQUEST.
;WMC0019      06F5      .7      ; SEE THE ROUTINE HEADER OF THE SUBROUTINE "SETUP_ERASE"
;WMC0019      06F5      .8      ; IN SYSACPFDT FOR DETAILS.
;WMC0019      06F5      .9      ;
;WMC0019      06F5      .10     ; INPUT:          R5 = IRP ADDRESS
;WMC0019      06F5      .11     ; OUTPUT:         NONE.
;WMC0019      06F5      .12     ; --
;WMC0019      06F5      .13
;WMC0019      06F5      .14     CLEANUP_ERASE:                ; FINISH UP AFTER AN ERASE REQUEST
;WMC0019      19      10 06F5      .15     BSBB      CHECK_ERASE                ; IS THIS AN ERASE I/O?
;WMC0019      15 50  E9 06F7      .16     BLBC      RO,69$                ; BRANCH IF NOT
;WMC0019      50 2C A5 D0 06FA      .17     MOVL      IRP$L_SVAPTE(R5),RO        ; GET ADDRESS OF PPT
;WMC0019      00000000'GF D1 0700      .18     BEQL      69$                ; BRANCH IF NONE
;WMC0019      50      0706      .19     CMLP      GAEXE$GL_ERASEPPT,RO        ; IS THIS THE SYSTEM PPT?
;WMC0019      06      13 0707      .20     BEQL      69$                ; BRANCH IF YES'
;WMC0019      50 0C  C2 0709      .21     SUBL2     #12,RO                ; CALC ADDRESS OF HEADER
;WMC0019      F8F1' 30 070C      .22     BSBW      EXE$DEANONPAGED        ; RETURN THE POOL TO THE SYSTEM
;WMC0019      05      070F      .23 69$:    RSB                ; RETURN
;WMC0019      0710      .24
;WMC0019      0710      .25
;WMC0019      0710      .26 ;++
;WMC0019      0710      .27 ; CHECK_ERASE
;WMC0019      0710      .28 ;
;WMC0019      0710      .29 ; LOCAL SUBROUTINE TO DETERMINE IF THIS IRP IS FOR AN ERASE I/O REQUEST.
;WMC0019      0710      .30 ; THIS LEVEL OF PARANOIA IS NECESSARY TO PREVENT THE TOTAL CHAOS THAT
;WMC0019      0710      .31 ; WOULD ARISE SHOULD AN IRP THAT 'LOOKS' LIKE AN ERASE IRP BE TREATED
;WMC0019      0710      .32 ; INCORRECTLY.
;WMC0019      0710      .33 ;
;WMC0019      0710      .34 ; INPUT:          R5 = IRP ADDRESS
;WMC0019      0710      .35 ; OUTPUT:         RO = STATUS; LOW BIT SET IMPLIES THIS IS AN ERASE IRP
;WMC0019      0710      .36 ; --
;WMC0019      0710      .37
;WMC0019      0710      .38     CHECK_ERASE:                ; CHECK FOR ERASE I/O REQUEST
;WMC0019      50 1C A5 D0 0710      .39     MOVL      IRP$L_UCB(R5),RO        ; GET UCB ADDRESS. NOTE: LOW BIT CLEAR
;WMC0019      29 20 A5 E1 0714      .40     BBC       #IO$V_ERASE,-        ; BRANCH IF ERASE MODIFIER NOT SET
;WMC0019      00      0716      .41     ;
;WMC0019      06      0719      .42     CMPZV     #IRP$V_FCODE,-        ; CHECK FUNCTION CODE
;WMC0019      15 20 A5 ED 071B      .43     #IRP$$_FCODE,-        ; ONLY DISKS AND TAPES SUPPORT DSE
;WMC0019      01 91 0721      .44     IRP$W_FUNC(R5),#IO$_DSE ;
;WMC0019      40 A0 0723      .45     BEQL      11$                ; BRANCH IF SO
;WMC0019      00      0725      .46     CMPB      #DC$ DISK,-        ; IS THIS A DISK DEVICE?
;WMC0019      06      0727      .47     UCB$B_DEVCLASS(RO) ;
;WMC0019      00      0729      .48     BNEQ      13$                ; NO - THEREFORE NOT AN ERASE
;WMC0019      20 A5 ED 072F      .49     CMPZV     #IRP$V_FCODE,-        ; CHECK FUNCTION CODE
;WMC0019      00      0729      .50     #IRP$$_FCODE,-        ; ONLY DISKS SUPPORT THE MODIFIED-
;WMC0019      20 A5 072A      .51     IRP$W_FUNC(R5),-        ; WRITE TYPE OF ERASE REQUEST
;WMC0019      0B      072C      .52     #IO$_WRITEPBLK ;
;WMC0019      10 13 072D      .53     BEQL      11$                ;
;WMC0019      00      072F      .54     CMPZV     #IRP$V_FCODE,-        ;
;WMC0019      06      0731      .55     #IRP$$_FCODE,-        ;
;WMC0019      20 A5 0732      .56     IRP$W_FUNC(R5),-        ;

```

;WMC0019	20		0734	.57		#IO\$_WRITEBLK	
;WMC0019	08	13	0735	.58	BEQL	11\$_	
;WMC0019	00	ED	0737	.59	CMPZV	#IRP\$_FCODE,-	
;WMC0019	06		0739	.60		#IRP\$_FCODE,-	
;WMC0019	20	A5	073A	.61		IRP\$_FUNC(R5),-	
;WMC0019		30	073C	.62		#IO\$_WRITEVBLK	
;WMC0019		03	12 073D	.63	BNEQ	13\$_	; NOT A WRITE - THEREFORE NOT AN ERASE
;WMC0019	50	01	88 073F	.64 11\$:	BISB2	#1,R0	; SET LOW BIT IN R0 TO INDICATE ERASE
;WMC0019		05	0742	.65 13\$:	RSB		; RETURN STATUS VALUE

```

0743 1037          .SBTTL  MOVE DATA TO USER BUFFER
0743 1038 ;
0743 1039 ; SUBROUTINE TO MOVE DATA FROM A SIMPLE BUFFERED I/O BUFFER TO A USER BUFFER
0743 1040 ;
0743 1041
0743 1042 MOVBUF:
51 57 D0 0743 1043 MOVL R7,R1 ; MOVE BUFFER
17 13 0746 1044 BEQL 5$ ; SET LENGTH OF USER BUFFER
50 04 A6 D0 0748 1045 MOVL 4(R6),R0 ; BR IF NULL STRING
02 00 EF 074C 1046 EXTZV #0,#2,IRP$B_RMOD(R5),R3 ; GET ADDRESS OF USER BUFFER
53 0B A5 074F ; GET REQUEST ACCESS MODE
00000000'EF 16 0752 1047 JSB EXE$PROBEW ; CHECK ACCESS
05 50 E9 0758 1048 BLBC R0,ACCVIO ; IF LBC, NO ACCESS
96 96 57 28 075B 1049 MOVC R7,@(R6)+,@(R6)+ ; MOVE DATA TO USER BUFFER
05 075F 1050 5$: RSB ; RETURN
38 A5 0C B0 0760 1051 ACCVIO: MOVW #SS$_ACCVIO,IRP$L_IOST1(R5) ; SET FINAL TRANSFER STATUS
05 0764 1052 RSB

```

```

0765 1054          SBTTL  UNLOCK AREAS IN IRPE'S
0765 1055 ;++
0765 1056 ; FUNCTIONAL DESCRIPTION:
0765 1057 ;
0765 1058 ;     THIS ROUTINE UNLOCKS THE AREAS DESCRIBED BY FIELDS IN THE IRPE'S.  EACH
0765 1059 ;     IRPE HAS SPACE TO HOLD TWO AREA DESCRIPTIONS.
0765 1060 ;
0765 1061 ; CALLING SEQUENCE:
0765 1062 ;
0765 1063 ;     BSBW  UNLOCK_MORE
0765 1064 ;
0765 1065 ; INPUT PARAMETERS:
0765 1066 ;
0765 1067 ;     R5 = I/O REQUEST PACKET ADDRESS
0765 1068 ;
0765 1069 ; SIDE EFFECTS:
0765 1070 ;
0765 1071 ;     R0 - R3 ARE NOT PRESERVED
0765 1072 ;--
0765 1073
0765 1074          ASSUME  IRP$L_EXTEND EQ IRPE$L_EXTEND
0765 1075
0765 1076 UNLOCK_MORE:
55  DD 0765 1077          PUSHL  R5              ; SAVE IRP ADDRESS
0767 1078
0767 1079 10$:          ; UNLOCK AREAS SPECIFIED IN NEXT IRPE
0767 1080
55  54 A5  D0 0767 1081          MOVL  IRPE$L_EXTEND(R5),R5      ; GET ADDRESS OF NEXT IRPE
53  2C A5  D0 076B 1082          MOVL  IRPE$L_SVAPTE1(R5),R3      ; GET SVAPTE OF FIRST AREA
0A 13 076F 1083          BEQL  20$              ; BR. IF NOTHING TO UNLOCK
52  30 A5  3C 0771 1084          MOVZWL IRPE$W_BOFF1(R5),R2      ; GET BYTE OFFSET IN PAGE
51  34 A5  D0 0775 1085          MOVL  IRPE$L_BCNT1(R5),R1      ; GET SIZE OF AREA
19 10 0779 1086          BSBW  UNLK              ; UNLOCK FIRST AREA
077B 1087
53  38 A5  D0 077B 1088 20$:          MOVL  IRPE$L_SVAPTE2(R5),R3      ; GET SVAPTE OF SECOND AREA
0A 13 077F 1089          BEQL  30$              ; BR. IF NOTHING TO UNLOCK
52  3C A5  3C 0781 1090          MOVZWL IRPE$W_BOFF2(R5),R2      ; GET BYTE OFFSET IN PAGE
51  40 A5  D0 0785 1091          MOVL  IRPE$L_BCNT2(R5),R1      ; GET SIZE OF AREA
09 10 0789 1092          BSBW  UNLK              ; UNLOCK SECOND AREA
078B 1093
D7 2A A5  0B E0 078B 1094 30$:          BBS   #IRPE$V_EXTEND,IRPE$W_STS(R5),10$ ; BR. IF THERE'S ANOTHER IRPE
55 8ED0 0790 1095          POPL  R5              ; RESTORE R5
05 0793 1096          RSB
0794 1097
0794 1098          ;
0794 1099          ; LOCAL SUBROUTINE TO UNLOCK PAGES
0794 1100          ;
0794 1101          ;     R1 = BYTE COUNT (OR SIZE OF AREA)
0794 1102          ;     R2 = BYTE OFFSET IN PAGE
0794 1103          ;     R3 = SVAPTE OF START OF AREA
0794 1104          ;
0794 1105
51  01FF C142 9E 0794 1106 UNLK:          MOVAB  511(R1)[R2],R1      ; COMBINE OFFSET AND SIZE AND ROUND
51  F7 8F 78 079A 1107          ASHL  #-V$$_BYTE,R1,R1      ; CONVERT TO NUMBER OF PAGES TO UNLOCK
51  079E
F85E 30 079F 1108          BSBW  MMG$UNLOCK          ; UNLOCK PAGES
05 07A2 1109          RSB

```

IOCIOPST  
V03-024

- I/O COMPLETION POSTING  
UNLOCK AREAS IN IRPE'S

3-JUN-1984 11:26:47 VAX-11 Macro V03-01 Page 32  
24-APR-1982 15:46:59 DISK\$VMSMASTER:[SYS.SRC]IOCIOPST(11)

07A3 1110  
07A3 1111  
07A3 1112 .END

IOCIOPST  
Symbol table

- I/O COMPLETION POSTING

ACB\$B_RMOD	=	0000000B		IRP\$L_AST	=	00000010	
ACB\$L_KAST	=	00000018		IRP\$L_ASTPRM	=	00000014	
ACB\$L_PID	=	0000000C		IRP\$L_BCNT	=	00000032	
ACB\$M_KAST	=	00000080		IRP\$L_DIAGBUF	=	0000004C	
ACB\$V_QUOTA	=	00000006		IRP\$L_EXTEND	=	00000054	
ACCVIO	=	00000760	R 02	IRP\$L_FQFL	=	00000060	
AQB\$L_ACPPID	=	0000000C		IRP\$L_IOSB	=	00000024	
BRW_QNXTSEG	=	000000BE	R 02	IRP\$L_IOST1	=	00000038	
BUFIO	=	0000011C	R 02	IRP\$L_IOST2	=	0000003C	
BUFPOST	=	00000566	R 02	IRP\$L_KEYDESC	=	0000005C	
BUG\$_NONEXSTACP	=	*****	X 02	IRP\$L_OBCNT	=	00000044	
BUG\$_NOTFCPWCB	=	*****	X 02	IRP\$L_PID	=	0000000C	
CA\$_MEASURE	=	00000002		IRP\$L_SEGVBN	=	00000048	
CCB\$L_DIRP	=	0000000C		IRP\$L_SVAPTE	=	0000002C	
CCB\$W_IOC	=	0000000A		IRP\$L_UCB	=	0000001C	
CHECK_ERASE	=	00000710	R 02	IRP\$L_WIND	=	00000018	
CLEANUP_ERASE	=	000006F5	R 02	IRP\$M_PAGIO	=	00000004	
CTL\$GL_CCBASE	=	*****	X 02	IRP\$M_SWAPIO	=	00000040	
CTL\$GL_PHD	=	*****	X 02	IRP\$M_VIRTUAL	=	00000010	
CXB\$L_LINK	=	00000010		IRP\$S_FCODE	=	00000006	
CXB\$W_LENGTH	=	0000000C		IRP\$V_BUFIO	=	00000000	
DC\$_DISK	=	00000001		IRP\$V_CHAINED	=	00000005	
DEV\$V_FOD	=	0000000E		IRP\$V_COMPLX	=	00000003	
DEV\$V_SQD	=	00000005		IRP\$V_DIAGBUF	=	00000007	
DIRIO	=	00000064	R 02	IRP\$V_EXTEND	=	0000000B	
DIRPOST	=	00000648	R 02	IRP\$V_FCODE	=	00000000	
EVT\$_COLPGA	=	*****	X 02	IRP\$V_FILACP	=	0000000C	
EVT\$_PFCOM	=	*****	X 02	IRP\$V_FUNC	=	00000001	
EXE\$DEANONPAGED	=	*****	X 02	IRP\$V_KEY	=	0000000F	
EXE\$GL_ERASEPPT	=	*****	X 02	IRP\$V_MBXIO	=	0000000A	
EXE\$INSERTIRP	=	*****	X 02	IRP\$V_PAGIO	=	00000002	
EXE\$INSIOQ	=	*****	X 02	IRP\$V_PHYSIO	=	00000008	
EXE\$PROBEW	=	*****	X 02	IRP\$V_SWAPIO	=	00000006	
EXE\$QXQPPKT	=	*****	X 02	IRP\$V_TERMIO	=	00000009	
IO\$V_ERASE	=	0000000A		IRP\$V_VIRTUAL	=	00000004	
IO\$_DSE	=	00000015		IRP\$W_BOFF	=	00000030	
IO\$_WRITELBLK	=	00000020		IRP\$W_CHAN	=	00000028	
IO\$_WRITEPBLK	=	0000000B		IRP\$W_FUNC	=	00000020	
IO\$_WRITEVBLK	=	00000030		IRP\$W_SIZE	=	00000008	
IOC\$BUFPOST	=	00000122	RG 02	IRP\$W_STS	=	0000002A	
IOC\$CVTLOGPHY	=	*****	X 02	IRPESL_BCNT1	=	00000034	
IOC\$DIRPOST1	=	000006A6	RG 02	IRPESL_BCNT2	=	00000040	
IOC\$GL_PSB	=	*****	X 02	IRPESL_EXTEND	=	00000054	
IOC\$GL_PSFL	=	*****	X 02	IRPESL_SVAPTE1	=	0000002C	
IOC\$IOPOST	=	00000004	RG 02	IRPESL_SVAPTE2	=	00000038	
IOC\$MAPVBLK	=	*****	X 02	IRPESV_EXTEND	=	0000000B	
IOC\$QNXTSEG	=	0000046F	RG 02	IRPESW_BOFF1	=	00000030	
IOC\$QNXTSEG1	=	0000047B	RG 02	IRPESW_BOFF2	=	0000003C	
IOC\$QTOACP	=	00000507	R 02	IRPESW_STS	=	0000002A	
IOC\$WAKACP	=	00000518	RG 02	JIB\$L_BYTCNT	=	00000020	
IOPOST	=	0000000D	R 02	MMG\$DECPHDREF1	=	*****	X 02
IPL\$_IOPOST	=	00000004		MMG\$GL_MAXPFN	=	*****	X 02
IPL\$_SYNCH	=	00000008		MMG\$GL_SYSPHD	=	*****	X 02
IRP\$B_EFN	=	00000022		MMG\$INSFPNT	=	*****	X 02
IRP\$B_RMOD	=	0000000B		MMG\$REFCNTNEG	=	*****	X 02
IRP\$C_LENGTH	=	000000C4		MMG\$RELPFN	=	*****	X 02
IRP\$L_ABCNT	=	00000040		MMG\$SUBSECF	=	*****	X 02

MMG\$UNLOCK	*****	X	02	RSN\$_ASTWAIT	=	00000001		
MMG\$UPDSECAST	*****	X	02	RSN\$_MAILBOX	=	00000002		
MOVBUF	00000743	R	02	SCH\$GL_PCBVEC	*****	X	02	
NOTACP	0000012D	R	02	SCH\$GQ_COLPGWQ	*****	X	02	
NOTFCPWCB	00000562	R	02	SCH\$POSTEF	*****	X	02	
PAGIO	0000018B	R	02	SCH\$QAST	*****	X	02	
PAGIO_DONE	00000218	R	02	SCH\$RAVAIL	*****	X	02	
PAGIO_DONE1	00000239	R	02	SCH\$RSE	*****	X	02	
PAGIO_DONE2	00000240	R	02	SCH\$WAKE	*****	X	02	
PAGIO_ERR	000002E3	R	02	SS\$_ACCVIO	=	0000000C		
PAGIO_ERR_DONE	0000026B	R	02	SS\$_ILLBLKNUM	=	000000DC		
PAGIO_KAST	00000277	R	02	SS\$_INCSGTRA	=	00002234		
PAGIO_OR_SWAPIO	000000C1	R	02	TMP...	=	00000000		
PAGRD_DONE	000001D6	R	02	UCB\$_DEVCLASS	=	00000040		
PAGRD_ERR	00000297	R	02	UCB\$_DEVCHAR	=	00000038		
PAGWRT_ERR	000001D3	R	02	UCB\$_MAXBCNT	=	000000B4		
PAGWRT_ERR1	00000326	R	02	UCB\$_MAXBLOCK	=	000000B0		
PAGWRT_ERR_DONE	000001CF	R	02	UCB\$_VCB	=	00000034		
PCB\$_JIB	=	00000080		UCB\$_QLEN	=	0000006A		
PCB\$_PHD	=	0000006C		UNLK	00000794	R	02	
PCB\$_BIOCNT	=	0000003A		UNLOCK	00000091	R	02	
PCB\$_DIOCNT	=	0000003E		UNLOCK_MORE	00000765	R	02	
PFN\$_AB_STATE	*****	X	02	VA\$_BYTE	=	000001FF		
PFN\$_AB_TYPE	*****	X	02	VA\$_BYTE	=	00000009		
PFN\$_AL_BAK	*****	X	02	VA\$_VPN	=	00000009		
PFN\$_AW_REFCNT	*****	X	02	VCB\$_AQB	=	00000010		
PFN\$_AW_SWPBN	*****	X	02	VIRTUAL_LOGIO	000003AC	R	02	
PFN\$_C_ACTIVE	=	00000007		WCB\$_ACCESS	=	0000000B		
PFN\$_C_BADPAGLST	=	00000002		WCB\$_NOTFCP	=	00000002		
PFN\$_C_PPGTBL	=	00000004		WQH\$_WQCNT	=	00000008		
PFN\$_C_RDERR	=	00000004		XQP	00000542	R	02	
PFN\$_M_COLLISION	=	00000010						
PFN\$_M_DELCON	=	00000010						
PFN\$_M_PAGTYP	=	00000007						
PFN\$_V_COLLISION	=	00000004						
PFN\$_V_GBLBAK	=	00000017						
PFN\$_V_MODIFY	=	00000007						
PFN_IO_DONE	0000037B	R	02					
PHD\$_L_DIOCNT	=	00000054						
PHD\$_W_PHVINDEXT	=	00000042						
PMS\$_END_RQ	*****	X	02					
PMS\$_GL_DIRIO	*****	X	02					
PMS\$_GL_SPLIT	*****	X	02					
PMS\$_START_RQ	*****	X	02					
PR\$_IPL	=	00000012						
PR\$_SIRR	=	00000014						
PRIS\$_IOCOM	=	00000001						
PRIS\$_NULL	=	00000000						
PRIS\$_TICOM	=	00000004						
PRIS\$_TOCOM	=	00000003						
PRITBL	00000000	R	02					
PTE\$_M_OWN	=	01800000						
PTE\$_M_PROT	=	78000000						
PTE\$_S_PFN	=	00000015						
PTE\$_V_PFN	=	00000000						
PTE\$_V_VALID	=	0000001F						
QNX_TSEG	0000040B	R	02					



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

PSECT name	Allocation	PSECT No.	Attributes											
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE		
\$ABS\$	00000000 ( 0.)	01 ( 1.)	NOPIC USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE		
\$AEXENONPAGED	000007A3 ( 1955.)	02 ( 2.)	NOPIC USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG		

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.09	00:00:02.62
Command processing	74	00:00:00.57	00:00:04.90
Pass 1	568	00:00:27.01	00:01:48.68
Symbol table sort	0	00:00:03.89	00:00:13.26
Pass 2	250	00:00:07.44	00:00:51.36
Symbol table output	23	00:00:00.20	00:00:00.73
Psect synopsis output	3	00:00:00.03	00:00:00.20
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	930	00:00:39.24	00:03:01.76

The working set limit was 1950 pages.  
141145 bytes (276 pages) of virtual memory were used to buffer the intermediate code.  
There were 130 pages of symbol table space allocated to hold 2474 non-local and 100 local symbols.  
1388 source lines were read in Pass 1, producing 20 object records in Pass 2.  
41 pages of virtual memory were used to define 40 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	28
_\$255\$DUA28:[SYS.LIB]STARLET.MLB;2	9
TOTALS (all libraries)	37

2625 GETS were required to define 37 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:IOCIOPST/OBJ=OBJ\$:IOCIOPST MSRC\$:IOCIOPST/UPDATE=(ENH\$:IOCIOPST)+EXECMLS/LIB



**COMDRV SUB**

(2)	93	COM\$DELATTNAST - DELIVER ATTENTION ASTS
(3)	131	COM\$FLUSHATTNS - FLUSH ATTENTION AST LIST
(4)	174	COM\$POST - POST I.O COMPLETION INDEPENDENT OF UNIT STATUS
(5)	204	COM\$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
(6)	249	COM\$SETATTNAST - SET UP ATTENTION AST
(7)	332.6	COM\$DELCTRLAST - DELIVER CONTROL ASTS
(8)	413	COM\$FLUSHCTRLS - FLUSH CONTROL AST LIST
(9)	465	COM\$SETCTRLAST - SET UP CONTROL AST
(10)	602	COM\$BLDCTRLAST - BUILD CONTROL AST
(11)	691	COM\$FILLCTRLAST - FILLIN A CONTROL AST CONTROL BLOCK

;MHB0137  
-1

;MHB0137  
;MHB0137  
;MHB0137  
;MHB0137  
RKS006  
RKS006  
RKS006  
;DWT0157  
;DWT0157  
;DWT0157  
;JLV0272  
;JLV0272  
;JLV0272  
;JLV0272  
;JLV0272  
;JLV0272

```
0000 1 .TITLE COMDRVSUB - COMMUNUCATION DRIVERS SUBROUTINES
0000 1 .IDENT 'V03-007'
0000 3 ;
0000 4 ;*****
0000 5 ;*
0000 6 ;* COPYRIGHT (c) 1978, 1980, 1982 BY
0000 7 ;* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 ;* ALL RIGHTS RESERVED.
0000 9 ;*
0000 10 ;* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 ;* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 ;* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 ;* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 ;* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 ;* TRANSFERRED.
0000 16 ;*
0000 17 ;* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 ;* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 ;* CORPORATION.
0000 20 ;*
0000 21 ;* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 ;* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 ;*
0000 24 ;*
0000 25 ;*****
0000 26 ;
0000 27 ;++
0000 28 ; FACILITY:
0000 29 ;
0000 30 ; VAX/VMS I/O DRIVERS
0000 31 ;
0000 32 ; ABSTRACT:
0000 33 ;
0000 34 ; THIS MODULE CONTAINS SUBROUTINES FOR THE TERMINAL,MAILBOX AND DMC11 DRIVERS.
0000 35 ;
0000 36 ; AUTHOR:
0000 37 ;
0000 38 ; R.HEINEN 8-SEPT-1977
0000 39 ;
0000 40 ; REVISION HISTORY:
0000 41 ;
0000 .1 ; V03-007 MHB0137 Mark Bramhall 12-Apr-1984
0000 .2 ; Define minimum deallocatable block size as FKB$C_LENGTH.
0000 .3 ; Restore stacked register after BADDALRQSZ bug check.
0000 .4 ;
0000 .5 ; V03-006 RKS006 RICK SPITZ 2-MAR-1984
0000 .6 ; Make sure that TAST is removed from UCB queue when
0000 .7 ; TAST replaced and currently busy.
0000 .8 ;
0000 .9 ; V03-005 DWT0157 David W. Thiel 30-DEC-1983
0000 .10 ; Modify COM$DRVDEALMEM to avoid unnecessary forks.
0000 .11 ;
0000 .12 ; V03-004 JLV0272 Jake VanNoy 14-JUN-1983
0000 .13 ; Add abort I/O flag to out of band logic. Also
0000 .14 ; add new entry points for checking PIDs to qualify
0000 .15 ; AST delivery.
0000 .16 ;
```

```
;RKS0003      0000 .17 ;      V03-003 RKS0003      RICK SPITZ      23-SEP-1982
;RKS0003      0000 .18 ;      INSURE THAT TAST IS NOT BUSY WHEN A REPLACE CURRENT
;RKS0003      0000 .19 ;      BLOCK OPERATION OCCURS. IF SO, FLAG THE CURRENT BLOCK
;RKS0003      0000 .20 ;      FOR DELETE AND BUILD A NEW BLOCK.
;RKS0003      0000 .21 ;
;KDM0002      0000 .22 ;      V03-002 KDM0002      Kathleen D. Morse      28-Jun-1982
;KDM0002      0000 .23 ;      Added $IODEF and $$SDEF.
-20           0000 62 ;
           0000 63 ;
           0000 64 ;--
           0000 65 ;
           0000 66 ; EXTERNAL SYMBOLS
           0000 67 ;
           0000 68      $ACBDEF      ; DEFINE AST CONTROL BLOCK
           0000 69      $CCBDEF      ; DEFINE CCB
           0000 70      $DYNDEF      ; DEFINE DYNAMIC MEMORY BLOCKS
;MHB0137      0000 .1      $FKBDEF      ; DEFINE FORK BLOCK
;KDM0002      0000 .2      $IODEF      ; DEFINE I/O FUNCTION CODES
           0000 71      $IPLDEF      ; DEFINE IPL LEVELS
           0000 72      $IRPDEF      ; DEFINE I/O PACKET
           0000 73      $PCBDEF      ; DEFINE PCB
           0000 74      $PRDEF      ; DEFINE PROCESSOR REGISTERS
           0000 75      $PRIDEF      ; DEFINE NEW PRIORITIES
           0000 76      $PRVDEF      ; DEFINE PRIVELEGE BITS
           0000 77      $PSLDEF      ; DEFINE PSL
;KDM0002      0000 78      $RSNDEF      ; DEFINE RESOURCES
           0000 .1      $$SDEF      ; DEFINE SYSTEM STATUS CODES
           0000 79      $TASTDEF      ; DEFINE TERMINAL AST BLOCK
           0000 80      $UCBDEF      ; DEFINE UCB
           0000 81 ;
           0000 82 ; LOCAL DEFINITIONS
           0000 83 ;
00000000      0000 84 P1= 0
00000004      0000 85 P2= 4
00000008      0000 86 P3= 8
0000000C      0000 87 P4= 12
00000010      0000 88 P5= 16
00000014      0000 89 P6= 20
           0000 90
00000000      0000 91      .PSECT WIONONPAGED
```

```

0000 93          .SBTTL  COM$DELATTNAST - DELIVER ATTENTION ASTS
0000 94 ;++
;JLV0272 0000 .1 ; COM$DELATTNAST - DELIVER ATTENTION ASTS
;JLV0272 0000 .2 ; COM$DELATTNASTP - DELIVER ATTENTION ASTS BY PID
-1       0000 96 ;
0000 97 ; FUNCTIONAL DESCRIPTION:
0000 98 ;
0000 99 ; THIS ROUTINE IS USED BY THE TERMINAL AND MAILBOX DRIVERS TO DELIVER
0000 100 ; ALL OF THE ASTS AWAITING ATTENTION. THE CONTROL BLOCKS ARE USED AS FORK BLOCKS
0000 101 ; TO IPL$_QUEUEAST.
0000 102 ;
0000 103 ; INPUTS:
0000 104 ;
0000 105 ;         R4 = ADDRESS OF LIST HEAD OF AST CONTROL BLOCKS
0000 106 ;         R5 = UCB OF UNIT
;JLV0272 0000 .1 ;         R6 = ACTIVE PID (AT ENTRY POINT COM$DELATTNASTP)
0000 107 ;
0000 108 ; OUTPUTS:
0000 109 ;
0000 110 ;         R2,R3,R4,R5 ARE PRESERVED.
0000 111 ;--
0000 112 COM$DELATTNAST:: ; DELIVER ATTENTION ASTS
;JLV0272 00F8 8F BB 0000 .1 PUSHR  #^M<R3,R4,R5,R6,R7> ; SAVE
;JLV0272 56 D4 0004 .2 CLRL  R6 ; CLEAR SO ALL ASTs WILL BE DELIVERED
;JLV0272 04 11 0006 .3 BRB  ATTN2 ; continue
;JLV0272 0008 .4
;JLV0272 0008 .5 COM$DELATTNASTP:: ; DELIVER ATTENTION ASTS BY PID
;JLV0272 00F8 8F BB 0008 .6 PUSHR  #^M<R3,R4,R5,R6,R7> ; SAVE
;JLV0272 000C .7 ATTN2:
;JLV0272 57 54 D0 000C .8 MOVL  R4,R7 ; R7 WILL TRACK LIST
;JLV0272 000F .9
;JLV0272 55 67 D0 000F .10 10$: MOVL  (R7),R5 ; GET NEXT ENTRY
;JLV0272 36 13 0012 .11 BEQL  50$ ; IF EQL THEN NONE
;JLV0272 56 D5 0014 .12 TSTL  R6 ; CHECKING PIDS?
;JLV0272 0B 13 0016 .13 BEQL  30$ ; NOT IF ZERO
;JLV0272 56 24 A5 D1 0018 .14 CMPL  ACB$_KAST+12(R5),R6 ; COMPARE PIDS
;JLV0272 05 13 001C .15 BEQL  30$ ; EQUAL, DELIVER AST
;JLV0272 001E .16 ;
;JLV0272 001E .17 ; PIDS NOT EQUAL, SKIP THIS ENTRY AND LEAVE IT IN QUEUE
;JLV0272 001E .18 ;
;JLV0272 57 55 D0 001E .19 MOVL  R5,R7 ; NEXT
;JLV0272 EC 11 0021 .20 BRB  10$ ; LOOP
;JLV0272 0023 .21 ;
;JLV0272 0023 .22 ; DELIVER AST AND CLOSE UP LIST
;JLV0272 0023 .23 ;
;JLV0272 67 65 D0 0023 .24 30$: MOVL  (R5),(R7) ; CLOSE LIST
-4       E6 AF 9F 0026 117 PUSHAB B^10$ ; SET UP RETURN ADDRESS
0029 118 FORK ; CREATE FORK PROCESS
002F 119 ;
002F 120 ; AST QUEUE FORK PROCESS
002F 121 ;
10 A5 18 A5 7D 002F 122 MOVQ  ACB$_KAST(R5),ACB$_AST(R5); REARRANGE ENTRIES
0B A5 20 A5 90 0034 123 MOVVB ACB$_KAST+8(R5),ACB$_RMOD(R5);
0C A5 24 A5 D0 0039 124 MOVL  ACB$_KAST+12(R5),ACB$_PID(R5);
18 A5 18 A5 D4 003E 125 CLRL  ACB$_KAST(R5) ;
52 01 9A 0041 126 MOVZBL #PRI$_IOCOM,R2 ; SET UP PRIORITY INCREMENT
00000000'GF 17 0044 127 JMP  G^SCH$QAST ; QUEUE THE AST

```

COMDRVSUB  
V03-007

- COMMUNUCATION DRIVERS SUBROUTINES 3-JUN-1984 10:56:56 VAX-11 Macro V03-01 Page 4  
COM\$DELATTNAST - DELIVER ATTENTION ASTS 24-APR-1982 15:46:24 DISK\$VMSMASTER:[SYS.SRC]COMDRVSUB.(2)

;JLV0272  
-1

00F8 8F BA 004A .1 50\$: POPR #^M<R3,R4,R5,R6,R7>  
05 004E 129 RSB ;



```

004F 131          .SBTTL  COM$FLUSHATTNS - FLUSH ATTENTION AST LIST
004F 132 ;++
004F 133 ; COM$FLUSHATTNS - FLUNS ATTENTION AST LIST
004F 134 ;
004F 135 ; THIS ROUTINE IS USED BY THE TERMINAL AND MAILBOX DRIVERS TO FLUSH
004F 136 ; AN ATTENTION AST LIST. THIS IS DONE AT CANCEL I/O TIME AND WHEN A
004F 137 ; QIO SPECIFIES A 0 AST ADDRESS ON A SET ATTENTION AST FUNCTION.
004F 138 ; IF THE AST CONTROL BLOCK OWNER IS NO LONGER IN THE SYSTEM THE AST IS ALSO
004F 139 ; FLUSHED.
004F 140 ;
004F 141 ;
004F 142 ; INPUTS:
004F 143 ;
004F 144 ;         R4 = PCB ADDRESS
004F 145 ;         R5 = UCB ADDRESS OF RELATED UNIT
004F 146 ;         R6 = CHANNEL NUMBER
004F 147 ;         R7 = LIST HEAD
004F 148 ;
004F 149 ; OUTPUTS:
004F 150 ;
004F 151 ;         R0 = SS$_NORMAL
004F 152 ;         R1,R2,R7 ARE DESTROYED.
004F 153 ;
004F 154 ;--
004F 155 COM$FLUSHATTNS::
004F 156          DSBINT  UCB$_DIPL(R5)          ; FLUSH ATTENTION AST LIST
004F 157 10$:    MOVL   (R7),R0                ; DISABLE INTERRUPTS
50   67   D0   0056 10$:    MOVL   (R7),R0                ; GET LIST ENTRY
004F 158          BEQL   50$                    ; IF EQL THEN DONE
24  A0   60   A4   D1   005B 159          CMPL   PCB$_PID(R4),ACB$_KAST+12(R0); PID MATCH?
004F 160          BNEQ   40$                    ; IF NEQ THEN NO
22  A0   56   B1   0062 161          CMPW   R6,ACB$_KAST+10(R0) ; CHANNEL MATCH?
004F 162          BNEQ   40$                    ; IF NEQ THEN NO
67   60   D0   0068 163          MOVL   (R0),(R7)                ; CLOSE UP LIST TO REMOVE ENTRY
004F 164          ENBINT                          ; REENABLE INTERRUPTS
004F 165          INCW   PCB$_ASTCNT(R4)         ; RESTORE AST QUOTA
38  A4   B6   006E 166          BSBB   COM$DRVDEALMEM          ; DEALLOCATE THE BLOCK
004F 167          BRB    COM$FLUSHATTNS          ; CONTINUE
57   50   D0   0075 168 40$:    MOVL   R0,R7                ; LOOK TO NEXT ENTRY
004F 169          BRB    10$                    ; CONTINUE
004F 170          ENBINT                          ; REENABLE INTERRUPTS
50   01   9A   007D 171 50$:    MOVZBL #SS$_NORMAL,R0          ; SET NORMAL RETURN
004F 172          RSB
004F 0080

```

```

0081 174 .SBTTL COM$POST - POST I.O COMPLETION INDEPENDENT OF UNIT STATUS
0081 175 ;++
0081 176 ; COM$POST - POST I/O COMPLETION INDEPENDENT OF UNIT STATUS
0081 177 ;
0081 178 ; FUNCTIONAL DESCRIPTION:
0081 179 ;
0081 180 ; THIS ROUTINE IS USED BY THE TERMINAL, MAILBOX AND DMC DRIVER TO COMPLETE
0081 181 ; I/O OPERATIONS INDEPENDENT OF THE STATUS OF THE UNIT. NO ATTEMPT IS MADE
0081 182 ; TO DE-QUEUE ANOTHER PACKET OR CHANGE THE BUSY STATUS OF THE UNIT.
0081 183 ;
0081 184 ; INPUTS:
0081 185 ;
0081 186 ; R3 = I/O PACKET ADDRESS
0081 187 ; R5 = UCB ADDRESS
0081 188 ;
0081 189 ; IMPLICIT INPUTS:
0081 190 ;
0081 191 ; CALLER AT DRIVER FORK IPL OR GREATER.
0081 192 ; IRP$L_MEDIA AND IRP$L_MEDIA+4 ARE THE IOSB QUAD WORD.
0081 193 ;
0081 194 ; OUTPUTS:
0081 195 ;
0081 196 ; R0,R1 ARE DESTROYED.
0081 197 ;--
0081 198 COM$POST:: ; COMPLETE I/O
70 A5 D6 0081 199 INCL UCB$L_OPCNT(R5) ; INCREMENT OPERATION COUNT
63 OE 0084 200 INSQUE (R3),@IOC$GL_PSBLL ; INSERT PACKET ON QUEUE
00000000'FF 0086
008B 201 SOFTINT #IPL$_IOPOST ; REQUEST FORK
05 008E 202 RSB ; RETURN

```

```
008F 204 .SBTTL COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
008F 205 ;++
008F 206 ; COM$DRVDEALMEM - DEALLOCATE DRIVER MEMORY
008F 207 ;
008F 208 ; FUNCTIONAL DESCRIPTION:
008F 209 ;
008F 210 ; THIS ROUTINE IS USED BY DRIVERS TO DEALLOCATE SYSTEM DYNAMIC MEMORY.
008F 211 ;
008F 212 ; IT CAN BE CALLED AT ANY IPL.
008F 213 ;
008F 214 ; INPUTS:
008F 215 ;
008F 216 ; R0 = ADDRESS OF THE BLOCK TO DEALLOCATE
008F 217 ;
008F 218 ; *****
008F 219 ;
;MHB0137 008F .1 ; THE BUFFER MUST BE AT LEAST "FKB$C_LENGTH" BYTES LONG
;MHB0137 008F .2 ; IF A FORK IS REQUIRED. THIS IS BECAUSE WE TURN THE
;MHB0137 008F .3 ; BUFFER INTO A FORK BLOCK FOR THE DELAYED DEALLOCATION.
-1 008F 221 ;
008F 222 ; *****
008F 223 ;
008F 224 ; OUTPUTS:
008F 225 ;
008F 226 ; R0-R5 ARE PRESERVED.
008F 227 ;--
008F 228 COM$DRVDEALMEM:: ; DEALLOCATE DRIVER MEMORY
;MHB0137 53 DD 008F .1 PUSHL R3 ; SAVE A REGISTER
;MHB0137 0091 .2 SAVIPL R3 ; FETCH CURRENT IPL LEVEL
;MHB0137 08 53 D1 0094 .3 Cmpl R3,#IPL$_SYNCH ; COMPARE CURRENT IPL TO SYNCH
;MHB0137 0B 14 0097 .4 BGTR 10$ ; BRANCH IF IPL LEVEL > SYNCH
;MHB0137 07 BB 0099 .5 PUSHR #^M<R0,R1,R2> ; SAVE MORE REGISTERS
;MHB0137 00000000'GF 16 009B .6 JSB G^EXE$DEANONPAGED ; DEALLOCATE MEMORY
;MHB0137 0F BA 00A1 .7 POPR #^M<R0,R1,R2,R3> ; RESTORE REGISTERS
;MHB0137 05 00A3 .8 RSB ; RETURN, ALL REGISTERS PRESERVED
;MHB0137 00A4 .9
;MHB0137 08 A0 18 B1 00A4 .10 10$: CMPW #FKB$C_LENGTH,FKB$W_SIZE(R0) ; BIG ENOUGH FOR A FORK BLOCK?
;MHB0137 25 1A 00A8 .11 BGTRU 30$ ; IF GTRU THEN NO - ERROR
;MHB0137 7E 54 7D 00AA .12 MOVQ R4,-(SP) ; SAVE FORKING REGS
;MHB0137 B0 00AD .13 MOVW #<DYN$C_FRK!- ; INSERT STRUCTURE TYPE
;MHB0137 00AE .14 <IPL$_QUEUEAST@8>>,- ; AND PROPER IPL
;MHB0137 0608 8F 00AE .15 FKB$_TYPE(R0) ; FOR THIS TO BE A FORK BLOCK
;MHB0137 0A A0 00B1
;MHB0137 00B3 .16 ASSUME FKB$_FIPL EQ FKB$_TYPE+1
-4 ;MHB0137 55 50 D0 00B3 233 MOVL R0,R5 ; COPY ADDRESS
;MHB0137 CB'AF 9F 00B6 234 PUSHAB B^20$ ; SET UP RETURN ADDRESS
;MHB0137 00000000'GF 16 00B9 235 JSB G^EXE$FORK ; CREATE FORK
00BF 236 ;
00BF 237 ; IPL$_QUEUEAST FORK ROUTINE
00BF 238 ;
;MHB0137 50 55 D0 00BF 239 MOVL R5,R0 ; DEALLOCATE THE BLOCK
;MHB0137 00000000'GF 17 00C2 240 JMP G^EXE$DEANONPAGED ;
;MHB0137 54 8E 7D 00C8 .1 20$: MOVQ (SP)+,R4 ; RESTORE REGISTERS
;MHB0137 53 8ED0 00CB .2 25$: POPL R3 ; RESTORE R3
;MHB0137 05 00CE .3 RSB ;
;MHB0137 00CF .4 ;
;MHB0137 00CF .5 ; BUGCHECK ON TOO SMALL A BUFFER
```

COMDRVSUB  
V03-007

- COMMUNUCATION DRIVERS SUBROUTINES 3-JUN-1984 10:56:56 VAX-11 Macro V03-01 Page 8  
COM\$DRVDEALMEM - DEALLOCATE DRIVER MEMOR 24-APR-1982 15:46:24 DISK\$VMMASTER:[SYS.SRC]COMDRVSUB.(5)

;MHB0137  
;MHB0137  
;MHB0137

		00CF	.6	;					
		00CF	.7	30\$:	BUG_CHECK	BADDALRQSZ			; BUGCHECK
F6	11	00D3	.8		BRB	25\$			; CONTINUE

```

00D5 249 .SBTTL COM$SETATTNAST - SET UP ATTENTION AST
00D5 250 ;++
00D5 251 ; COM$SETATTNAST - SET UP ATTENTION AST
00D5 252 ;
00D5 253 ; FUNCTIONAL DESCRIPTION:
00D5 254 ;
00D5 255 ; THIS ROUTINE IS A SUBROUTINE USED BY THE TERMINAL AND MAILBOX DRIVERS
00D5 256 ; TO PROCESS REQUESTS FOR ENABLE OR DISABLE OF ATTENTION ASTS.
00D5 257 ; P1 IS THE ADDRESS OF THE AST SERVICE FOR ENABLES. P1 = 0 FOR DISABLE.
00D5 258 ; FOR DISABLES, THE SPECIFIED LIST IS SEARCHED AND THE ENTRY EXTRACTED AND
00D5 259 ; DEALLOCATED.
00D5 260 ; FOR ENABLES, A CONTROL BLOCK IS SET UP THAT WILL DOUBLE AS THE AST CONTROL
00D5 261 ; BLOCK WHEN THE AST IS DELIVERED. THE BLOCK IS FORMATTED AS FOLLOWS:
00D5 262 ;
00D5 263 ; ACB$B_RMOD = IPR$_QUEUEAST
00D5 264 ; ACB$_KAST = AST_PC
00D5 265 ; ACB$_KAST+4 = AST_PARAMETER (P2)
00D5 266 ; ACB$_KAST+8 = ACCESS_MODE_OF_REQUEST
00D5 267 ; ACB$_KAST+10 = CHANNEL_NUMBER
00D5 268 ; ACB$_KAST+12 = PID_OF_REQUEST
00D5 269 ;
00D5 270 ; THE NEW BLOCK IS PLACED AT THE HEAD OF THE CURRENT LIST.
00D5 271 ;
00D5 272 ; IN BOTH CASES THE I/O IS COMPLETED.
00D5 273 ;
00D5 274 ; INPUTS:
00D5 275 ;
00D5 276 ; R3 = I/O PACKET ADDRESS
00D5 277 ; R4 = CURRENT PCB
00D5 278 ; R5 = UCB ADDRESS
00D5 279 ; R6 = ASSIGNED CCB
00D5 280 ; R7 = ADDRESS OF THE CONTROL AST LIST HEAD TO CHANGE
00D5 281 ; AP = ADDRESS OF THE QIO_ARGLIST
00D5 282 ;
00D5 283 ; OUTPUTS:
00D5 284 ;
00D5 285 ; R0 = STATUS OF THE I/O
00D5 286 ; R3 = PACKET ADDRESS
00D5 287 ; R5 = UCB ADDRESS
00D5 288 ;
00D5 289 ; NO OTHER REGISTERS ARE PRESERVED.
00D5 290 ;
00D5 291 ; COMPLETION CODES:
00D5 292 ;
00D5 293 ; SS$_NORMAL
00D5 294 ; SS$_EXQUOTA -- BUFFERED I/O OR AST QUOTA FAILURE
00D5 295 ; SS$_INSUFMEM -- DYNAMIC MEMORY FAILURE
00D5 296 ; --
00D5 297 COM$SETATTNAST::
56 28 A3 3C 00D5 298 MOVZWL IRP$W_CHAN(R3),R6 ; SET UP ATTENTION AST
58 6C DO 00D9 299 MOVL P1(AP),R8 ; GET PACKET CHANNEL NUMBER
;DWT0157 5B 13 00DC .1 BEQL 10$ ; GET USER AST ADDRESS
-1 ; IF EQL THEN DISABLE FUNCTION
00DE 301 ;
00DE 302 ; REQUEST TO ENABLE AST
00DE 303 ;
00DE 304 ; SET UP AST BLOCK
00DE 305 ;

```

```

50 1C 3C 00DE 306 MOVZWL #SS$_EXQUOTA,R0 ; ASSUME AST QUOTA EXCEEDED
38 A4 B5 00E1 307 TSTW PCB$_ASTCNT(R4) ; AST QUOTA ALLOW CONTINUE?
56 15 00E4 308 BLEQ 20$ ; IF LEQ THEN NO
38 A4 B7 00E6 309 DECW PCB$_ASTCNT(R4) ; ADJUST QUOTA
51 28 9A 00E9 310 MOVZBL #ACB$_KAST+16,R1 ; SET SIZE OF NEEDED BLOCK
53 DD 00EC 311 PUSHL R3 ; SAVE PACKET ADDRESS
00000000'GF 16 00EE 312 JSB G^EXE$ALLOCBUF ; ALLOCATE THE BUFFERED BLOCK
53 B8D0 00F4 313 POPL R3 ; RESTORE PACKET ADDRESS
42 50 E9 00F7 314 BLBC R0,20$ ; IF LOW SET THEN ALLOCATED
0B A2 06 90 00FA 315 MOVB #IPL$_QUEUEAST,ACB$_RMOD(R2); INSERT FORK IPL
18 A2 58 D0 00FE 316 MOVL R8,ACB$_KAST(R2) ; INSERT AST ROUTINE ADDRESS
1C A2 04 AC D0 0102 317 MOVL P2(AP),ACB$_KAST+4(R2) ; INSERT PARAMETER FOR AST
02 00 EF 0107 318 EXTZV #0,#2,P3(AP),R0 ; GET REQUEST ACCESS MODE
50 08 AC 010A JSB G^EXE$MAXACMODE ; MAXIMIZE ACCESS MODE
00000000'GF 16 010D 319 MOVZBL R0,ACB$_KAST+8(R2) ; INSERT IN CONTROL BLOCK
20 A2 50 9A 0113 320 BISB #ACB$_QUOTA,ACB$_KAST+8(R2); INSERT TARGET ACCESS MODE
20 A2 40 8F 88 0117 321 MOVW R6,ACB$_KAST+10(R2) ; SAVE CHANNEL
22 A2 56 B0 011C 322 MOVL PCB$_PID(R4),ACB$_KAST+12(R2); INSERT PID ADDRESS OF REQUESTOR
24 A2 60 A4 D0 0120 323
0125 324 ;
0125 325 ; LOCK OUT INTERRUPTS TO ENTER BLOCK ON UCB
0125 326 ;
0125 327 DSBINT UCB$_DIPL(R5) ; LOCK OUT INTERRUPTS
62 67 D0 012C 328 MOVL (R7),(R2) ; MERGE WITH CURRENT ENTRY
67 52 D0 012F 329 MOVL R2,(R7) ; INSERT NEW ENTRY VALUE
ENBINT ; LOWER IPL
50 01 9A 0135 331 MOVZBL #SS$_NORMAL,R0 ; SET NORMAL RETURN
05 0138 332 RSB ; RETURN VIA CALLER
0139 .1
FF13 31 0139 .2 10$: BRW COM$FLUSHATTNS ; DISABLE FUNCTION
013C .3
00000000'GF 17 013C .4 20$: JMP G^EXE$ABORTIO ; ABORT THE I/O

```

;DWT0157  
;DWT0157  
;DWT0157  
;DWT0157

```

;DWT0157          0142      .6          .SBTTL  COM$DELCTRLAST - DELIVER CONTROL ASTS
;DWT0157          0142      .7      ;++
;JLV0272          0142      .8      ;COM$DELCTRLAST - DELIVER CONTROL ASTS
;JLV0272          0142      .9      ;COM$DELCTRLASTP - DELIVER CONTROL ASTS AND CHECK PIDS
-5                0142      338      ;
                0142      339      ; FUNCTIONAL DESCRIPTION:
                0142      340      ;
                0142      341      ; THIS ROUTINE IS USED BY THE TERMINAL SERVICES TO DELIVER
                0142      342      ; ALL OF THE ASTS AWAITING ATTENTION WHICH MATCH THE CONDITION.
                0142      343      ; THE AST BLOCK IS NOT DELETED AND IS REUSED.
                0142      344      ;
                0142      345      ; INPUTS:
                0142      346      ;
                0142      347      ;         R3 = MATCH CHARACTER
                0142      348      ;         R4 = ADDRESS OF LIST HEAD OF AST CONTROL BLOCKS
                0142      349      ;         R5 = UCB OF UNIT
                0142      .1      ;         R6 = ACTIVE PID (AT ENTRY POINT COM$DELCTRLASTP)
                0142      .2      ;
                0142      .3      ; OUTPUTS:
                0142      .4      ;
                0142      .5      ;         R3 = (low byte) CHARACTER TO INCLUDE IN DATA STREAM OR NULL
                0142      .6      ;         TAST$V_ABO - set to signal driver to abort I/O in progress
                0142      .7      ;         TAST$V_INC - set to include this character in datastream
                0142      .8      ;
                0142      .9      ;         R1,R4 DESTROYED
                0142      .10     ;--
                0142      .11     ;
                0142      .12     COM$DELCTRLAST::                ; DELIVER CONTROL ASTS
                0142      .13     PUSHR  #^M<R2,R3,R5,R6>        ; SAVE REGISTERS
                0142      .14     CLRL   R6                    ; CLEAR SO ALL ASTs WILL BE DELIVERED
                0142      .15     BRB   CTRL2                  ; continue
                0142      .16     ;
                0142      .17     COM$DELCTRLASTP::              ; DELIVER CONTROL ASTS BY PID
                0142      .18     PUSHR  #^M<R2,R3,R5,R6>        ;
                0142      .19     CTRL2:
                0142      .20     CLRL   R2                    ; INIT RETURN CHARACTER
                0142      .21     MOVL   R4,R1                  ; LIST HEAD ADDRESS
                0142      .22     10$:
                0142      .23     MOVL   (R1),R1                ; ADDRESS FIRST BLOCK
                0142      .24     BEQL   50$                    ; NO MORE
                0142      .25     SUBL3  #TAST$L_FLINK,R1,R5     ; ADDRESS START OF BLOCK
                0142      .26     BBC   4(SP),TAST$L_MASK(R5),10$ ; SKIP IF CHARACTER NOT IN MASK
                0142      .27     ;
                0142      .28     ;
                0142      .29     ;
                0142      .30     ;
                0142      .31     ;
                0142      .32     ;
                0142      .33     ;
                0142      .34     ;
                0142      .35     ;
                0142      .36     ;
                0142      .37     ;
                0142      .38     ;
                0142      .39     ;
                0142      .40     ;
                0142      .41     ;
                0142      .42     ;
                0142      .43     ;
                0142      .44     ;
                0142      .45     ;
                0142      .46     ;
                0142      .47     ;
                0142      .48     ;
                0142      .49     ;
                0142      .50     ;
                0142      .51     ;
                0142      .52     ;
                0142      .53     ;
                0142      .54     ;
                0142      .55     ;
                0142      .56     ;
                0142      .57     ;
                0142      .58     ;
                0142      .59     ;
                0142      .60     ;
                0142      .61     ;
                0142      .62     ;
                0142      .63     ;
                0142      .64     ;
                0142      .65     ;
                0142      .66     ;
                0142      .67     ;
                0142      .68     ;
                0142      .69     ;
                0142      .70     ;
                0142      .71     ;
                0142      .72     ;
                0142      .73     ;
                0142      .74     ;
                0142      .75     ;
                0142      .76     ;
                0142      .77     ;
                0142      .78     ;
                0142      .79     ;
                0142      .80     ;
                0142      .81     ;
                0142      .82     ;
                0142      .83     ;
                0142      .84     ;
                0142      .85     ;
                0142      .86     ;
                0142      .87     ;
                0142      .88     ;
                0142      .89     ;
                0142      .90     ;
                0142      .91     ;
                0142      .92     ;
                0142      .93     ;
                0142      .94     ;
                0142      .95     ;
                0142      .96     ;
                0142      .97     ;
                0142      .98     ;
                0142      .99     ;
                0142      100     ;
                0142      101     ;
                0142      102     ;
                0142      103     ;
                0142      104     ;
                0142      105     ;
                0142      106     ;
                0142      107     ;
                0142      108     ;
                0142      109     ;
                0142      110     ;
                0142      111     ;
                0142      112     ;
                0142      113     ;
                0142      114     ;
                0142      115     ;
                0142      116     ;
                0142      117     ;
                0142      118     ;
                0142      119     ;
                0142      120     ;
                0142      121     ;
                0142      122     ;
                0142      123     ;
                0142      124     ;
                0142      125     ;
                0142      126     ;
                0142      127     ;
                0142      128     ;
                0142      129     ;
                0142      130     ;
                0142      131     ;
                0142      132     ;
                0142      133     ;
                0142      134     ;
                0142      135     ;
                0142      136     ;
                0142      137     ;
                0142      138     ;
                0142      139     ;
                0142      140     ;
                0142      141     ;
                0142      142     ;
                0142      143     ;
                0142      144     ;
                0142      145     ;
                0142      146     ;
                0142      147     ;
                0142      148     ;
                0142      149     ;
                0142      150     ;
                0142      151     ;
                0142      152     ;
                0142      153     ;
                0142      154     ;
                0142      155     ;
                0142      156     ;
                0142      157     ;
                0142      158     ;
                0142      159     ;
                0142      160     ;
                0142      161     ;
                0142      162     ;
                0142      163     ;
                0142      164     ;
                0142      165     ;
                0142      166     ;
                0142      167     ;
                0142      168     ;
                0142      169     ;
                0142      170     ;
                0142      171     ;
                0142      172     ;
                0142      173     ;
                0142      174     ;
                0142      175     ;
                0142      176     ;
                0142      177     ;
                0142      178     ;
                0142      179     ;
                0142      180     ;
                0142      181     ;
                0142      182     ;
                0142      183     ;
                0142      184     ;
                0142      185     ;
                0142      186     ;
                0142      187     ;
                0142      188     ;
                0142      189     ;
                0142      190     ;
                0142      191     ;
                0142      192     ;
                0142      193     ;
                0142      194     ;
                0142      195     ;
                0142      196     ;
                0142      197     ;
                0142      198     ;
                0142      199     ;
                0142      200     ;
                0142      201     ;
                0142      202     ;
                0142      203     ;
                0142      204     ;
                0142      205     ;
                0142      206     ;
                0142      207     ;
                0142      208     ;
                0142      209     ;
                0142      210     ;
                0142      211     ;
                0142      212     ;
                0142      213     ;
                0142      214     ;
                0142      215     ;
                0142      216     ;
                0142      217     ;
                0142      218     ;
                0142      219     ;
                0142      220     ;
                0142      221     ;
                0142      222     ;
                0142      223     ;
                0142      224     ;
                0142      225     ;
                0142      226     ;
                0142      227     ;
                0142      228     ;
                0142      229     ;
                0142      230     ;
                0142      231     ;
                0142      232     ;
                0142      233     ;
                0142      234     ;
                0142      235     ;
                0142      236     ;
                0142      237     ;
                0142      238     ;
                0142      239     ;
                0142      240     ;
                0142      241     ;
                0142      242     ;
                0142      243     ;
                0142      244     ;
                0142      245     ;
                0142      246     ;
                0142      247     ;
                0142      248     ;
                0142      249     ;
                0142      250     ;
                0142      251     ;
                0142      252     ;
                0142      253     ;
                0142      254     ;
                0142      255     ;
                0142      256     ;
                0142      257     ;
                0142      258     ;
                0142      259     ;
                0142      260     ;
                0142      261     ;
                0142      262     ;
                0142      263     ;
                0142      264     ;
                0142      265     ;
                0142      266     ;
                0142      267     ;
                0142      268     ;
                0142      269     ;
                0142      270     ;
                0142      271     ;
                0142      272     ;
                0142      273     ;
                0142      274     ;
                0142      275     ;
                0142      276     ;
                0142      277     ;
                0142      278     ;
                0142      279     ;
                0142      280     ;
                0142      281     ;
                0142      282     ;
                0142      283     ;
                0142      284     ;
                0142      285     ;
                0142      286     ;
                0142      287     ;
                0142      288     ;
                0142      289     ;
                0142      290     ;
                0142      291     ;
                0142      292     ;
                0142      293     ;
                0142      294     ;
                0142      295     ;
                0142      296     ;
                0142      297     ;
                0142      298     ;
                0142      299     ;
                0142      300     ;
                0142      301     ;
                0142      302     ;
                0142      303     ;
                0142      304     ;
                0142      305     ;
                0142      306     ;
                0142      307     ;
                0142      308     ;
                0142      309     ;
                0142      310     ;
                0142      311     ;
                0142      312     ;
                0142      313     ;
                0142      314     ;
                0142      315     ;
                0142      316     ;
                0142      317     ;
                0142      318     ;
                0142      319     ;
                0142      320     ;
                0142      321     ;
                0142      322     ;
                0142      323     ;
                0142      324     ;
                0142      325     ;
                0142      326     ;
                0142      327     ;
                0142      328     ;
                0142      329     ;
                0142      330     ;
                0142      331     ;
                0142      332     ;
                0142      333     ;
                0142      334     ;
                0142      335     ;
                0142      336     ;
                0142      337     ;
                0142      338     ;
                0142      339     ;
                0142      340     ;
                0142      341     ;
                0142      342     ;
                0142      343     ;
                0142      344     ;
                0142      345     ;
                0142      346     ;
                0142      347     ;
                0142      348     ;
                0142      349     ;
                0142      350     ;
                0142      351     ;
                0142      352     ;
                0142      353     ;
                0142      354     ;
                0142      355     ;
                0142      356     ;
                0142      357     ;
                0142      358     ;
                0142      359     ;
                0142      360     ;
                0142      361     ;
                0142      362     ;
                0142      363     ;
                0142      364     ;
                0142      365     ;
                0142      366     ;
                0142      367     ;
                0142      368     ;
                0142      369     ;
                0142      370     ;
                0142      371     ;
                0142      372     ;
                0142      373     ;
                0142      374     ;
                0142      375     ;
                0142      376     ;
                0142      377     ;
                0142      378     ;
                0142      379     ;
                0142      380     ;
                0142      381     ;
                0142      382     ;
                0142      383     ;
                0142      384     ;
                0142      385     ;
                0142      386     ;
                0142      387     ;
                0142      388     ;
                0142      389     ;
                0142      390     ;
                0142      391     ;
                0142      392     ;
                0142      393     ;
                0142      394     ;
                0142      395     ;
                0142      396     ;
                0142      397     ;
                0142      398     ;
                0142      399     ;
                0142      400     ;
                0142      401     ;
                0142      402     ;
                0142      403     ;
                0142      404     ;
                0142      405     ;
                0142      406     ;
                0142      407     ;
                0142      408     ;
                0142      409     ;
                0142      410     ;
                0142      411     ;
                0142      412     ;
                0142      413     ;
                0142      414     ;
                0142      415     ;
                0142      416     ;
                0142      417     ;
                0142      418     ;
                0142      419     ;
                0142      420     ;
                0142      421     ;
                0142      422     ;
                0142      423     ;
                0142      424     ;
                0142      425     ;
                0142      426     ;
                0142      427     ;
                0142      428     ;
                0142      429     ;
                0142      430     ;
                0142      431     ;
                0142      432     ;
                0142      433     ;
                0142      434     ;
                0142      435     ;
                0142      436     ;
                0142      437     ;
                0142      438     ;
                0142      439     ;
                0142      440     ;
                0142      441     ;
                0142      442     ;
                0142      443     ;
                0142      444     ;
                0142      445     ;
                0142      446     ;
                0142      447     ;
                0142      448     ;
                0142      449     ;
                0142      450     ;
                0142      451     ;
                0142      452     ;
                0142      453     ;
                0142      454     ;
                0142      455     ;
                0142      456     ;
                0142      457     ;
                0142      458     ;
                0142      459     ;
                0142      460     ;
                0142      461     ;
                0142      462     ;
                0142      463     ;
                0142      464     ;
                0142      465     ;
                0142      466     ;
                0142      467     ;
                0142      468     ;
                0142      469     ;
                0142      470     ;
                0142      471     ;
                0142      472     ;
                0142      473     ;
                0142      474     ;
                0142      475     ;
                0142      476     ;
                0142      477     ;
                0142      478     ;
                0142      479     ;
                0142      480     ;
                0142      481     ;
                0142      482     ;
                0142      483     ;
                0142      484     ;
                0142      485     ;
                0142      486     ;
                0142      487     ;
                0142      488     ;
                0142      489     ;
                0142      490     ;
                0142      491     ;
                0142      492     ;
                0142      493     ;
                0142      494     ;
                0142      495     ;
                0142      496     ;
                0142      497     ;
                0142      498     ;
                0142      499     ;
                0142      500     ;
                0142      501     ;
                0142      502     ;
                0142      503     ;
                0142      504     ;
                0142      505     ;
                0142      506     ;
                0142      507     ;
                0142      508     ;
                0142      509     ;
                0142      510     ;
                0142      511     ;
                0142      512     ;
                0142      513     ;
                0142      514     ;
                0142      515     ;
                0142      516     ;
                0142      517     ;
                0142      518     ;
                0142      519     ;
                0142      520     ;
                0142      521     ;
                0142      522     ;
                0142      523     ;
                0142      524     ;
                0142      525     ;
                0142      526     ;
                0142      527     ;
                0142      528     ;
                0142      529     ;
                0142      530     ;
                0142      531     ;
                0142      532     ;
                0142      533     ;
                0142      534     ;
                0142      535     ;
                0142      536     ;
                0142      537     ;
                0142      538     ;
                0142      539     ;
                0142      540     ;
                0142      541     ;
                0142      542     ;
                0142      543     ;
                0142      544     ;
                0142      545     ;
                0142      546     ;
                0142      547     ;
                0142      548     ;
                0142      549     ;
                0142      550     ;
                0142      551     ;
                0142      552     ;
                0142      553     ;
                0142      554     ;
                0142      555     ;
                0142      556     ;
                0142      557     ;
                0142      558     ;
                0142      559     ;
                0142      560     ;
                0142      561     ;
                0142      562     ;
                0142      563     ;
                0142      564     ;
                0142      565     ;
                0142      566     ;
                0142      567     ;
                0142      568     ;
                0142      569     ;
                0142      570     ;
                0142      571     ;
                0142      572     ;
                0142      573     ;
                0142      574     ;
                0142      575     ;
                0142      576     ;
                0142      577     ;
                0142      578     ;
                0142      579     ;
                0142      580     ;
                0142      581     ;
                0142      582     ;
                0142      583     ;
                0142      584     ;
                0142      585     ;
                0142      586     ;
                0142      587     ;
                0142      588     ;
                0142      589     ;
                0142      590     ;
                0142      591     ;
                0142      592     ;
                0142      593     ;
                0142      594     ;
                0142      595     ;
                0142      596     ;
                0142      597     ;
                0142      598     ;
                0142      599     ;
                0142      600     ;
                0142      601     ;
                0142      602     ;
                0142      603     ;
                0142      604     ;
                0142      605     ;
                0142      606     ;
                0142      607     ;
                0142      608     ;
                0142      609     ;
                0142      610     ;
                0142      611     ;
                0142      612     ;
                0142      613     ;
                0142      614     ;
                0142      615     ;
                0142      616     ;
                0142      617     ;
                0142      618     ;
                0142      619     ;
                0142      620     ;
                0142      621     ;
                0142      622     ;
                0142      623     ;
                0142      624     ;
                0142      625     ;
                0142      626     ;
                0142      627     ;
                0142      628     ;
                0142      629     ;
                0142      630     ;
                0142      631     ;
                0142      632     ;
                0142      633     ;
                0142      634     ;
                0142      635     ;
                0142      636     ;
                0142      637     ;
                0142      638     ;
                0142      639     ;
                0142      640     ;
                0142      641     ;
                0142      642     ;
                0142      643     ;
                0142      644     ;
                0142      645     ;
                0142      646     ;
                0142      647     ;
                0142      648     ;
                0142      649     ;
                0142      650     ;
                0142      651     ;
                0142      652     ;
                0142      653     ;
                0142      654     ;
                0142      655     ;
                0142      656     ;
                0142      657     ;
                0142      658     ;
                0142      659     ;
                0142      660     ;
                0142      661     ;
                0142      662     ;
                0142      663     ;
                0142      664     ;
                0142      665     ;
                0142      666     ;
                0142      667     ;
                0142      668     ;
                0142      669     ;
                0142      670     ;
                0142      671     ;
                0142      672     ;
                0142      673     ;
                0142      674     ;
                0142      675     ;
                0142      676     ;
                0142      677     ;
                0142      678     ;
                0142      679     ;
                0142      680     ;
                0142      681     ;
                0142      682     ;
                0142      683     ;
                0142      684     ;
                0142      685     ;
                0142      686     ;
                0142      687     ;
                0142      688     ;
                0142      689     ;
                0142      690     ;
                0142      691     ;
                0142      692     ;
                0142      693     ;
                0142      694     ;
                0142      695     ;
                0142      696     ;
                0142      697     ;
                0142      698     ;
                0142      699     ;
                0142      700     ;
                0142      701     ;
                0142      702     ;
                0142      703     ;
                0142      704     ;
                0142      705     ;
                0142      706     ;
                0142      707     ;
                0142      708     ;
                0142      709     ;
                0142      710     ;
                0142      711     ;
                0142      712     ;
                0142      713     ;
                0142      714     ;
                0142      715     ;
                0142      716     ;
                0142      717     ;
                0142      718     ;
                0142      719     ;
                0142      720     ;
                0142      721     ;
                0142      722     ;
                0142      723     ;
                0142      724     ;
                0142      725     ;
                0142      726     ;
                0142      727     ;
                0142      728     ;
                0142      729     ;
                0142      730     ;
                0142      731     ;
                0142      732     ;
                0142      733     ;
                0142      734     ;
                0142      735     ;
                0142      736     ;
                0142      737     ;
                0142      738     ;
                0142      739     ;
                0142      740     ;
                0142      741     ;
                0142      742     ;
                0142      743     ;
                0142      744     ;
                0142      745     ;
                0142      746     ;
                0142      747     ;
                0142      748     ;
                0142      749     ;
                0142      750     ;
                0142      751     ;
                0142      752     ;
                0142      753     ;
                0142      754     ;
                0142      755     ;
                0142      756     ;
                0142      757     ;
                0142      758     ;
                0142      759     ;
                0142      760     ;
                0142      761     ;
                0142      762     ;
                0142      763     ;
                0142      764     ;
                0142      765     ;
                0142      766     ;
                0142      767     ;
                0142      768     ;
                0142      769     ;
                0142      770     ;
                0142      771     ;
                0142      772     ;
                0142      773     ;
                0142      774     ;
                0142      775     ;
                0142      776     ;
                0142      777     ;
                0142      778     ;
                0142      779     ;
                0142      780     ;
                0142      781     ;
                0142      782     ;
                0142      783     ;
                0142      784     ;
                0142      785     ;
                0142      786     ;
                0142      787     ;
                0142      788     ;
                0142      789     ;
                0142      790     ;
                0142      791     ;
                0142      792     ;
                0142      793     ;
                0142      794     ;
                0142      795     ;
                0142      796     ;
                0142      797     ;
                0142      798     ;
                0142      799     ;
                0142      800     ;
                0142      801     ;
                0142      802     ;
                0142      803     ;
                0142      804     ;
                0142      805     ;
                0142      806     ;
                0142      807     ;
                0142      808     ;
                0142      809     ;
                0142      810     ;
                0142      811     ;
                0142      812     ;
                0142      813     ;
                0142      814     ;
                0142      815     ;
                0142      816     ;
                0142      817     ;
                0142      818     ;
                0142      819     ;
                0142      820     ;
                0142      821     ;
                0142      822     ;
                0142      823     ;
                0142      824     ;
                0142      825     ;
                0142      826     ;
                0142      827     ;
                0142      828     ;
                0142      829     ;
                0142      830     ;
                0142      831     ;
                0142      832     ;
                0142
```

```

;JLV0272      52  4000 8F  A8  0175  .15      BISW  #TAST$M_ABO,R2      ; SET ABORT
;JLV0272      0D  11  017A  .16      BRB   15$                ;
;JLV0272      01  E1  017C  .17 12$:   BBC   #TAST$V_INCLUDE,-
;JLV0272      08 2D  A5  017E  .19      TAST$B_CTRL(R5),15$    ; SKIP IF STRIP CHARACTER
;JLV0272      52  04  AE  9A  0181  .20      MOVZBL 4(SP),R2        ; CHARACTER NOT STRIPPED
;JLV0272      00 52  0F  E2  0185  .21      BBSS  #TAST$V_INC,R2,15$ ; MARK CHARACTER PRESENT
;JLV0272      03  E2  0189  .22 15$:   BBSS  #TAST$V_BUSY,-
;JLV0272      C5 2D  A5  018B  .24      TAST$B_CTRL(R5),10$    ; SKIP IF IN USE/ MARK IN USE
;JLV0272      -5  24 A5  04  AE  9A  018E  373  MOVZBL 4(SP),TAST$L_ASTPRM(R5) ; RETURN CHARACTER TO USER
;              0193  374 20$:
;              0B  A5  06  90  0193  375      MOVVB  #IPL$_QUEUEAST,ACB$B_RMOD(R5) ; INSERT FORK IPL,TO ALLOW USE AS
;              0197  376      ; FORK BLOCK
;              B9  AF  9F  0197  377      PUSHAB B^10$                ; SET UP RETURN ADDRESS
;              019A  378      FORK                ; CREATE FORK PROCESS
;              01A0  379 ;
;              01A0  380 ; AST QUEUE FORK PROCESS
;              01A0  381 ;
;              01A0  382      ASSUME TAST$L_ASTPRM EQ <TAST$L_AST + 4>
;              01A0  383      ASSUME ACB$L_ASTPRM EQ <ACB$L_AST + 4>
;JLV0272      10 A5  20  A5  7D  01A0  .1 22$: MOVQ  TAST$L_AST(R5),ACB$L_AST(R5) ; REARRANGE ENTRIES
;JLV0272      30  88  01A5  .2      BISB  #<ACB$M_NODELETE!ACB$M_PKAST>,-
;JLV0272      2C  A5  01A7  .3      TAST$B_RMOD(R5)        ; PK AST, NODELETE
;JLV0272      0B A5  2C  A5  90  01A9  .4      MOVVB TAST$B_RMOD(R5),ACB$B_RMOD(R5) ; MODE
;JLV0272      0C A5  28  A5  D0  01AE  .5      MOVL  TAST$L_PID(R5),ACB$L_PID(R5)   ; PID
;JLV0272      18 A5  CA  AF  DE  01B3  .6      MOVAL B^ASTACNTNG,ACB$L_KAST(R5)    ; ADDRESS OF PIGGY-BACK KERNEL ROUTI
;JLV0272      52  01  9A  01B8  .7      MOVZBL #PRI$_IOCOM,R2              ; SET UP PRIORITY INCREMENT
;JLV0272      00000000'GF 17 01BB  .8      JMP   G^SCH$QAST                ; QUEUE THE AST
;JLV0272      04  AE  52  D0  01C1  .9 50$: MOVL  R2,4(SP)                ; RETURN ANY CHARACTER
;JLV0272      006C 8F  BA  01C5  .11     POPR  #^M<R2,R3,R5,R6>            ; RESTORE REGISTERS
;JLV0272      05  01C9  395     RSB
;              01CA  396 ;
;              01CA  397 ; PIGGY-BACK KERNEL MODE ROUTINE TO:
;              01CA  398 ; 1. MARK TAST AVAILABLE FOR USE
;              01CA  399 ; 2. DEALLOCATE "LOST" BLOCK(S) AND RETURN THEIR QUOTA
;              01CA  400 ;
;              01CA  401 ;
;              01CA  402 ASTACNTNG:
;              2D  A5  08  8A  01CA  403     BICB  #TAST$M_BUSY, TAST$B_CTRL(R5) ; Mark block available.
;              15 2D  A5  04  E1  01CE  404     BBC   #TAST$V_LOST, -          ; Branch if still using this block.
;              01D3  405     TAST$B_CTRL(R5), 150$ ; Otherwise,
;              50  28  A5  3C  01D3  406     MOVZWL TAST$L_PID(R5), R0        ; use PID index to locate PCB for
;              00000000'FF40 D0 01D7  407     MOVL  @L^SCH$GL_PCBVEC[R0], R0 ; this process.
;              50  01DE
;              38  A0  B6  01DF  408     INCW  PCB$W_ASTCNT(R0)        ; Then return AST quota for and
;              50  55  D0  01E2  409     MOVL  R5,R0                ; deallocate this block.
;              FE18' 30 01E5  410     BSBW  EXE$DEANONPAGED
;              05  01E8  411 150$: RSB                ; That completes piggy-back accounting.

```



```

01E9 413          .SBTTL  COM$FLUSHCTRLS - FLUSH CONTROL AST LIST
01E9 414 ;++
01E9 415 ; COM$FLUSHCTRLS - FLUSH CONTROL AST LIST
01E9 416 ;
01E9 417 ; THIS ROUTINE IS USED BY THE TERMINAL SERVICES TO FLUSH
01E9 418 ; THE CONTROL AST LIST. THIS IS DONE AT CANCEL I/O TIME AND WHEN A
01E9 419 ; QIO SPECIFIES A 0 AST ADDRESS ON A SET AST FUNCTION.
01E9 420 ; THE SUMMARY MASK POINTED TO BY R2 IS UPDATED.
01E9 421 ;
01E9 422 ; INPUTS:
01E9 423 ;
01E9 424 ;           R2 = ADDRESS OF SUMMARY MASK
01E9 425 ;           R4 = PCB ADDRESS
01E9 426 ;           R5 = UCB ADDRESS OF RELATED UNIT
01E9 427 ;           R6 = CHANNEL NUMBER
01E9 428 ;           R7 = LIST HEAD
01E9 429 ;
01E9 430 ; OUTPUTS:
01E9 431 ;
01E9 432 ;           R0 = SS$ NORMAL
01E9 433 ;           R1 AND R7 ARE DESTROYED.
01E9 434 ;
01E9 435 ;--
01E9 436 COM$FLUSHCTRLS::
01E9 437          DSBINT  UCB$B_DIPL(R5)          ; FLUSH CONTROL AST LIST
01E9 438          CLRL   R1                      ; DISABLE INTERRUPTS
01E9 439 5$:      MOVL   (R7),R0                 ; INIT RETURN MASK
01E9 440          BEQL   50$                     ; GET LIST ENTRY
01E9 441          SUBL   #TAST$L_FLINK,R0        ; IF EQL THEN DONE
01E9 442          Cmpl   PCB$L_PID(R4),TAST$L_PID(R0); COMPUTE START OF BLOCK
01E9 443          BNEQ   40$                     ; PID MATCH?
01E9 444          CMPW   R6,TAST$W_CHAN(R0)      ; IF NEQ THEN NO
01E9 445          BNEQ   40$                     ; CHANNEL MATCH?
01E9 446          MOVL   TAST$L_FLINK(R0),(R7)   ; IF NEQ THEN NO
01E9 447          ; CLOSE UP LIST TO REMOVE ENTRY
01E9 448          BBC    #TAST$V_BUSY,TAST$B_CTRL(R0),20$; BLOCK CAN BE DELETED NOW?
01E9 449          BISB   #TAST$M_LOST,TAST$B_CTRL(R0); NO, FLAG IT FOR LATER DELETION
01E9 450          BRB    5$                       ; THAT'S ALL WE CAN DO RIGHT NOW
01E9 451          ;
01E9 452 20$:     JSB    G^COM$DRVDEALMEM        ; DEALLOCATE THE BLOCK
01E9 453          INCW   PCB$W_ASTCNT(R4)        ; RESTORE AST QUOTA
01E9 454          BRB    5$                       ; CONTINUE
01E9 455          ;
01E9 456 40$:     BISL   TAST$L_MASK(R0),R1      ; OR IN ACTIVE CONTROL CHARACTERS
01E9 457          MOVAL  TAST$L_FLINK(R0),R7     ; LOOK TO NEXT ENTRY
01E9 458          BRB    5$                       ; CONTINUE
01E9 459          ;
01E9 460 50$:     MOVL   R1,(R2)                 ; UPDATE SUMMARY MASK
01E9 461          ENBINT ; REENABLE INTERRUPTS
01E9 462          MOVZBL #SS$_NORMAL,R0         ; SET NORMAL RETURN
01E9 463          RSB

```

```
0235 465 .SBTTL COM$SETCTRLAST - SET UP CONTROL AST
0235 466 ;++
0235 467 ; COM$SETCTRLAST - SET CONTROL AST
0235 468 ;
0235 469 ; FUNCTIONAL DESCRIPTION:
0235 470 ;
0235 471 ; This routine is used by the terminal services FDT routines during the
0235 472 ; processing of requests to enable out-of-band ASTs. The current list is
0235 473 ; scanned for a block with a requestor matching that of the request being
0235 474 ; processed. If a match is found, the specified mask and AST routine address
0235 475 ; in the matching block are replaced. If no match is found, COM$BLDCTRLAST is
0235 476 ; called to create a new block. The summary mask pointed to by R2 is updated
0235 477 ; to be the inclusive or of all masks in the control AST list pointed to by R7.
0235 478 ;
0235 479 ; If either the AST routine address or the out-of-band enable mask --
0235 480 ; @<P2(AP)+4> -- is zero, all out-of-band AST requests entered by this
0235 481 ; requestor are flushed from queue pointed to by R7.
0235 482 ;
0235 483 ; INPUTS:
0235 484 ;
0235 485 ; R2 = CURRENT SUMMARY MASK ADDRESS
0235 486 ; R3 = I/O PACKET ADDRESS
0235 487 ; R4 = CURRENT PCB
0235 488 ; R5 = UCB ADDRESS
0235 489 ; R7 = ADDRESS OF THE CONTROL AST LIST HEAD TO CHANGE
0235 490 ; AP = ADDRESS OF THE QIO ARGLIST
0235 491 ;
0235 492 ; P1(AP) = ADDRESS OF AST ROUTINE TO CALL WHEN OUT-OF-BAND CHARACTER IS
0235 493 ; TYPED (0 ==> FLUSH QUEUE)
0235 494 ; P2(AP) = ADDRESS OF SHORT FORM TERMINATOR MASK GIVING WHICH OUT-OF-
0235 495 ; BAND CHARACTERS WILL PRECIPITATE DELIVERY OF AN AST
0235 496 ; (0 MASK VALUE ==> FLUSH QUEUE; This address will be passed
0235 497 ; as the AST parameter when the AST is delivered)
0235 498 ; P3(AP) = ACCESS MODE IN WHICH THE AST IS TO BE DELIVERED
0235 499 ; (This is maximized against the caller's access mode)
0235 500 ;
0235 501 ; IPL at entry is assumed to be IPL$_ASTDEL.
0235 502 ;
0235 503 ; OUTPUTS:
0235 504 ;
0235 505 ; R0 = STATUS OF THE I/O
0235 506 ; (SS$_NORMAL only; all others return via EXE$ABORTIO)
0235 .1 ; R2 = ADDRESS OF TAST BLOCK
0235 507 ; R3 = I/O PACKET ADDRESS
0235 508 ; R4 = CURRENT PCB ADDRESS
0235 509 ; R5 = UCB ADDRESS
0235 510 ; NO OTHER REGISTERS ARE PRESERVED.
0235 511 ;
0235 512 ; IPL at exit is the same as IPL at entry.
0235 513 ;
0235 514 ; The summary mask pointed to by R2 is updated to be the inclusive or of
0235 515 ; all masks in the control AST list pointed to by R7.
0235 516 ;
0235 517 ; The control AST list pointed to by R7 is flushed, extended, or an
0235 518 ; entry in the list is updated.
0235 519 ;
0235 520 ; COMPLETION CODES:
```

;JLV0272

```

0235 521 ;
0235 522 ;          SSS_NORMAL
0235 523 ;
0235 524 ;          The following return codes are reported by a JMP to EXE$ABORTIO:
0235 525 ;          SSS_ACCVIO  -- specified mask not accessible
0235 526 ;          SSS_EXQUOTA -- buffered I/O or AST quota failure
0235 527 ;          SSS_INSUFMEM -- dynamic memory failure
0235 528 ;--
0235 529
0235 530 ; COM$SETCTRLAST, COM$BLDCTRLAST, and COM$FILLCTRLAST build and use the
0235 531 ; following information block. COM$SETCTRLAST builds the block on the stack
0235 532 ; and points R11 to it. COM$BLDCTRLAST and COM$FILLCTRLAST use information
0235 533 ; stored to build or update a TAST control block.
0235 534
0235 535          $OFFSET 0, POSITIVE, <-
0235 536          ASTROUT, -                ; User's AST routine address
0235 537          ASTPARAM, -              ; AST routine parameter
0235 538          USRMSKADR, -           ; Pointer to user's OOB mask
0235 539          CURMSKADR, -           ; Pointer to current summary mask
0235 540          <TINFOFSIZE, 0> -        ; Size of the block
0235 541          >
0000          ASTROUT:
0004          ASTPARAM:
0008          USRMSKADR:
000C          CURMSKADR:
0010          TINFOFSIZE:
0235 542
0235 543
0235 544 COM$SETCTRLAST::                ; SET UP CONTROL AST
0235 545
56 28 A3 3C 0235 546          MOVZWL IRP$W_CHAN(R3), R6        ; Get packet channel number.
58 6C D0 0239 547          MOVL P1(AP), R8                ; Get user AST address.
          AB 13 023C 548          BEQL COM$FLUSHCTRLS        ; If its zero, flush OOB requests.
59 04 AC D0 023E 549          MOVL P2(AP), R9                ; Get address of user OOB mask.
          50 0C 3C 0242 .1          MOVZWL #SS$_ACCVIO, R0      ; Assume no access
          0245 .2          IFNORD #8, (R9), 5$              ; Probe for read access.
          50 14 3C 024B .3          MOVZWL #SS$_BADPARAM, R0    ; Assume invalid mask format
          69 D5 024E .4          TSTL (R9)                  ; Verify short form terminator
          08 12 0250 .5          BNEQ 5$                    ; mask format.
          04 A9 D5 0252 .6          TSTL 4(R9)                ; Any OOB characters specified?
          09 12 0255 .7          BNEQ 7$                    ; If there are some, proceed.
          FF8F 31 0257 .8          BRW COM$FLUSHCTRLS        ; Otherwise, flush OOB requests.
          00000000'GF 17 025A .9 5$:          JMP G^EXE$ABORTIO    ; I/O failed, abort request
          0260 560 7$:
          0260 561
          0260 562 ;
          0260 563 ; We are now holding a valid request of enable an out-of-band character AST
          0260 564 ;
          0260 565
          0260 566          ASSUME ASTPARAM EQ <ASTROUT + 4>
          5E FO AE 9E 0260 567          MOVAB -TINFOFSIZE(SP), SP      ; Allocate info. block on the stack.
          5B 5E D0 0264 568          MOVL SP, R11                ; Save info. block address.
          6B 58 7D 0267 569          MOVQ R8, ASTROUT(R11)        ; Save AST routine and parameter info.
          08 AB 04 A9 DE 026A 570          MOVAL 4(R9), USRMSKADR(R11)    ; Save address of user OOB mask.
          OC AB 52 D0 026F 571          MOVL R2, CURMSKADR(R11)    ; Save current summary mask address.
          58 7C 0273 572          CLRQ R8                    ; Clear accumulation summary mask and
          0275 573          ; a local flags longword.

```

```

;JLV0272          5A  D4  0275  .1      CLRL  R10          ; TAST address
                  0277  574      DSBINT UCB$B_DIPL(R5) ; Interlock queue access.
                  027E  575
                52  67  D0  027E  576 10$:  MOVL  (R7), R2          ; Get list entry.
                  36  13  0281  577      BEQL  20$             ; Branch if no more entries in list.
                52  E4  A2  9E  0283  578      MOVAB -TAST$L_FLINK(R2), R2 ; Compute start of block.
                28  A2  60  A4  D1  0287  579      CMLP  PCB$L_PID(R4), TAST$L_PID(R2) ; Do the PIDs match?
                  21  12  028C  580      BNEQ  15$             ; Branch if PIDs don't match.
                2E  A2  56  B1  028E  581      CMPW  R6, TAST$W_CHAN(R2) ; Is the channel right?
                  1B  12  0292  582      BNEQ  15$             ; Branch if not the right channel.
                  0294  583
RKS006          OD  2D  A2  03  E1  0294  .1      BBC   #TAST$V_BUSY,TAST$B_CTRL(R2),12$; Block can be reused now
RKS006          2D  A2  10  88  0299  .2      BISB  #TAST$M_LOST,TAST$B_CTRL(R2) ; No, flag it for later deletion
RKS006          67  1C  A2  D0  029D  .3      MOVL  TAST$L_FLINK(R2),(R7) ; Remove the entry from the queue
RKS006          59  02  C8  02A1  .4      BISL  #2, R9          ; Indicate that a match was found
RKS006          13  11  02A4  .5      BRB   20$             ; And build a new one, since the old
RKS006          02A6  .6                  ; one is busy and will be deleted when
RKS006          02A6  .7                  ; the AST is delivered.
RKS006          5A  52  D0  02A6  .8 12$:  MOVL  R2,R10          ; Save TAST address
                  006D  30  02A9  584      BSBW  COM$FILLCTRLAST ; Update matching TAST control block.
                  59  01  C8  02AC  585      BISL  #1, R9          ; Indicate that an update was done.
                  02AF  586
                58  30  A2  C8  02AF  587 15$:  BISL  TAST$L_MASK(R2), R8 ; Accumulate OOB mask data for queue.
                57  1C  A2  DE  02B3  588      MOVAL TAST$L_FLINK(R2), R7 ; Advance to next queue entry.
                  C5  11  02B7  589      BRB   10$             ; Loop until entire queue processed.
                  02B9  590
                OC  BB  58  D0  02B9  591 20$:  MOVL  R8, @CURMSKADR(R11) ; Make accumulated OOB mask for queue
                  02BD  592                  ; the current OOB mask for the queue.
                  02BD  593                  ; Return to caller's IPL.
                  50  01  3C  02C0  594      ENBINT #SS$_NORMAL, R0 ; Assume work completed successfully.
RKS006          52  5A  D0  02C3  .1      MOVL  R10,R2         ; Valid only if R9 LBS
RKS006          02  59  E8  02C6  .2      BLBS  R9, 30$       ; If no blocks in the queue were
RKS006          0E  10  02C9  .3      BSBB  COM$BLDCTRLAST ; updated, build a new block.
RKS006          5E  10  AE  9E  02CB  .4 30$:  MOVAB TINFOSIZE(SP), SP ; Restore stack.
RKS006          01  50  E9  02CF  .5      BLBC  R0, 50$       ; Was COM$BLDCTRLAST successful?
RKS006          05  02D2  .6      RSB   ; if so, return to caller.
RKS006          02D3  .7 50$:
RKS006          0000000'GF 17 02D3  .8      JMP   GAEXE$ABORTIO ; If it failed, abort I/O request.
RKS006          02D9  .9

```

```
02D9 602 .SBTTL COM$BLDCTRLAST - BUILD CONTROL AST
02D9 603 ;++
02D9 604 ; COM$BLDCTRLAST - BUILD NEW CONTROL AST PACKET
02D9 605 ;
02D9 606 ; FUNCTIONAL DESCRIPTION:
02D9 607 ;
02D9 608 ; This routine builds a new terminal AST control block and inserts it after
02D9 609 ; the TAST entry pointed to by R7. The control block will double as an AST
02D9 610 ; control block when a AST is delivered. The block will be reused until the
02D9 611 ; out-of-band AST request is canceled. The summary mask pointed to by
02D9 612 ; CURMSKADR(R11) is inclusively or'ed with the user out-of-band mask pointed
02D9 613 ; to by USRMSKADR(R11).
02D9 614 ;
02D9 615 ; INPUTS:
02D9 616 ;
02D9 617 ; R3 = I/O_PACKET ADDRESS
02D9 618 ; R4 = CURRENT PCB
02D9 619 ; R5 = UCB ADDRESS
02D9 620 ; R6 = CHANNEL ON WHICH OOB REQUEST IS BEING MADE
02D9 621 ; R7 = ADDRESS OF THE CONTROL AST LIST ENTRY PRECEDING THE POINT WHERE
02D9 622 ; THE NEW ENTRY IS TO BE ADDED.
02D9 623 ; R11= THE ADDRESS OF A TAST INFORMATION LIST (SEE COM$SETCTRLAST)
02D9 624 ; AP = ADDRESS OF THE QIO ARGUMENT LIST
02D9 625 ;
02D9 626 ; P3(AP) = ACCESS MODE IN WHICH THE AST IS TO BE DELIVERED
02D9 627 ; (This is maximized against the caller's access mode)
02D9 628 ;
02D9 629 ; ASTROUT(R11) = ADDRESS OF AST ROUTINE TO CALL WHEN OUT-OF-BAND
02D9 630 ; CHARACTER IS TYPED
02D9 631 ; ASTPARAM(R11) = AST PARAMETER VALUE TO BE PASSED TO AST ROUTINE WHEN
02D9 632 ; OUT-OF-BAND AST IS DELIVERED
02D9 633 ; CURMSKADR(R11)= ADDRESS OF THE CURRENT OUT-OF-BAND SUMMARY MASK
02D9 634 ; USRMSKADR(R11)= ADDRESS OF OUT-OF-BAND MASK SPECIFIED BY USER FOR
02D9 635 ; THIS AST ENABLE
02D9 636 ;
02D9 637 ; IPL at entry is assumed to be IPL$_ASTDEL.
02D9 638 ;
02D9 639 ; OUTPUTS:
02D9 640 ;
02D9 641 ; R0 = STATUS OF THE I/O
02D9 642 ; R1 & R2 DESTROYED
02D9 643 ; ALL OTHER REGISTERS PRESERVED
02D9 644 ;
02D9 645 ; IPL at entry is assumed to be IPL$_ASTDEL.
02D9 646 ;
02D9 647 ; A TAST control block is allocated, filled in, and linked after the
02D9 648 ; entry pointed to by R7
02D9 649 ;
02D9 650 ; COMPLETION CODES:
02D9 651 ;
02D9 652 ; SS$_NORMAL
02D9 653 ; SS$_EXQUOTA -- BUFFERED I/O OR AST QUOTA FAILURE
02D9 654 ; SS$_INSUFMEM -- DYNAMIC MEMORY FAILURE
02D9 655 ;--
02D9 656
02D9 657 COM$BLDCTRLAST:
02D9 658
```

```

38 A4 B5 02D9 659 TSTW PCB$W_ASTCNT(R4) ; Is there enough AST quota?
31 15 02DC 660 BLEQ 91$ ; Branch if insufficient AST quota.
38 A4 B7 02DE 661 DECW PCB$W_ASTCNT(R4) ; Deduct from AST quota.
53 DD 02E1 662 PUSHL R3 ; Save reg. destroyed by EXE$ALLOCBUF.
51 34 9A 02E3 663 MOVZBL #TAST$C_LENGTH, R1 ; Set size of TAST block.
00000000 GF 16 02E6 664 JSB G^EXE$ALLOCBUF ; Allocate for the TAST control block.
53 8ED0 02EC 665 POPL R3 ; Restore saved register.
21 50 E9 02EF 666 BLBC R0, 93$ ; Branch if allocation failed.
25 10 02F2 667 BSBF COM$FILLCTRLAST ; Fill in newly allocated block.
02F4 668
02F4 669 ;
02F4 670 ; INSERT NEWLY BUILD TAST CONTROL BLOCK
02F4 671 ;
02F4 672
02F4 673 DSBINT UCB$B_DIPL(R5) ; Interlock access to queue links.
1C A2 67 D0 02FB 674 MOVL (R7), TAST$L_FLINK(R2) ; Move list for. pointer to new entry.
67 1C A2 DE 02FF 675 MOVAL TAST$L_FLINK(R2), (R7) ; Link new entry to current list.
OC BB 30 A2 C8 0303 676 BISL TAST$L_MASK(R2), @CURMSKADR(R11) ; Update summary mask.
0308 677 ENBINT ; Restore previous IPL.
50 01 3C 030B 678 MOVZWL #SS$_NORMAL, R0 ; Indicate that build succeeded.
05 030E 679 RSB ; Return to caller.
030F 680
030F 681 ;
030F 682 ; ERROR RETURNS:
030F 683 ;
030F 684
50 1C 3C 030F 685 91$: MOVZWL #SS$_EXQUOTA, R0 ; AST quota exceeded.
05 0312 686 RSB
0313 687
50 0124 8F 3C 0313 688 93$: MOVZWL #SS$_INSFMEM, R0 ; Insufficient dynamic memory.
05 0318 689 RSB

```

```

0319 691          .SBTTL COM$FILLCTRLAST - FILLIN A CONTROL AST CONTROL BLOCK
0319 692 ;++
0319 693 ; COM$FILLCTRLAST - FILLIN A CONTROL AST CONTROL BLOCK
0319 694 ;
0319 695 ; FUNCTIONAL DESCRIPTION:
0319 696 ;
0319 697 ; This routine fills in the terminal AST control block pointed to by R2. The
0319 698 ; block may be either a previously allocated block which is already linked to
0319 699 ; a control AST queue, or a newly allocated block which is being filled in for
0319 700 ; the first time.
0319 701 ;
0319 702 ; INPUTS:
0319 703 ;
0319 704 ;         R2 = ADDRESS OF TAST CONTROL BLOCK TO BE FILLED IN
0319 705 ;         R3 = I/O PACKET ADDRESS
0319 706 ;         R4 = CURRENT PCB
0319 707 ;         R6 = CHANNEL ON WHICH OOB AST REQUEST IS BEING MADE
0319 708 ;         R11= THE ADDRESS OF A TAST INFORMATION LIST (SEE COM$SETCTRLAST)
0319 709 ;         AP = ADDRESS OF THE QIO ARGLIST
0319 710 ;
0319 711 ;         P3(AP) = ACCESS MODE IN WHICH THE AST IS TO BE DELIVERED
0319 712 ;                 (This is maximized against the caller's access mode)
0319 713 ;
0319 714 ;         ASTROUT(R11) = ADDRESS OF AST ROUTINE TO CALL WHEN OUT-OF-BAND
0319 715 ;                 CHARACTER IS TYPED
0319 716 ;         ASTPARAM(R11) = AST PARAMETER VALUE TO BE PASSED TO AST ROUTINE WHEN
0319 717 ;                 OUT-OF-BAND AST IS DELIVERED
0319 718 ;         USRMSKADR(R11)= ADDRESS OF OUT-OF-BAND MASK SPECIFIED BY USER FOR
0319 719 ;                 THIS AST ENABLE
0319 720 ;
0319 721 ;         If this routine is called to operate on a TAST block which is already
0319 722 ;         linked to a control queue, it should be called at device IPL.
0319 723 ;         Otherwise, it can be called at IPL$_ASTDEL.
0319 724 ;
0319 725 ; OUTPUTS:
0319 726 ;
0319 727 ;         R0 & R1 ARE DESTROYED.
0319 728 ;         ALL OTHER REGISTERS ARE PRESERVED.
0319 729 ;
0319 730 ;         IPL at exit is the same as IPL at entry.
0319 731 ;
0319 732 ;         The TAST control block pointed to by R2 is filled in.
0319 733 ;
0319 734 ; COMPLETION CODES:
0319 735 ;
0319 736 ;         There is no completion status. This routine is always successful.
0319 737 ;--
0319 738
0319 739 COM$FILLCTRLAST:
0319 740
0319 741         ASSUME TAST$_ASTPRM EQ <TAST$_AST + 4>
0319 742         ASSUME ASTPARAM EQ <ASTROUT + 4>
0319 743         MOVQ   ASTROUT(R11), -           ; Plant AST routine and
0319 744         TAST$_AST(R2)           ; parameter addresses.
0319 745         MOVL   @USRMSKADR(R11), -
0319 746         TAST$_MASK(R2)         ; Plant OOB mask.
0319 747         CLRL   R0               ; Assume no flags
0319 748
0319 749
0319 750
0319 751
0319 752
0319 753
0319 754
0319 755
0319 756
0319 757
0319 758
0319 759
0319 760
0319 761
0319 762
0319 763
0319 764
0319 765
0319 766
0319 767
0319 768
0319 769
0319 770
0319 771
0319 772
0319 773
0319 774
0319 775
0319 776
0319 777
0319 778
0319 779
0319 780
0319 781
0319 782
0319 783
0319 784
0319 785
0319 786
0319 787
0319 788
0319 789
0319 790
0319 791
0319 792
0319 793
0319 794
0319 795
0319 796
0319 797
0319 798
0319 799
0319 800
0319 801
0319 802
0319 803
0319 804
0319 805
0319 806
0319 807
0319 808
0319 809
0319 810
0319 811
0319 812
0319 813
0319 814
0319 815
0319 816
0319 817
0319 818
0319 819
0319 820
0319 821
0319 822
0319 823
0319 824
0319 825
0319 826
0319 827
0319 828
0319 829
0319 830
0319 831
0319 832
0319 833
0319 834
0319 835
0319 836
0319 837
0319 838
0319 839
0319 840
0319 841
0319 842
0319 843
0319 844
0319 845
0319 846
0319 847
0319 848
0319 849
0319 850
0319 851
0319 852
0319 853
0319 854
0319 855
0319 856
0319 857
0319 858
0319 859
0319 860
0319 861
0319 862
0319 863
0319 864
0319 865
0319 866
0319 867
0319 868
0319 869
0319 870
0319 871
0319 872
0319 873
0319 874
0319 875
0319 876
0319 877
0319 878
0319 879
0319 880
0319 881
0319 882
0319 883
0319 884
0319 885
0319 886
0319 887
0319 888
0319 889
0319 890
0319 891
0319 892
0319 893
0319 894
0319 895
0319 896
0319 897
0319 898
0319 899
0319 900
0319 901
0319 902
0319 903
0319 904
0319 905
0319 906
0319 907
0319 908
0319 909
0319 910
0319 911
0319 912
0319 913
0319 914
0319 915
0319 916
0319 917
0319 918
0319 919
0319 920
0319 921
0319 922
0319 923
0319 924
0319 925
0319 926
0319 927
0319 928
0319 929
0319 930
0319 931
0319 932
0319 933
0319 934
0319 935
0319 936
0319 937
0319 938
0319 939
0319 940
0319 941
0319 942
0319 943
0319 944
0319 945
0319 946
0319 947
0319 948
0319 949
0319 950
0319 951
0319 952
0319 953
0319 954
0319 955
0319 956
0319 957
0319 958
0319 959
0319 960
0319 961
0319 962
0319 963
0319 964
0319 965
0319 966
0319 967
0319 968
0319 969
0319 970
0319 971
0319 972
0319 973
0319 974
0319 975
0319 976
0319 977
0319 978
0319 979
0319 980
0319 981
0319 982
0319 983
0319 984
0319 985
0319 986
0319 987
0319 988
0319 989
0319 990
0319 991
0319 992
0319 993
0319 994
0319 995
0319 996
0319 997
0319 998
0319 999
0319 1000

```

;JLV0272  
;JLV0272  
;JLV0272

20 A2 6B 7D  
30 A2 08 BB D0  
50 D4 0322 .3

```

;JLV0272          0324 .4      ;
;JLV0272          0324 .5      ; Check for ABORT
;JLV0272          0324 .6      ;
;JLV0272      05 20 A3 0C E1 0324 .7      BBC      #IO$V_TT_ABORT, -
;JLV0272          0329 .8      IRP$W_FUNC(R3),5$      ; Branch if not abort
;JLV0272          50 20 88 0329 .9      BISB      #TAST$M_ABORT,R0      ; Else set abort flag.
;JLV0272          08 11 032C .10     BRB      10$      ; Ignore INCLUDE
;JLV0272          032E .11     ;
;JLV0272          032E .12     ; Check for INCLUDE
;JLV0272          032E .13     ;
;JLV0272      03 20 A3 0B E1 032E .14 5$:  BBC      #IO$V_INCLUDE, -
;JLV0272          0333 .15     IRP$W_FUNC(R3), 10$      ; BR if striping 00B chars.
;JLV0272          50 02 88 0333 .16     BISB      #TAST$M_INCLUDE,R0      ; Else set no-strip flag.
;JLV0272          0336 .17     ;
;JLV0272          2D A2 50 90 0336 .18 10$:  MOVB      R0,TAST$B_CTRL(R2)      ; Set TAST control field.
;JLV0272          02 00 EF 033A .19     EXTZV     #0, #2, P3(AP), R0      ; Get requested delivery access mode.
;JLV0272          50 08 AC 033D      ;
-5          51 DC 0340 750     MOVPSL     R1      ; Get access mode of requestor.
          51 02 16 EF 0342 751     EXTZV     #PSL$V_PRVMOD, #PSL$$_PRVMOD, R1, R1 ; If requestor's access
          51 50 91 0347 752     CMPB      R0, R1      ; mode is bigger than delivery access
          50 03 18 034A 753     BGEQ      20$      ; mode, then delivery AST in
          50 51 90 034C 754     MOVB      R1, R0      ; requestor's access mode.
          2C A2 50 90 034F 755 20$:  MOVB      R0, TAST$B_RMOD(R2)      ; Plant delivery access mode.
          2E A2 56 B0 0353 756     MOVW      R6, TAST$W_CHAN(R2)      ; Plant requestor's channel.
          28 A2 60 A4 D0 0357 757     MOVL      PCB$L_PID(R4), TAST$L_PID(R2) ; Plant requestor's PID.
          05 035C 758     RSB      ; Return to caller.
          035D 759     ;
          035D 760     .END

```



ACB\$B_RMOD	=	0000000B		PSL\$\$_PRVMOD	=	00000002	
ACB\$_AST	=	00000010		PSL\$_V_PRVMOD	=	00000016	
ACB\$_ASTPRM	=	00000014		SAVABS...	=	00000010	
ACB\$_KAST	=	00000018		SCH\$GL_PCBVEC	*****	X	02
ACB\$_PID	=	0000000C		SCH\$QAST	*****	X	02
ACB\$_NODELETE	=	00000020		SS\$_ACCVIO	=	0000000C	
ACB\$_PKAST	=	00000010		SS\$_BADPARAM	=	00000014	
ACB\$_QUOTA	=	00000040		SS\$_EXQUOTA	=	0000001C	
ASTACNTNG		000001CA	R	02	SS\$_INSMEM	=	00000124
ASTPARM		00000004		SS\$_NORMAL	=	00000001	
ASTROUT		00000000		TAST\$B_CTRL	=	0000002D	
ATTN2		0000000C	R	02	TAST\$B_RMOD	=	0000002C
BUG\$_BADDALRQSZ		*****	X	02	TAST\$_C_LENGTH	=	00000034
COM\$BLDCTRLAST		000002D9	R	02	TAST\$_AST	=	00000020
COM\$DELATTNAST		00000000	RG	02	TAST\$_ASTPRM	=	00000024
COM\$DELATTNASTP		00000008	RG	02	TAST\$_FLINK	=	0000001C
COM\$DELCTRLAST		00000142	RG	02	TAST\$_MASK	=	00000030
COM\$DELCTRLASTP		0000014A	RG	02	TAST\$_PID	=	00000028
COM\$DRVDEALMEM		0000008F	RG	02	TAST\$_M_ABO	=	00004000
COM\$FILLCTRLAST		00000319	R	02	TAST\$_M_ABORT	=	00000020
COM\$FLUSHATTNS		0000004F	RG	02	TAST\$_M_BUSY	=	00000008
COM\$FLUSHCTRLS		000001E9	RG	02	TAST\$_M_INCLUDE	=	00000002
COM\$POST		00000081	RG	02	TAST\$_M_LOST	=	00000010
COM\$SETATTNAST		000000D5	RG	02	TAST\$_V_ABORT	=	00000005
COM\$SETCTRLAST		00000235	RG	02	TAST\$_V_BUSY	=	00000003
CTRL2		0000014E	R	02	TAST\$_V_INC	=	0000000F
CURMSKADR		0000000C		TAST\$_V_INCLUDE	=	00000001	
DIR...	=	00000001		TAST\$_V_LOST	=	00000004	
DYN\$_C_FRK	=	00000008		TAST\$_W_CHAN	=	0000002E	
EXE\$ABORTIO		*****	X	02	TINFO\$SIZE	=	00000010
EXE\$ALLOCBUF		*****	X	02	UCB\$_B_DIPL	=	0000005E
EXE\$DEANONPAGED		*****	X	02	UCB\$_L_OPCNT	=	00000070
EXE\$FORK		*****	X	02	USRMSKADR	=	00000008
EXE\$MAXACMODE		*****	X	02			
FKB\$_B_FIPL	=	0000000B					
FKB\$_B_TYPE	=	0000000A					
FKB\$_C_LENGTH	=	00000018					
FKB\$_W_SIZE	=	00000008					
IO\$_V_INCLUDE	=	0000000B					
IO\$_V_TT_ABORT	=	0000000C					
IOC\$GL_PSBL		*****	X	02			
IPL\$_IOPOST	=	00000004					
IPL\$_QUEUEAST	=	00000006					
IPL\$_SYNCH	=	00000008					
IRP\$_W_CHAN	=	00000028					
IRP\$_W_FUNC	=	00000020					
P1	=	00000000					
P2	=	00000004					
P3	=	00000008					
P4	=	0000000C					
P5	=	00000010					
P6	=	00000014					
PCB\$_L_PID	=	00000060					
PCB\$_W_ASTCNT	=	00000038					
PR\$_IPL	=	00000012					
PR\$_SIRR	=	00000014					
PR\$_I_IOCOM	=	00000001					

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000010 ( 16.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WIONONPAGED	0000035D ( 861.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.07	00:00:01.22
Command processing	99	00:00:00.52	00:00:06.81
Pass 1	457	00:00:19.68	00:01:34.99
Symbol table sort	0	00:00:02.98	00:00:10.33
Pass 2	150	00:00:05.11	00:00:26.99
Symbol table output	11	00:00:00.10	00:00:00.10
Psect synopsis output	2	00:00:00.03	00:00:00.73
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	750	00:00:28.49	00:02:21.17

The working set limit was 1650 pages.  
102023 bytes (200 pages) of virtual memory were used to buffer the intermediate code.  
There were 100 pages of symbol table space allocated to hold 1859 non-local and 37 local symbols.  
860 source lines were read in Pass 1, producing 15 object records in Pass 2.  
34 pages of virtual memory were used to define 32 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	18
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	29

2032 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:COMDRVSUB/OBJ=OBJ\$:COMDRVSUB MSRC\$:COMDRVSUB/UPDATE=(ENH\$:COMDRVSUB)+EXECMLS/LIB

**PCDRIVER.LIS  
PCDRIVER.COM  
PCDRIVER.OPT  
LOADER.COM  
READY.FOR  
RANDWV.FOR  
MODE.FOR**

(1)	85	External and local symbol definitions
(1)	202	Standard tables
(1)	312	PT_CONTROL_INIT, Controller initialization routine
(1)	341	PT_UNIT_INIT, Unit initialization routine
(1)	389	PT_FDT_ROUTINE, Punch FDT routine
(1)	471	ACP_FDT ACP \$QIO FDT Routine
(1)	527	NULL_BYTES FDT Routines for header/trailer
(1)	543	PR_FDT_ROUTINE Reader FDT Routine
(1)	603	PT_START, Start I/O routine
(1)	647	PT_PUNCH, Start a punch operation
(1)	905	PR_START, Start I/O routine (Reader)
(1)	1012	PT_INTERRUPT, Interrupt service routine
(1)	1068	PT_CANCEL, Cancel I/O routine
(1)	1168	PT_REG_DUMP, Device register dump routine

```
0000 1 .TITLE PCDRIVER - VAX/VMS PAPER TAPE READER/PUNCH DRIVER
0000 2 .IDENT /V06/
0000 3
0000 4 ;
0000 5 ; COPYRIGHT (C) 1979
0000 6 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
0000 7 ;
0000 8 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT
0000 9 ; NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
0000 10 ; EQUIPMENT CORPORATION.
0000 11 ;
0000 12 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 13 ; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DEC.
0000 14 ;
0000 15 ;++
0000 16 ;
0000 17 ; FACILITY:
0000 18 ;
0000 19 ; VAX/VMS Paper Tape Reader/Punch Driver
0000 20 ;
0000 21 ; ABSTRACT:
0000 22 ;
0000 23 ; This driver controls the PC11 paper tape reader/punch. READ and
0000 24 ; WRITE $QIOs are accepted (VBLK, LBLK, AND PBLK), as are MOUNT,
0000 25 ; ACPCONTROL, and WRITEOF $QIOs (which call on the ACP -- PCACP).
0000 26 ; MODIFY, DELETE, and DEACCESS $QIOs are valid for the punch,
0000 27 ; as are CREATE and READPROMPT $QIOs (valid for both reader/punch).
0000 28 ;
0000 29 ; READVBLK and READLBLK cause characters to be read from the
0000 30 ; reader with leading null bytes ignored (for the first read
0000 31 ; request); READPBLK is the same, except that leading null bytes
0000 32 ; are treated as data characters. An EOF byte is recognized if
0000 33 ; the user requested looking for EOF (default if not physical).
0000 34 ;
0000 35 ; WRITEVBLK and WRITELBLK cause characters to be punched (a
0000 36 ; header and trailer of null bytes is punched--default is to
0000 37 ; punch NULL_BYTES null bytes in the header and trailer tape).
0000 38 ; WRITEPBLK is the same, except that no header/trailer is punched.
0000 39 ; If first $QIO wasn't physical, an EOF is punched before trailer.
0000 40 ;
0000 41 ; The MOUNT $QIO is used to synchronize with the ACPMNT program
0000 42 ; which loads the paper tape punch ACP (PCACP). The ACPCONTROL
0000 43 ; $QIO is a no-op function which serves to awaken the ACP if it is
0000 44 ; hibernating. The WRITEOF $QIO is turned into an equivalent
0000 45 ; WRITEVBLK $QIO by the ACP.
0000 46 ;
0000 47 ; The MODIFY $QIO changes the default (NULL_BYTES) number of null
0000 48 ; bytes to punch in the header/trailer tape. P2 specifies the
0000 49 ; new number of null bytes to be punched.
0000 50 ;
0000 51 ; The DELETE $QIO restores the default (NULL_BYTES) number of null
0000 52 ; bytes to punch in the header/trailer tape.
0000 53 ;
0000 54 ; The DEACCESS $QIO returns the current number of null bytes which
0000 55 ; are to be punched for the header/trailer tape in the high order
0000 56 ; longword of the IOSB specified in the $QIO call.
0000 57 ;
```

```
0000 58 ; The CREATE $QIO is used to enable searching for an EOF byte (for
0000 59 ; the reader), or to enable the punching of an EOF byte (for the
0000 60 ; punch).
0000 61 ;
0000 62 ; The READPROMPT $QIO is used to disable searching for an EOF byte
0000 63 ; (for the reader), or to disable the punching of an EOF byte (for
0000 64 ; the punch).
0000 65 ;
0000 66 ; When used, P1 is the starting address of the buffer, and P2 is
0000 67 ; the buffer length. The driver performs buffered I/O.
0000 68 ;
0000 69 ; An EOF character is equal to a decimal 26 (control-Z).
0000 70 ;
0000 71 ; Note that the ACP (PCACP) is not required for the driver to
0000 72 ; work. PCDRIVER may be loaded and used independent of the ACP.
0000 73 ; The ACP's main function is to serve as an instructional tool
0000 74 ; showing how ACPs are written; the ACP does not provide any
0000 75 ; added functionality for either the reader or punch.
0000 76 ;
0000 77 ; AUTHOR:
0000 78 ;
0000 79 ; Vik Muiznieks -- July 1979
0000 80 ;
0000 81 ; HISTORY
0000 82 ; Fred Marsh
0000 83 ; Added $DYNDEF $PRVDEF and $PCBDEF to external symbol definitions
0000 84 ;--
0000 85 ;.SBTTL External and local symbol definitions
```

```
0000 87 ;
0000 88 ; External symbols
0000 89 ;
0000 90 $CRBDEF ; Channel request block
0000 91 $VCBDEF ; Volume control block
0000 92 $DCDEF ; Device classes and types
0000 93 $DDBDEF ; Device data block
0000 94 $DEVDEF ; Device characteristics
0000 95 $EMBDEF ; Error logging fields(not in template)
0000 96 $IDBDEF ; Interrupt data block
0000 97 $IODEF ; I/O function codes
0000 98 $IPLDEF ; Hardware IPL definitions
0000 99 $IRPDEF ; I/O request packet
0000 100 $JIBDEF ; Job Information Block
0000 101 $SSDEF ; System status codes
0000 102 $UCBDEF ; Unit control block
0000 103 $VECDEF ; Interrupt vector block
0000 104 $DYNDEF ;
0000 105 $PCBDEF ;
0000 106 $PRVDEF ;
0000 107
0000 108
0000 109 ;
0000 110 ; Argument list (AP) offsets for device-dependent $QIO parameters
0000 111 ; (Only P1 and P2 are used in this driver)
0000 112 ;
0000 113
00000000 0000 114 P1 = 0 ; First $QIO parameter
00000004 0000 115 P2 = 4 ; Second $QIO parameter
00000008 0000 116 P3 = 8 ; Third $QIO parameter
0000000C 0000 117 P4 = 12 ; Fourth $QIO parameter
00000010 0000 118 P5 = 16 ; Fifth $QIO parameter
00000014 0000 119 P6 = 20 ; Sixth $QIO parameter
0000 120 ;
0000 121 ; Other constants
0000 122 ;
0000 123
0000000F 0000 124 LOOP_CNT=15 ; before issuing off-line msg.
00000002 0000 125 SHORT_WAIT=2 ; # sec for device to go online
0000000C 0000 126 BUF_OVR_HD=12 ; system buffer overhead
0000000A 0000 127 PT_TIMEOUT_SEC=10 ; 10 second device timeout
00000002 0000 128 PT_NUM_REGS=2 ; Each device has 2 registers
00000064 0000 129 MY_CLASS=100 ; My own device class number
00000064 0000 130 MY_TYPE=100 ; My own device type number
0000000C 0000 131 ERBFSIZE=12 ; Size of err. log buf. in bytes
00000000 0000 132 NULL=0 ; null byte
0000001A 0000 133 EOF=26 ; EOF byte = control-Z
00000032 0000 134 NULL_BYTES=50 ; for header/trailer
00000001 0000 135 PUNCH_HDR=1 ; bit pos. flagging
0000 136 ; header/trailer punched
00000004 0000 137 START=4 ; mask for punch. hdr.
00000010 0000 138 END=16 ; mask for punch. trail.
00000008 0000 139 PNCH=8 ; flag-->punch op. perf. (mask)
00000020 0000 140 RD_HDR=32 ; flag-->read hdr. tape (mask)
00000007 0000 141 STARTING=7 ; bit pos.--> 1st operation
00000006 0000 142 FRST_QIO=6 ; flag-->read 1st header (pos.)
00000001 0000 143 BIT0=1 ; mask-->EOF byte handling
```

PCDRIVER  
V06

- VAX/VMS PAPER TAPE READER/PUNCH DRIVER 12-SEP-1984 17:59:27 VAX-11 Macro V03-01 Page 4  
External and local symbol definitions 22-AUG-1984 20:06:00 WORK2:[COURSE.DRIVER]PCDRIVER.MAR;(1)

00000100 0000 144 BIT8=256 ; mask-->EOF byte handling  
00000200 0000 145 BIT9=512 ; mask-->EOF byte handling



```
0000 147 ;
0000 148 ; Definitions that follow the standard UCB fields
0000 149 ;
0000 150 $DEFINI UCB ; Start of UCB definitions
0000 151
000000A0 0000 152 .=UCB$L_DPC+4 ; Position at end of error log
00A0 153 ; extension to UCB
00A0 154 $DEF UCB$L_SAV_R4 .BLKL 1 ; copy of R4 for CANCEL routine
00A4 155
00A4 156 $DEF UCB$W_PT_PPS .BLKW 1 ; copy of status reg. (err. log)
00A6 157
00A6 158 $DEF UCB$B_PT_OFLCNT .BLKB 1 ; Off-line counter
00A7 159
00A7 160 $DEF UCB$B_PT_PPB .BLKB 1 ; Copy of data reg. for err. log
00A8 161
00A8 162 $DEF UCB$W_MY_REFCNT .BLKW 1 ; Counter for channels assigned
00AA 163
00AA 164 $DEF UCB$K_PT_UCBLEN ; Length of extended UCB
00AA 165
00AA 166 $DEFEND UCB ; End of UCB definitions
0000 167 ;
0000 168 ; Device register offsets from CSR address
0000 169 ;
0000 170 $DEFINI PT ; Start of status definitions
0000 171
00000002 0000 172 $DEF PT_PRS ; reader Control/status register
0000 173 .BLKW 1
0002 174 ;
0002 175 ; Bit positions for device control/status registers
0002 176 ;
0002 177 _VIELD PT_CSR,0,<- ; Control/status register
0002 178 <RDENA,,M>,- ; Reader enable bit
0002 179 <,5>,- ; Skip five bits
0002 180 <IE,,M>,- ; Reader interrupt enable bit
0002 181 <DONE,,M>,- ; Reader done bit
0002 182 <,3>,- ; Skip three bits
0002 183 <BUSY,,M>,- ; Reader busy bit
0002 184 <,3>,- ; Skip three bits
0002 185 <ERROR,,M>,- ; Reader error bit
0002 186 >
00000004 0002 187 $DEF PT_PRB ; reader data register
0004 188 .BLKW 1
00000006 0004 189 $DEF PT_PPS ; Punch Control/status register
0006 190 .BLKW 1
0006 191 _VIELD PT_STS,0,<- ; Control/status register
0006 192 <,6>,- ; disregard bits
0006 193 <IE,,M>,- ; interrupt enable
0006 194 <READY,,M>,- ; device ready bit
0006 195 <,7>,- ; disregard bits
0006 196 <ERROR,,M>,- ; device error bit
0006 197 >
00000008 0006 198 $DEF PT_PPB ; Punch data register
0006 199 .BLKW 1
0008 200 $DEFEND PT ; End of device register
0000 201 ; definitions.
0000 202 .SBTTL Standard tables
```

```
0000 204
0000 205 ;
0000 206 ; Driver prologue table
0000 207 ;
0000 208
0000 209          DPTAB      -                ; DPT-creation macro
0000 210          END=PT_END,-            ; End of driver label
0000 211          ADAPTER=UBA,-          ; Adapter type
0000 212          UCBSIZE=<UCB$K_PT_UCBLEN>,- ; Length of UCB
0000 213          NAME=PCDRIVER          ; Driver name
0038 214          DPT_STORE INIT        ; Start of load
0038 215          ; initialization table
0038 216          DPT_STORE UCB,UCB$B_FIPL,B,8 ; Device fork IPL
003C 217          DPT_STORE UCB,UCB$B_DIPL,B,20 ; Device int. IPL (BR4)
0040 218          DPT_STORE UCB,UCB$L_DEVCHAR,L,<- ; Device characteristics
0040 219          DEV$M_AVL!-           ; e.g., dev. available
0040 220          DEV$M_ELG!-           ; error logging enabled
0040 221          DEV$M_REC!-           ; record oriented
0040 222          DEV$M_IDV!-           ; input device-reader
0040 223          DEV$M_ODV>            ; output device-punch
0047 224          DPT_STORE UCB,UCB$B_DEVCLASS,B,MY_CLASS ; store my device class
004B 225          DPT_STORE UCB,UCB$B_DEVTYPE,B,MY_TYPE ; Store my device type
004F 226          DPT_STORE UCB,UCB$W_DEVSTS,W,0 ; Clear all flags
0054 227          DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,NULL_BYTES ; init. # null bytes
0059 228
0059 229          DPT_STORE REINIT        ; Start of reload
0059 230          ; initialization table
0059 231          DPT_STORE DDB,DDB$L_DDT,D,PC$DDT ; Address of DDT
005E 232          DPT_STORE CRB,CRB$L_INTD+4,D,- ; Address of reader int.
005E 233          PR_INTERRUPT           ; service routine
0063 234          DPT_STORE CRB,CRB$L_INTD2+4,D,- ; Address of punch int.
0063 235          PT_INTERRUPT           ; service routine
0068 236          DPT_STORE CRB,-         ; Address of controller
0068 237          CRB$L_INTD+VEC$L_INITIAL,- ; initialization routine
0068 238          D,PT_CONTROL_INIT
006D 239
006D 240          DPT_STORE END            ; End of initialization
0000 241          ; tables
0000 242
0000 243 ;
0000 244 ; Driver dispatch table
0000 245 ;
0000 246
0000 247          DDTAB      -                ; DDT-creation macro
0000 248          DEVNAM=PC,-             ; Name of device
0000 249          START=PT_START,-         ; Start I/O routine
0000 250          FUNCTB=PT_FUNCNTABLE,-   ; FDT address
0000 251          CANCEL=PT_CANCEL,-       ; Cancel I/O routine
0000 252          REGDMP=PT_REG_DUMP,-     ; Register dump routine
0000 253          UNITINIT=PT_UNIT_INIT,- ; Unit init. routine
0000 254          ERLGBF=ERBF$SIZE+EMB$L_DV_REGSAV ; Size of err. log. buf.
0038 255          ; (plus system overhead)
0038 256 ;
0038 257 ; Function dispatch table
0038 258 ;
0038 259          PT_FUNCNTABLE:            ; FDT for driver
0038 260          FUNCTAB , -                ; Valid I/O functions
```

```
0038 261 <READVBLK,- ; READ virtual
0038 262 READLBLK,- ; READ logical
0038 263 READPBLK,- ; READ physical
0038 264 MOUNT,- ; MOUNT ACP function
0038 265 CREATE,- ; CREATE function
0038 266 READPROMPT,- ; READPROMPT function
0038 267 ACPCONTROL,- ; ACPCONTROL function
0038 268 MODIFY,- ; set # null bytes
0038 269 DELETE,- ; reset # null bytes
0038 270 DEACCESS,- ; read # null bytes
0038 271 WRITEOF,- ; write EOF function
0038 272 WRITEVBLK,- ; Write virtual
0038 273 WRITELBLK,- ; Write logical
0038 274 WRITEPBLK> ; Write physical
0040 275 FUNCTAB ,- ; Buffered functions
0040 276 <READVBLK,- ; READ virtual
0040 277 READLBLK,- ; READ logical
0040 278 READPBLK,- ; READ physical
0040 279 MOUNT,- ; MOUNT ACP function
0040 280 CREATE,- ; CREATE function
0040 281 READPROMPT,- ; READPROMPT function
0040 282 ACPCONTROL,- ; ACPCONTROL function
0040 283 MODIFY,- ; set # null bytes
0040 284 DELETE,- ; reset # null bytes
0040 285 DEACCESS,- ; read # null bytes
0040 286 WRITEOF,- ; write EOF function
0040 287 WRITEVBLK,- ; Write virtual
0040 288 WRITELBLK,- ; Write logical
0040 289 WRITEPBLK> ; Write physical
0048 290 FUNCTAB PT_FDT_ROUTINE,- ; FDT write routine for
0048 291 <WRITEVBLK,- ; write virtual,
0048 292 WRITELBLK,- ; write logical,
0048 293 WRITEPBLK> ; and write physical.
0054 294 FUNCTAB PR_FDT_ROUTINE,- ; FDT READ routine for
0054 295 <READVBLK,- ; READ virtual,
0054 296 READLBLK,- ; READ logical,
0054 297 READPBLK> ; and READ physical.
0060 298 FUNCTAB ACP_FDT,<MOUNT> ; FDT ACP routines
006C 299 FUNCTAB Q_TO_ACP,- ; Check for ACP FDTs
006C 300 <ACPCONTROL,- ; ACPCONTROL routine
006C 301 WRITEOF> ; WRITE EOF routine
0078 302 FUNCTAB FOR_PNCH,- ; Make sure for punch
0078 303 <MODIFY,- ; set # null bytes
0078 304 DELETE,- ; reset # null bytes
0078 305 DEACCESS> ; read # null bytes
0084 306 FUNCTAB SET_FDT,<MODIFY> ; set # null bytes
0090 307 FUNCTAB +EXE$QIODRVPKT,- ; no more checking
0090 308 <DELETE,- ; for null byte reset or
0090 309 CREATE,- ; CREATE function or
0090 310 READPROMPT,- ; READPROMPT function or
0090 311 DEACCESS> ; reading # null bytes
009C 312 .SBTTL PT_CONTROL_INIT, Controller initialization routine
```

```
009C 314
009C 315 ;++
009C 316 ; PT_CONTROL_INIT, Readies controller for I/O operations
009C 317 ;
009C 318 ; Functional description:
009C 319 ;
009C 320 ; The operating system calls this routine in 3 places:
009C 321 ;
009C 322 ; at system startup
009C 323 ; during driver loading and reloading
009C 324 ; during recovery from a power failure
009C 325 ;
009C 326 ; Inputs:
009C 327 ;
009C 328 ; R4 - address of the CSR (controller status register)
009C 329 ; R5 - address of the IDB (interrupt data block)
009C 330 ; R6 - address of the DDB (device data block)
009C 331 ; R8 - address of the CRB (channel request block)
009C 332 ;
009C 333 ; Outputs:
009C 334 ;
009C 335 ; The routine must preserve all registers except R0-R3.
009C 336 ;
009C 337 ;--
009C 338
009C 339 PT_CONTROL_INIT: ; Initialize controller
05 009C 340 RSB ; Return
009D 341 .SBTTL PT_UNIT_INIT, Unit initialization routine
```

```
009D 343
009D 344 ;++
009D 345 ; PT_UNIT_INIT, Readies unit for I/O operations
009D 346 ;
009D 347 ; Functional description:
009D 348 ;
009D 349 ; The operating system calls this routine after calling the
009D 350 ; controller initialization routine:
009D 351 ;
009D 352 ; at system startup
009D 353 ; during driver loading
009D 354 ; during recovery from a power failure
009D 355 ;
009D 356 ; When using the SYSGEN CONNECT command, the reader is
009D 357 ; connected first (as PC0:), and the punch is connected
009D 358 ; second (as PC1:). The initialization routine sets both
009D 359 ; units on-line, and clears the appropriate device
009D 360 ; characteristics bit (i.e., punch does only output oper-
009D 361 ; ations, reader does only input operations). Two UCBs
009D 362 ; are used so that operations can be overlapped on the
009D 363 ; reader and punch. No controller channel is ever requested
009D 364 ; (with REQCHAN) since the PC11 reader/punch is essentially
009D 365 ; two independent devices in one box, and no controller is
009D 366 ; really present. (Therefore operations may occur on both
009D 367 ; the reader and the punch at the same time.)
009D 368 ;
009D 369 ; Inputs:
009D 370 ;
009D 371 ; R4 - address of the CSR (controller status register)
009D 372 ; R5 - address of the UCB (unit control block)
009D 373 ;
009D 374 ; Outputs:
009D 375 ;
009D 376 ; The routine must preserve all registers except R0-R3.
009D 377 ;
009D 378 ;--
009D 379
009D 380 PT_UNIT_INIT: ; Initialize unit
64 A5 10 A8 009D 381 BISW #UCB$M_ONLINE,UCB$W_STS(R5) ; Set unit online
50 54 A5 B0 00A1 382 MOVW UCB$W_UNIT(R5),R0 ; Get unit #
50 B5 00A5 383 TSTW R0 ; Is unit # 0?
0A 13 00A7 384 BEQL 10$ ; If so, reader
38 A5 04000000 8F CA 00A9 385 BICL #DEV$M_IDV,UCB$L_DEVCHAR(R5) ; No input if punch
08 11 00B1 386 BRB 20$
38 A5 08000000 8F CA 00B3 387 10$: BICL #DEV$M_ODV,UCB$L_DEVCHAR(R5) ; No output if reader
05 00BB 388 20$: RSB ; Return
00BC 389 .SBTTL PT_FDT_ROUTINE, Punch FDT routine
```

```

00BC 391
00BC 392 ;++
00BC 393 ; PT_FDT_ROUTINE, Punch FDT routine
00BC 394 ;
00BC 395 ; Functional description:
00BC 396 ;
00BC 397 ; This FDT routine makes all the standard accessibility checks
00BC 398 ; for buffered I/O operations, allocates a buffer from system
00BC 399 ; pool, and copies the user data to be punched to the system
00BC 400 ; buffer. The standard IRP and PCB fields are updated. In the
00BC 401 ; case of a WRITEOF function code, the IRP is queued to the ACP.
00BC 402 ; The routine makes sure that the punch UCB is being
00BC 403 ; referenced by the user.
00BC 404 ;
00BC 405 ; Inputs:
00BC 406 ;
00BC 407 ; R0-R2 - scratch registers
00BC 408 ; R3 - address of the IRP (I/O request packet)
00BC 409 ; R4 - address of the PCB (process control block)
00BC 410 ; R5 - address of the UCB (unit control block)
00BC 411 ; R6 - address of the CCB (channel control block)
00BC 412 ; R7 - bit number of the I/O function code
00BC 413 ; R8 - address of the FDT table entry for this routine
00BC 414 ; R9-R11 - scratch registers
00BC 415 ; AP - address of the 1st function dependent $Q10 parameter
00BC 416 ;
00BC 417 ; Outputs:
00BC 418 ;
00BC 419 ; The routine must preserve all registers except R0-R2, and
00BC 420 ; R9-R11.
00BC 421 ;
00BC 422 ;--
00BC 423
00BC 424 PT_FDT_ROUTINE: ; Punch FDT routine ala CRDRIVER
50 000000F4 8F D0 00BC 425 MOVL #SS$ ILLIOFUNC,R0 ; Assume wrong UCB
06 38 A5 1B E0 00C3 426 BBS #DEV$V_ODV,UCB$_DEVCHAR(R5),5$ ; Check for punch UCB
00000000'GF 17 00C8 427 JMP GAEXE$ABORTIO ; Error if not
50 50 6C D0 00CE 428 5$: MOVL P1(AP),R0 ; Get add. of buffer
51 04 AC 3C 00D1 429 MOVZWL P2(AP),R1 ; Get length of buffer
5F 13 00D5 430 BEQL 10$ ; Zero length transfer?
00000000'GF 16 00D7 431 JSB GAEXE$WRITECHK ; Check access of user
00DD 432 ; buffer -- no return if
00DD 433 ; no accessibility
32 A3 51 B0 00DD 434 MOVW R1,IRP$_BCNT(R3) ; Save user buffer len.
50 09 BB 00E1 435 PUSHR #AM<R0,R3> ; Save registers
51 0C C0 00E3 436 ADDL2 #BUF_OVR HD,R1 ; Account for overhead
00000000'GF 16 00E6 437 JSB GAEXE$BUFFRQUOTA ; Check buffer quota
50 50 E9 00EC 438 BLBC R0,20$ ; Abort if insufficient
00000000'GF 16 00EF 439 JSB GAEXE$ALLOCBUF ; Allocate system buffer
47 50 E9 00F5 440 BLBC R0,20$ ; Abort on failure
50 09 BA 00F8 441 POPR #AM<R0,R3> ; Restore registers
2C A3 52 D0 00FA 442 MOVL R2,IRP$_SVAPTE(R3) ; Store returned buf add
30 A3 51 B0 00FE 443 MOVW R1,IRP$_BOFF(R3) ; And byte quota charged
50 50 DD 0102 444 PUSHL R0 ; Save reg.
50 0080 C4 D0 0104 445 MOVL PCB$_JIB(R4),R0 ; Get JIB address
20 A0 51 C2 0109 446 SUBL R1,JIB$_BYTCNT(R0) ; Charge process for buf
50 8E00 010D 447 POPL R0 ; Restore reg.

```

```

53      52  38  BB  0110  448      PUSHR   #^M<R3,R4,R5>          ; Save reg.'s for MOV C3
      82  53  D0  0112  449      ADDL3   #BUF_OVR_HD,R2,R3      ; Find system data area
      62  50  D0  0116  450      MOVL    R3,(R2)+             ; Save its address
      51  50  D0  0119  451      MOVL    R0,(R2)             ; Save user buffer add.
63      60  51  28  011C  452      SUBL2   #BUF_OVR_HD,R1      ; Set transfer length
      011F  453      MOV C3   R1,(R0),(R3)      ; Copy user buffer to
      0123  454      ; system data buffer
      38  BA  0123  455      POPR    #^M<R3,R4,R5>      ; Restore reg.'s
      59  20  A3  B0  0125  456      MOVW   IRP$W_FUNC(R3),R9   ; Get function code
28      59  06  00  ED  0129  457      CMPZV  S^#IO$V_FCODE,S^#IO$S_FCODE,R9,- ; See if WRITEOF $QIO
      012E  458      ; If so,
      43  13  012E  459      BEQL   TO_ACP              ; Queue IRP to ACP
00000000'GF 17  0130  460      JMP    G^EXE$QIODRVPKT     ; Queue driver packet
      0136  461      ; FDT routine done
      0136  462
      0136  463 ;
      0136  464 ;
      0136  465 ;
      50  01  3C  0136  466 10$: MOVZWL  #SS$ NORMAL,R0      ; Normal completion
00000000'GF 17  0139  467      JMP    G^EXE$FINISHIOC     ; Goto finish I/O
      0C  BA  013F  468 20$: POPR    #^M<R2,R3>      ; Clean up stack
00000000'GF 17  0141  469      JMP    G^EXE$ABORTIO       ; Quota or buffer
      0147  470      ; allocation failure
      0147  471      .SBTTL ACP_FDT ACP $QIO FDT Routine
```

```
0147 473 ;++
0147 474 ; ACP_FDT, ACP $QIO FDT routine
0147 475 ;
0147 476 ; Functional description:
0147 477 ;
0147 478 ; This FDT routine increments the transaction count in the VCB,
0147 479 ; and queues the IRP to the paper tape ACP (PCACP). It is
0147 480 ; used for the MOUNT $QIO. MOUNT privilege is required to
0147 481 ; successfully issue the MOUNT $QIO request.
0147 482 ;
0147 483 ; Entry point Q_TO_ACP is used to service both the WRITEOF and
0147 484 ; ACPCONTROL $QIOs. Control is re-transferred to TO_ACP from
0147 485 ; routine PT_FDT_ROUTINE in the case of a WRITEOF $QIO.
0147 486 ;
0147 487 ; No check is made on the punch or reader UCB, since
0147 488 ; requests for ACP operation will find the errors.
0147 489 ;
0147 490 ; Inputs:
0147 491 ;
0147 492 ; R0-R2 - scratch registers
0147 493 ; R3 - address of the IRP (I/O request packet)
0147 494 ; R4 - address of the PCB (process control block)
0147 495 ; R5 - address of the UCB (unit control block)
0147 496 ; R6 - address of the CCB (channel control block)
0147 497 ; R7 - bit number of the I/O function code
0147 498 ; R8 - address of the FDT table entry for this routine
0147 499 ; R9-R11 - scratch registers
0147 500 ; AP - address of the 1st function dependent $QIO parameter
0147 501 ;
0147 502 ; Outputs:
0147 503 ;
0147 504 ; The routine must preserve all registers except R0-R2, and
0147 505 ; R9-R11.
0147 506 ;
0147 507 ;--
0147 508
0147 509 ACP_FDT:
0147 510 MOVZWL #SS$ NOPRIV,R0 ; MOUNT $QIO issued
0147 511 IFNPRIV MOUNT,NO_PRIV ; Assume insuf. priv.
0147 512 BBCC #UCB$V MOUNTING,UCB$W_STS(R5),NT_MNT ; Have MOUNT privilege?
0147 513 BRB Q_TO_ACP ; If clr, not sync.
0147 514 NT_MNT: MOVZWL #SS$ DEVNOTMOUNT,R0 ; Yes, continue
0147 515 NO_PRIV: JMP G^EXE$ABORTIO ; Not mounting status
0147 516 Q_TO_ACP: ; Signal error
0147 517 MOVL UCB$L_VCB(R5),R0 ; IRP request okay
0147 518 BEQL NT_MNT ; Get VCB address
0147 519 CMPZV SA#IO$V_FCODE,SA#IO$S_FCODE,- ; Is ACP mounted?
0147 520 IRP$W_FUNC(R3),SA#IO$_WRITEOF ; Check for WRITEOF
0147 521 BNEQ NT_WRT ; Function code
0147 522 BRW PT_FDT_ROUTINE ; If neq, Q to ACP
0147 523 TO_ACP: MOVL UCB$L_VCB(R5),R0 ; If eql, WRITEOF
0147 524 NT_WRT: SETIPL #IPL$_SYNCH ; Restore VCB address
0147 525 INCW VCB$W_TRANS(R0) ; Synch. access to VCB
0147 526 JMP G^EXE$QIOACPPKT ; Increment trans. count
0147 527 .SBTTL NULL_BYTES FDT Routines for header/trailer ; Queue IRP to ACP
```



```

0183 529 ;
0183 530 ; FDT routines for processing MODIFY, DELETE, and DEACCESS
0183 531 ; $QIOS. Very similar, and very simple.
0183 532 ;
0183 533 FOR_PNCH: ; Have punch UCB?
50 000000F4 8F D0 0183 534 MOVL #SS$_ILLIOFUNC,R0 ; Assume not
06 38 A5 1B E0 018A 535 BBS #DEV$_ODV,UCB$_DEVCHAR(R5),5$ ; Check for punch
00000000'GF 17 018F 536 JMP G^EXE$ABORTIO ; Not punch UCB
50 01 D0 0195 537 5$: MOVL #SS$_NORMAL,R0 ; Have punch
05 0198 538 RSB ; Return to FDT table
0199 539 SET_FDT: ; Get #, null bytes
38 A3 04 AC 3C 0199 540 MOVZWL P2(AP),IRP$_MEDIA(R3) ; Store # of null bytes
00000000'GF 17 019E 541 JMP G^EXE$QIODRVPKT ; Synchronize with other
01A4 542 ; I/O requests for unit
01A4 543 .SBTTL PR_FDT_ROUTINE Reader FDT Routine

```

```

01A4 545 ;++
01A4 546 ; PR_FDT_ROUTINE, READER FDT routine
01A4 547 ;
01A4 548 ; Functional description:
01A4 549 ;
01A4 550 ; This FDT routine makes all the standard accessibility checks
01A4 551 ; for buffered I/O operations, and allocates a buffer from system
01A4 552 ; pool (into which the data will be read).
01A4 553 ; The standard IRP and PCB fields are updated.
01A4 554 ; A check is made to determine if the reader UCB is being
01A4 555 ; referenced.
01A4 556 ; Inputs:
01A4 557 ; R0-R2 - scratch registers
01A4 558 ; R3 - address of the IRP (I/O request packet)
01A4 559 ; R4 - address of the PCB (process control block)
01A4 560 ; R5 - address of the UCB (unit control block)
01A4 561 ; R6 - address of the CCB (channel control block)
01A4 562 ; R7 - bit number of the I/O function code
01A4 563 ; R8 - address of the FDT table entry for this routine
01A4 564 ; R9-R11 - scratch registers
01A4 565 ; AP - address of the 1st function dependent $QIO parameter
01A4 566 ; Outputs:
01A4 567 ; The routine must preserve all registers except R0-R2, and
01A4 568 ; R9-R11.
01A4 569 ;--
01A4 570 PR_FDT_ROUTINE: ; reader FDT routine ala CRDRIVER
50 000000F4 8F D0 01A4 571 MOVL #SS$ ILLIOFUNC,R0 ; Assume wrong UCB
06 38 A5 1A E0 01AB 572 BBS #DEV$V_IDV,UCB$L_DEVCHAR(R5),5$ ; Test for reader UCB
00000000'GF 17 01B0 573 JMP G^EXE$ABORTIO ; Error if not
50 50 6C D0 01B6 574 5$: MOVL P1(AP),R0 ; Get add. of buffer
51 04 AC 3C 01B9 575 MOVZWL P2(AP),R1 ; Get length of buffer
46 13 01BD 576 BEQL 10$ ; Zero length transfer?
00000000'GF 16 01BF 577 JSB G^EXE$READCHK ; Check access of user
01C5 578 ; buffer -- no return if
01C5 579 ; no accessibility
32 A3 51 B0 01C5 580 MOVW R1,IRP$W_BCNT(R3) ; Save user buffer len.
09 BB 01C9 581 PUSHR #^M<R0,R3> ; Save registers
51 0C C0 01CB 582 ADDL2 #BUF_OVR_HD,R1 ; Account for overhead
00000000'GF 16 01CE 583 JSB G^EXE$BUFFRQUOTA ; Check buffer quota
37 50 E9 01D4 584 BLBC R0,20$ ; Abort if insufficient
00000000'GF 16 01D7 585 JSB G^EXE$ALLOCBUF ; Allocate system buffer
2E 50 E9 01DD 586 BLBC R0,20$ ; Abort on failure
09 BA 01E0 587 POPR #^M<R0,R3> ; Restore registers
2C A3 52 D0 01E2 588 MOVL R2,IRP$L_SVAPTE(R3) ; Store returned buf add
30 A3 51 B0 01E6 589 MOVW R1,IRP$W_BOFF(R3) ; And byte quota charged
50 DD 01EA 590 PUSHL R0 ; Save reg.
50 0080 C4 D0 01EC 591 MOVL PCB$L_JIB(R4),R0 ; Get JIB address
20 A0 51 C2 01F1 592 SUBL R1,JIB$L_BYTCNT(R0) ; Charge process for buf
50 8ED0 01F5 593 POPL R0 ; Restore reg.
82 0C A2 9E 01F8 594 MOVAB BUF_OVR_HD(R2),(R2)+ ; Save data area add.
62 50 D0 01FC 595 MOVL R0,(R2) ; Save user buffer add.
00000000'GF 17 01FF 596 JMP G^EXE$QIODRVPKT ; Queue driver packet
0205 597 ; Entered on zero-length transfer or allocation failure
50 01 3C 0205 598 10$: MOVZWL #SS$ NORMAL,R0 ; Normal completion
00000000'GF 17 0208 599 JMP G^EXE$FINISHIOC ; Goto finish I/O
0C BA 020E 600 20$: POPR #^M<R2,R3> ; Clean up stack
00000000'GF 17 0210 601 JMP G^EXE$ABORTIO ; Quota or buffer

```

PCDRIVER  
V06

- VAX/VMS PAPER TAPE READER/PUNCH DRIVER 12-SEP-1984 17:59:27 VAX-11 Macro V03-01 Page 15  
PR\_FDT\_ROUTINE Reader FDT Routine 22-AUG-1984 20:06:00 WORK2:[COURSE.DRIVER]PCDRIVER.MAR;(1)

0216 602  
0216 603

.SBTTL PT\_START, Start I/O routine

; allocation failure

```

0216 605
0216 606 ;
0216 607 ; Enter here at the start of an I/O operation. The first step
0216 608 ; is to determine what I/O function code was specified in the
0216 609 ; $QIO being serviced.
0216 610 ;
0216 611 PT_START:
00 64 A5 05 E4 0216 612 BBSC #UCB$V_POWER,UCB$W_STS(R5),1$ ; Det. func. req.
009A C5 20 A3 B0 021B 613 1$: MOVW IRP$W_FUNC(R3),UCB$W_FUNC(R5) ; Make sure bit clear
; Allow inhibit of error
0221 614 ; logging via modifiers
06 00 ED 0221 615 CMPZV #IRP$V_FCODE,#IRP$$_FCODE,- ; Look for MODIFY $QIO
36 20 A3 0224 616 IRP$W_FUNC(R3),#IO$_MODIFY
06 00 ED 0227 617 BEQL 10$ ; Found if eq1
34 20 A3 0229 618 CMPZV #IRP$V_FCODE,#IRP$$_FCODE,- ; Look for DEACCESS $QIO
06 00 ED 022C 619 IRP$W_FUNC(R3),#IO$_DEACCESS
4C 13 022F 620 BEQL 20$ ; Found if eq1
06 00 ED 0231 621 CMPZV #IRP$V_FCODE,#IRP$$_FCODE,- ; Look for DELETE $QIO
35 20 A3 0234 622 IRP$W_FUNC(R3),#IO$_DELETE
2E 13 0237 623 BEQL 3$ ; Found if eq1
06 00 ED 0239 624 CMPZV #IRP$V_FCODE,#IRP$$_FCODE,- ; Look for CREATE $QIO
33 20 A3 023C 625 IRP$W_FUNC(R3),#IO$_CREATE
19 13 023F 626 BEQL 30$ ; Found if eq1
06 00 ED 0241 627 CMPZV #IRP$V_FCODE,#IRP$$_FCODE,- ; Look for READPROMPT
37 20 A3 0244 628 IRP$W_FUNC(R3),#IO$_READPROMPT ; $QIO
3A 12 0247 629 BNEQ 100$ ; Read/write if neq
11 38 A5 1A E0 0249 630 BBS #DEV$V_IDV,UCB$L_DEVCHAR(R5),2$ ; Reader? dsable--> set
68 A5 01 AA 024E 631 25$: BICW #BIT0,UCB$W_DEVSTS(R5) ; Enable function
68 A5 0100 8F AB 0252 632 BISW #BIT8,UCB$W_DEVSTS(R5) ; Mark enable for read phys.
11 11 0258 633 BRB 5$ ; All done
EF 38 A5 1A E0 025A 634 30$: BBS #DEV$V_IDV,UCB$L_DEVCHAR(R5),25$ ; Reader? enable-->clr
68 A5 0201 8F AB 025F 635 2$: BISW #BIT0!BIT9,UCB$W_DEVSTS(R5) ; Disable function
04 11 0265 636 BRB 5$ ; All done
42 A5 32 B0 0267 637 3$: MOVW #NULL_BYTES,UCB$W_DEVBUFSIZ(R5) ; Reset # of null bytes
51 D4 026B 638 5$: CLRL R1 ; No dev. dep. info.
50 01 D0 026D 639 7$: MOVL #SS$_NORMAL,R0 ; Successful completion
0270 640 REQCOM ; All done
42 A5 38 A3 B0 0276 641 10$: MOVW IRP$L_MEDIA(R3),UCB$W_DEVBUFSIZ(R5) ; Set # null bytes
EE 11 027B 642 BRB 5$ ; And finish up
51 42 A5 3C 027D 643 20$: MOVZWL UCB$W_DEVBUFSIZ(R5),R1 ; Read # null bytes
EA 11 0281 644 BRB 7$ ; And finish up
03 2A A3 01 E1 0283 645 100$: BBC #IRP$V_FUNC,IRP$W_STS(R3),PT_PUNCH ; Write if clear
024C 31 0288 646 BRW PR_START ; Otherwise, read func
028B 647 .SBTTL PT_PUNCH, Start a punch operation
```

```

028B 649 ;++
028B 650 ; PT_PUNCH - Start a punch operation
028B 651 ;
028B 652 ; Functional description:
028B 653 ;
028B 654 ; The punch interrupt enable bit is set. One character at a
028B 655 ; time (from the system buffer pointed to by UCB$L_SVAPTE) is
028B 656 ; moved to the punch's data buffer, and an interrupt is awaited.
028B 657 ; If powerfail occurs, the entire operation is retried (if it is
028B 658 ; the first $QIO for the user); otherwise, the operation is aborted
028B 659 ; (unless the powerfail occurs while punching the trailer, in
028B 660 ; which case the entire trailer may not be punched). In the case
028B 661 ; of an aborted punch, the high-order word of the first longword
028B 662 ; in the IOSB will contain the number of bytes punched before the
028B 663 ; powerfail occurred (error code = SS$_TAPEPOSLOST).
028B 664 ;
028B 665 ; A counter is kept on the number of bytes left to be punched
028B 666 ; (UCB$W_BCNT). When this counter goes to zero, the request
028B 667 ; is complete.
028B 668 ;
028B 669 ; In the case of a virtual or logical I/O operation
028B 670 ; a header of UCB$W_DEVBUSIZ null bytes is punched
028B 671 ; before the actual data is punched; similarly a trailer
028B 672 ; tape (of UCB$W_DEVBUSIZ null bytes) is punched when the
028B 673 ; channel to the punch is deassigned/deallocated by the
028B 674 ; user (in the cancel I/O routine (PT_CANCEL)). The user can
028B 675 ; suppress the defaults and/or punching an EOF byte (cntrl-Z).
028B 676 ;
028B 677 ; Inputs:
028B 678 ; R3 - address of the IRP (I/O request packet)
028B 679 ; R5 - address of the UCB (unit control block)
028B 680 ; Outputs:
028B 681 ; R0 - 1st longword of I/O status: contains status code and
028B 682 ; number of bytes transferred
028B 683 ; R1 - 2nd longword of I/O status: device-dependent
028B 684 ;
028B 685 ; The routine must preserve all registers except R0-R2 and R4.
028B 686 ;--
028B 687 PT_PUNCH:
028B 688 BICW #END,UCB$W_DEVSTS(R5) ; Process an I/O packet
78 A5 10 AA 028B 689 MOVQ IRP$L_SVAPTE(R3),UCB$L_SVAPTE(R5) ; Not punching trailer
028B 690 ; Note quad transfer
0294 691 ADDL2 #BUF_OVR_HD,UCB$L_SVAPTE(R5) ; for powerfail recovery
78 A5 0C CO 0294 692 PNCH_HDR: ; Skip sys buf header
0298 693 .ENABL LSB ; Trailer punch entry pt
0298 694 MOVL UCB$L_CRB(R5),R0 ; Get CRB
50 24 A5 DO 0298 695 MOVL @CRB$L_INTD+VEC$L_IDB(R0),R4 ; Get CSR address
54 2C B0 DO 029C 696 BISW #PT_STS_M IE,PT_PPS(R4) ; Set int. enable bit
04 A4 0040 8F A8 02A0 697 BBSS #PUNCH_HDR,UCB$W_DEVSTS(R5),5$ ; Hdr already pnchd?
08 68 A5 01 E2 02A6 698 MOVW UCB$W_REFC(R5),UCB$W_MY_REFCNT(R5) ; Save reference count
00A8 C5 5C A5 B0 02AB 699 BRB 10$ ; No - punch header
03 11 02B1 699 BRW MAINLP ; Punch data
00B4 31 02B3 700 5$: ; Punching trailer?
68 A5 10 B3 02B6 701 10$: BITW #END,UCB$W_DEVSTS(R5) ; Yes, if neq
2A 12 02BA 702 BNEQ 12$ ; Get function code
50 20 A3 B0 02BC 703 MOVW IRP$W_FUNC(R3),R0 ; Get function code
0B 50 06 00 ED 02C0 704 CMPZV S^#IO$V_FCODE,S^#IO$S_FCODE,R0,- ; No header punched for
02C5 705 S^#IO$_WRITEPBLK ; Physical I/O func.

```

```
68 A5 0200 0F 12 02C5 706 BNEQ 11$ ; Not phys. if neq
      04 13 02C7 707 BITW #BIT9,UCB$_DEVSTS(R5) ; Want trailer byte?
      01 13 02CD 708 BEQL 8$ ; No, if eq
68 A5 0094 01 A8 02CF 709 BISW #BIT0,UCB$_DEVSTS(R5) ; Mark want trailer
      31 02D3 710 8$: BRW MAINLP ; Skip header punching
68 A5 0100 0D A8 02D6 711 11$: BISW #PNCH!BIT0!START,UCB$_DEVSTS(R5) ; Set flag bits
      04 13 02E0 712 BITW #BIT8,UCB$_DEVSTS(R5) ; No EOF byte?
68 A5 01 01 AA 02E2 714 BEQL 12$ ; No, if eq
      44 A5 42 A5 3C 02E6 715 12$: MOVZWL UCB$_DEVBUFSIZ(R5),UCB$_DEVDEPEND(R5); Set # nullbytes
68 A5 10 B3 02EB 716 BITW #END,UCB$_DEVSTS(R5) ; Punching trailer?
      1F 13 02EF 717 BEQL NUL ; No, if eq
68 A5 01 B3 02F1 718 BITW #BIT0,UCB$_DEVSTS(R5) ; Punch EOF byte?
      19 13 02F5 719 BEQL NUL ; No if eq
      44 A5 D6 02F7 720 INCL UCB$_DEVDEPEND(R5) ; One extra byte pnchd.
00A7 C5 1A 90 02FA 721 PUNEOF: MOVB #EOF,UCB$_PT_PPB(R5) ; Save for error log.
      02FF 722 DSBINT ; Raise to powerfail
39 64 A5 05 E4 0305 723 BBSC #UCB$_POWER,UCB$_STS(R5),20$ ; Check for powerfail
      06 A4 1A 90 030A 724 MOVB #EOF,PT_PPB(R4) ; Punch EOF byte
      14 11 030E 725 BRB WAIT ; Wait for interrupt
00A7 C5 00 90 0310 726 NUL: MOVB #NULL,UCB$_PT_PPB(R5) ; Save for err. log.
      0315 727 NULLP: DSBINT ; No interrupts
23 64 A5 05 E4 031B 728 BBSC #UCB$_POWER,UCB$_STS(R5),20$ ; Check for powerfail
      06 A4 00 90 0320 729 MOVB #NULL,PT_PPB(R4) ; Punch null byte
      0324 730 WAIT: WFIKPCH 25$,#PT_TIMEOUT_SEC ; Wait for I/O comp.
      032E 731 ;
      032E 732 ; Enter here after interrupt service routine
      032E 733 ;
      0065 31 032E 734 BRW AFTINT ; Check for errors
      44 A5 D7 0331 735 BACK: DECL UCB$_DEVDEPEND(R5) ; Dec null byte count
      DF 12 0334 736 BNEQ NULLP ; Loop if more to punch
68 A5 04 AA 0336 737 BICW #START,UCB$_DEVSTS(R5) ; Clear hdr. punch flag
68 A5 10 B3 033A 738 BITW #END,UCB$_DEVSTS(R5) ; Punching header?
      2A 13 033E 739 BEQL MAINLP ; Yes if eq
      031B 31 0340 740 BRW INCAN ; Return for trailer
      0343 741 20$: ; Powerfail recovery
      0343 742 ENBINT ; Allow interrupts
00 68 A5 01 E4 0346 743 BBSC #PUNCH_HDR,UCB$_DEVSTS(R5),22$ ; Clear hdr. pnchd flag
68 A5 10 B3 034B 744 22$: BITW #END,UCB$_DEVSTS(R5) ; Header or trailer?
      07 13 034F 745 BEQL 23$ ; Header if eq
68 58 A5 D0 0351 746 MOVL UCB$_IRP(R5),R3 ; Retrieve IRP address
      FF40 31 0355 747 BRW PNCH_HDR ; Retry punching trailer
      008D 31 0358 748 23$: BRW POW_FAIL ; Retry punching header
      035B 749 ; after powerfail recov.
68 A5 10 B3 035B 750 25$: BITW #END,UCB$_DEVSTS(R5) ; Header or trailer?
      01 13 035F 751 BEQL 30$ ; Header if equal
      05 0361 752 RSB ; Return if trailer
00 68 A5 01 E4 0362 753 30$: BBSC #PUNCH_HDR,UCB$_DEVSTS(R5),35$ ; Report timeout
      00F5 31 0367 754 35$: BRW PT_TIMEOUT ; Make sure flag clear
      036A 755 .DSABL LSB
03 64 A5 03 E1 036A 756 MAINLP: BBC #UCB$_CANCEL,UCB$_STS(R5),1$ ; Check for $CANCEL
      0152 31 036F 757 BRW CANCEL ; $CANCEL if set
      0372 758 1$: DSBINT UCB$_DIPL(R5) ; No device interrupts
00A7 C5 78 B5 90 0379 759 MOVB @UCB$_SVAPTE(R5),UCB$_PT_PPB(R5) ; Save data for err.
      037F 760 ; log buf. dump routine
      037F 761 SETIPL #IPL$_POWER ; Raise to powerfail
5E 64 A5 05 E4 0382 762 BBSC #UCB$_POWER,UCB$_STS(R5),PT_RESTART ; Powerfail?
```

```
06 A4 78 B5 90 0387 763      MOVB    @UCB$L_SVAPTE(R5),PT_PP(B4)      ; Load punch data buffer
038C 764      WFIKPCH PT_TIMEOUT,#PT_TIMEOUT_SEC      ; 10 sec. timeout
0396 765 ;
0396 766 ;      Enter here from interrupt service routine (PT_INTERRUPT)
0396 767 ;
00A4 C5 04 A4 B0 0396 768 AFTINT: MOVW    PT_PPS(R4),UCB$W_PT_PPS(R5) ; Read CSR (save for
039C 769 ;      ; error logging later)
039C 770      IOFORK ;      ; Return to fork level
00A4 C5 0080 8F B3 03A2 771      BITW    #PT_STS_M_READY,UCB$W_PT_PPS(R5); Punch ready again?
66 13 03A9 772      BEQL    PT_ERROR ;      ; If so, error
00A4 C5 8000 8F B3 03AB 773      BITW    #PT_STS_M_ERROR,UCB$W_PT_PPS(R5); Error during punch?
5D 12 03B2 774      BNEQ    PT_ERROR ;      ; Yes, report it
68 A5 04 B3 03B4 775      BITW    #START,UCB$W_DEVSTS(R5) ;      ; Punching user data?
03 13 03B8 776      BEQL    10$ ;      ; Yes, if eq1
FF74 31 03BA 777      BRW    BACK ;      ; Punching header/trailer
78 A5 D6 03BD 778 10$: INCL    UCB$L_SVAPTE(R5) ;      ; Point to next byte to
03C0 779 ;      ; be punched
7E A5 B7 03C0 780      DECW    UCB$W_BCNT(R5) ;      ; Decrement byte count
A5 12 03C3 781      BNEQ    MAINLP ;      ; All done?
04 A4 0040 8F AA 03C5 782      BICW    #PT_STS_M_IE,PT_PPS(R4) ;      ; Clear int. enable bit
00 68 A5 07 E3 03CB 783      BBCS    #STARTING,UCB$W_DEVSTS(R5),FINISH ; Punched >= 1 oper.
03D0 784 ; After a transfer completes successfully, return the number of bytes
03D0 785 ; transferred and a success status code.
10 32 A3 F0 03D0 786 FINISH: INSV    IRP$W_BCNT(R3),#16,- ; Load number of bytes trans-
50 10 03D4 787 ;      ; ferred into high word of R0.
50 01 B0 03D6 788      MOVW    #SS$_NORMAL,R0 ; Load a success code into R0.
03D9 789 ;
03D9 790 ; Call I/O postprocessing.
03D9 791 ;
03D9 792 COMPLETE_IO: ;      ; Driver processing is finished.
68 A5 20 AA 03D9 793      BICW    #RD_HDR,UCB$W_DEVSTS(R5); Make sure reader flag clear
51 D4 03DD 794      CLRL    R1 ;      ; No device dependent information
03DF 795      REQCOM ;      ; Complete I/O.
03E5 796 ;
03E5 797 ; Recover from powerfail by retrying entire operation (for punch).
03E5 798 ; If powerfail occurred on first $QIO request from user.
03E5 799 ;
03E5 800 PT_RESTART: ;      ; Powerfail entry point
03E5 801 ENBINT ;      ; Allow interrupts (POP IPL)
03E8 802 POW_FAIL: ;      ; Enter after timeout
OF 68 A5 07 E0 03E8 803 BBS    #STARTING,UCB$W_DEVSTS(R5),POWER_FAIL ; No recovery if
03ED 804 ;      ; not punching 1st rec.
53 58 A5 D0 03ED 805      MOVL    UCB$L_IRP(R5),R3 ; Retrieve IRP address
00 68 A5 01 E4 03F1 806      BBSC    #PUNCH_HDR,UCB$W_DEVSTS(R5),1$ ; Clear header punched bit
FE92 31 03F6 807 1$: BRW    PT_PUNCH ; And retry operation
03F9 808 ; No recovery possible from reader powerfail; tell user how many
03F9 809 ; bytes read before powerfail occurred. Also, no recovery from
03F9 810 ; punch if not punching 1st $QIO request for this user. Tell user
03F9 811 ; how many bytes were punched before powerfail occurred.
03F9 812 PR_POWFAIL: ;      ; Powerfail entry point
03F9 813 ENBINT ;      ; Allow interrupts (POP IPL)
03FC 814 POWER_FAIL: ;      ; Enter after timeout
0287 30 03FC 815      BSBW    SETOFF ;      ; Clear int. enable bit
03FF 816 ; Note that a BSB-RSB sequence can be used since
03FF 817 ; the subroutine will not call any system macros
03FF 818 ; or routines (which would cause unpredictable
03FF 819 ; results when control would be returned to the
```

```

; caller's caller)
50 10 10 32 A3 F0 0404 822 SUBW2 UCB$W_BCNT(R5),IRP$W_BCNT(R3) ; Record bytes read/pun.
      50 0224 8F B0 040A 823 INSV IRP$W_BCNT(R3),#16,#16,R0 ; For status return
      C8 11 040F 824 MOVW #SS$TAPEPOSTLOST,R0 ; Return status
      0411 825 ; BRB COMPLETE_IO ; Complete I/O
      0411 826 ; Device error has occurred -- signal via SS$DRVERR
      0411 827 ; error code
      0411 828 ;
      0411 829 PT_ERROR: ; Error handling
      0411 830 BSBW SETOFF ; Clear interrupt enable bit
      00000000'GF 16 0414 831 JSB G^ERL$DEVICERR ; Report device error
      50 008C 8F 3C 041A 832 MOVZWL #SS$DRVERR,R0 ; Return error status
      32 A3 7E A5 A2 041F 833 SUBW2 UCB$W_BCNT(R5),IRP$W_BCNT(R3) ; Record bytes read/pun.
50 10 10 32 A3 F0 0424 834 INSV IRP$W_BCNT(R3),#16,#16,R0 ; For status return
      AD 11 042A 835 BRB COMPLETE_IO ; Call I/O postprocessing
      042C 836 ;
      042C 837 ; Assume that "no more tape in reader" is error generated, and
      042C 838 ; tell user how many bytes were read before error was encountered
      042C 839 ; in case P2 parameter specified more bytes than were read.
      042C 840 ;
      042C 841 PR_SPERR: ; Entered if no tape in reader,
      042C 842 ; reader off-line after at least
      042C 843 ; one previous character read,
      042C 844 ; or no power to reader
      042C 845 BSBW SETOFF ; Clear int. enable bit
      50 10 10 32 A3 F0 0434 847 INSV IRP$W_BCNT(R3),#16,#16,R0 ; Store num bytes read
      50 0870 8F B0 043A 848 MOVW #SS$ENDOFFILE,R0 ; In return status
      00 68 A5 01 E3 043F 849 BBSC #PUNCH_HDR,UCB$W_DEVSTS(R5),DON ; Set return status code
      0444 850 ; Special case for $COPY
      0444 851 ; so that reading will end on end
      0444 852 DON: BRW COMPLETE_IO ; of file (COPY issues 2nd read)
      50 00000870 8F D0 0447 853 COPY: MOVL #SS$ENDOFFILE,R0 ; Complete I/O
      FF88 31 044E 854 BRW COMPLETE_IO ; Fudge return to COPY
      0451 855 ; ; Finish operation
      0451 856 ; Device timeout handling. Return an error status code for punch.
      0451 857 ; Send message to operator, and loop until reader is set online.
      0451 858 ;
      0451 859 PR_TIMEOUT: ; Timeout handling
      0451 860 SETIPL UCB$B_FIPL(R5) ; Lower to fork first
      A2 64 A5 05 E4 0455 861 BBSC #UCB$V_POWER,UCB$W_STS(R5),POWER_FAIL ; Powerfail?
      11 11 045A 862 BRB COMMON ; Enter common timeout code
      FF89 31 045C 863 SKIP2: BRW POW_FAIL ; Handle powerfail
      045F 864 PT_TIMEOUT: ; Timeout handling
      045F 865 SETIPL UCB$B_FIPL(R5) ; Lower to fork first
      38 38 A5 1A E0 0463 866 BBS #DEV$V_IDV,UCB$L_DEVCHAR(R5),PUN_OT ; Reader timeout?
      EF 64 A5 05 E4 0468 867 BBSC #UCB$V_POWER,UCB$W_STS(R5),SKIP2 ; Powerfail?
      52 64 A5 03 E0 046D 868 COMMON: BBS #UCB$V_CANCEL,UCB$W_STS(R5),CANCEL ; Cancel I/O?
      29 38 A5 1B E0 0472 869 BBS #DEV$V_ODV,UCB$L_DEVCHAR(R5),PUN_OT ; Punch timeout?
      B0 68 A5 07 E0 0477 870 BBS #STARTING,UCB$W_DEVSTS(R5),PR_SPERR ; Reader EOF err?
0099 00A6 C5 01 0F 9D 047C 871 ACBB #LOOP_CNT,#1,UCB$B_PT_OFCLCNT(R5),MAIN ; Rdr-wait 30 sec.
      00A6 C5 94 0484 872 CLRB UCB$B_PT_OFCLCNT(R5) ; Clear off-line counter
      18 BB 0488 873 PUSHR #AM<R3,R4> ; Save registers
      54 00'8F 9A 048A 874 MOVZBL #MSG$DEVOFFLIN,R4 ; Set message type
      53 00000000'GF 9E 048E 875 MOVAB G^SYS$GL_OPRMBX,R3 ; Get opr. mbx. address
      00000000'GF 16 0495 876 JSB G^EXE$SNDEVMSG ; Send msg. to opr.

```



```

18 BA 049B 877 POPR #^M<R3,R4> ; Restore registers
007D 31 049D 878 BRW MAIN ; Try again
04A0 879 ;
04A0 880 ; Note that the punch is always assumed to be on-line, and
04A0 881 ; an error is recorded if it is not. Also, the I/O request
04A0 882 ; is terminated with an error code. (There is no on/off switch
04A0 883 ; for the punch unit on the PC11). The reader does have an
04A0 884 ; on/off switch, and the I/O operation is retried until the
04A0 885 ; reader unit is set on-line, with a message being sent to
04A0 886 ; the operator's mailbox (console) every 30 seconds informing
04A0 887 ; the operator that the paper-tape reader (PC0:) is off-line.
04A0 888 ; If an error occurs after some data has been read, it is
04A0 889 ; assumed that the error is "no more tape in reader" rather
04A0 890 ; than "reader is off-line".
04A0 891 ;
03 64 A5 05 E5 04A0 892 PUN_OT: BBCC #UCB$V_POWER,UCB$W_STS(R5),1$ ; Reader powerfail?
FF54 31 04A5 893 BRW POWER_FAIL ; Yes, go report powerfail
01DB 30 04A8 894 1$: BSBW SETOFF ; Clear interrupt enable bit
00000000'GF 16 04AB 895 JSB G^ERL$DEVICTMO ; Report timeout error
50 022C 8F 3C 04B1 896 MOVZWL #SS$_TIMEOUT,R0 ; Return error status.
32 A3 7E A5 A2 04B6 897 SUBW2 UCB$W_BCNT(R5),IRP$W_BCNT(R3) ; Record bytes read/pun.
50 10 10 32 A3 F0 04BB 898 INSV IRP$W_BCNT(R3),#16,#16,R0 ; For status return
FF15 31 04C1 899 BRW COMPLETE_IO ; Call I/O postprocessing.
04C4 900 CANCEL: ; Handle cancel I/O
32 A3 7E A5 A2 04C4 901 SUBW2 UCB$W_BCNT(R5),IRP$W_BCNT(R3) ; Num bytes read/punch
50 10 10 32 A3 F0 04C9 902 INSV IRP$W_BCNT(R3),#16,#16,R0 ; In return status
50 0830 8F B0 04CF 903 MOVW #SS$_CANCEL,R0 ; Return cancel error status
FF02 31 04D4 904 BRW COMPLETE_IO ; Call I/O postprocessing
04D7 905 .SBTTL PR_START, Start I/O routine (Reader)

```

```

04D7 907 ;++
04D7 908 ; PR_START - Start a reader operation
04D7 909 ;
04D7 910 ; Functional description:
04D7 911 ;
04D7 912 ;     One character at a time (from the system buffer pointed to
04D7 913 ;     by UCB$_SVAPTE) is moved to the reader's data buffer. The
04D7 914 ;     interrupt enable bit is then set, after which powerfail is
04D7 915 ;     tested for, and an interrupt is awaited. (If powerfail
04D7 916 ;     occurs, the entire operation is aborted.)
04D7 917 ;
04D7 918 ;     A counter is kept on the number of bytes left to be read
04D7 919 ;     (UCB$_BCNT). When this counter goes to zero, the request
04D7 920 ;     is complete. The request is also considered complete when an
04D7 921 ;     EOF byte is read, if looking for EOF bytes is enabled.
04D7 922 ;
04D7 923 ;     Before the first character is read (in a virtual or
04D7 924 ;     logical I/O operation), the tape is first scanned for
04D7 925 ;     the first non-null character (i.e. the header part of
04D7 926 ;     the tape is skipped). In the case of a physical I/O
04D7 927 ;     function, the header null bytes are considered data.
04D7 928 ; Inputs:
04D7 929 ;     R3     - address of the IRP (I/O request packet)
04D7 930 ;     R5     - address of the UCB (unit control block)
04D7 931 ; Outputs:
04D7 932 ;     R0     - 1st longword of I/O status: contains status code and
04D7 933 ;             number of bytes transferred
04D7 934 ;     R1     - 2nd longword of I/O status: device-dependent
04D7 935 ;     The routine must preserve all registers except R0-R2 and R4.
04D7 936 ;--
04D7 937 PR_START:
03 68 A5 01 E5 04D7 938 BBCC #PUNCH_HDR,UCB$_DEVSTS(R5),1$ ; Process an I/O packet
                                FF68 31 04DC 939 BRW COPY ; Special case for $COPY?
78 A5 2C B3 D0 04DF 940 1$: MOVL @IRP$_SVAPTE(R3),UCB$_SVAPTE(R5) ; Yes, if bit set
7E A5 32 A3 D0 04E4 941 MOVL IRP$_BCNT(R3),UCB$_BCNT(R5) ; Get buffer address
                                00A6 C5 94 04E9 942 CLRB UCB$_PT_OFLCNT(R5) ; Byte count, and offset
                                50 20 A3 B0 04ED 943 MOVW IRP$_FUNC(R3),R0 ; Clear off-line counter
27 68 A5 06 E2 04F1 944 BBSS #FRST_QIO,UCB$_DEVSTS(R5),MAIN ; Get function code
00A8 C5 5C A5 B0 04F6 945 MOVW UCB$_REFC(R5),UCB$_MY_REFCNT(R5) ; First $QIO?
                                06 00 ED 04FC 946 CMPZV S^#IO$_FCODE,S^#IO$_FCODE,- ; Save reference count
                                0C 50 04FF 947 R0,S^#IO$_READPBLK ; No header read if
                                0E 13 0501 948 BEQL FIX ; Phys. I/O operation
                                68 A5 20 A8 0503 949 BISW #RD_HDR,UCB$_DEVSTS(R5) ; Continue processing
68 A5 0200 8F B3 0507 950 BITW #BIT9,UCB$_DEVSTS(R5) ; Update default flags
                                0E 13 050D 951 BEQL MAIN ; Disable EOF search?
                                08 11 050F 952 BRB DSABL ; No, if eq1
68 A5 0100 8F B3 0511 953 FIX: BITW #BIT8,UCB$_DEVSTS(R5) ; Yes, if neq
                                04 12 0517 954 BNEQ MAIN ; Look for EOF bytes?
                                68 A5 01 A8 0519 955 DSABL: BISW #BIT0,UCB$_DEVSTS(R5) ; Look for EOF bytes if eq1
                                051D 956 MAIN: DSBINT UCB$_DIPL(R5) ; Update default
                                50 24 A5 D0 0524 957 MOVL UCB$_CRB(R5),R0 ; No device interrupts
                                54 2C B0 D0 0528 958 MOVL @CRB$_INTD+VEC$_IDB(R0),R4 ; Get CRB
64 8000 8F B3 052C 959 BITW #PT_CSR_M_ERROR,PT_PRS(R4) ; Get CSR address
                                12 13 0531 960 BEQL READ ; Reader on-line?
                                0533 961 WFIKPC PR_TIMEOUT,#SHORT_WAIT ; If eq1, yes
                                053D 962 IOFORK ; No, see if now on
                                D8 11 0543 963 BRB MAIN ; Lower to fork IPL
                                ; Check if on-line

```

```

0545 964 READ:  SETIPL #IPL$_POWER ; Raise to powerfail
02 64 A5 05 E4 0548 965 BBSC #UCB$_POWER,UCB$_STS(R5),30$ ; Powerfail?
03 11 054D 966 BRB GO ; No, try operation
FEA7 31 054F 967 30$: BRW PR_POWFAIL ; Yes--abort operation
41 8F 9B 0552 968 GO: MOVZBW #PT_CSR_M_IE!PT_CSR_M_RDENA,- ; Start reader and
64 0555 969 PT_PRS(R4) ; Enable reader int.
0556 970 WFIKPCH PT_TIMEOUT,#PT_TIMEOUT_SEC ; 10 sec. timeout
0560 971 ; Enter here from interrupt service routine (PR_INTERRUPT)
00A4 C5 64 80 0560 972 MOVW PT_PRS(R4),UCB$_PT_PPS(R5) ; Read CSR (and store
0565 973 ; for error logging)
00A7 C5 02 A4 90 0565 974 MOVVB PT_PRB(R4),UCB$_PT_PP(B5) ; Get data read (also
056B 975 ; keep for err. log.)
056B 976 IOFORK ; Lower IPL
00A4 C5 8000 8F B3 0571 977 BITW #PT_CSR_M_ERROR,UCB$_PT_PPS(R5) ; Error during read?
68 12 0578 978 BNEQ SKIP1 ; Yes, special case
00A4 C5 0800 8F B3 057A 979 BITW #PT_CSR_M_BUSY,UCB$_PT_PPS(R5) ; Busy bit set?
5C 12 0581 980 BNEQ SKIP ; If so, error
00A4 C5 0080 8F B3 0583 981 BITW #PT_CSR_M_DONE,UCB$_PT_PPS(R5) ; Done bit set?
53 13 058A 982 BEQL SKIP ; If not, error
00 68 A5 07 E3 058C 983 BBSC #STARTING,UCB$_DEVSTS(R5),5$ ; Read >= 1 char.
68 A5 20 B3 0591 984 5$: BITW #RD_HDR,UCB$_DEVSTS(R5) ; Reading tape header?
0B 13 0595 985 BEQL 10$ ; No, if eql
00A7 C5 00 91 0597 986 CMPB #NULL,UCB$_PT_PP(B5) ; See if null byte
2A 13 059C 987 BEQL 15$ ; If eql, null
68 A5 20 AA 059E 988 BICW #RD_HDR,UCB$_DEVSTS(R5) ; Not reading header now
68 A5 01 B3 05A2 989 10$: BITW #BIT0,UCB$_DEVSTS(R5) ; Looking for EOF byte?
0A 12 05A6 990 BNEQ 13$ ; No, if neq
00A7 C5 1A 91 05A8 991 CMPB #EOF,UCB$_PT_PP(B5) ; EOF byte found?
03 12 05AD 992 BNEQ 13$ ; No, if neq
FE7A 31 05AF 993 BRW PR_SPERR ; Report EOF condition
78 B5 00A7 C5 90 05B2 994 13$: MOVVB UCB$_PT_PP(B5),@UCB$_SVAPTE(R5) ; Record data read
78 A5 D6 05B8 995 INCL UCB$_SVAPTE(R5) ; Point to where next
05BB 996 ; byte is to be read
7E A5 B7 05BB 997 DECW UCB$_BCNT(R5) ; Decrement byte count
12 13 05BE 998 BEQL 25$ ; All done?
03 64 A5 03 E1 05C0 999 BBC #UCB$_CANCEL,UCB$_STS(R5),15$ ; Check for $CANCEL
FEFC 31 05C5 1000 BRW CANCEL ; $CANCEL if set
05C8 1001 15$: DSBINT UCB$_DIPL(R5) ; Put IPL on stack
FF73 31 05CF 1002 BRW READ ; Read next character
64 00A4 8F AA 05D2 1003 25$: BICW #PT_CSR_M_IE,PT_PRS(R4) ; Clear int. enable bit
00 68 A5 07 E4 05D7 1004 BBSC #STARTING,UCB$_DEVSTS(R5),30$ ; Clear bit for next $QIO
FDF1 31 05DC 1005 30$: BRW FINISH ; And finish up
FE2F 31 05DF 1006 SKIP: BRW PT_ERROR ; Need word offsets
03 68 A5 07 E1 05E2 1007 SKIP1: BBC #STARTING,UCB$_DEVSTS(R5),40$ ; Retry if assume
05E7 1008 ; reader off-line
FE42 31 05E7 1009 BRW PR_SPERR ; Otherwise, assume
05EA 1010 ; no more tape in reader
FE80 31 05EA 1011 40$: BRW COMMON ; Need word offset
05ED 1012 .SBTTL PT_INTERRUPT, Interrupt service routine

```

```
05ED 1014 ;++
05ED 1015 ; PT_INTERRUPT, Analyzes interrupts, processes solicited interrupts
05ED 1016 ;
05ED 1017 ; Functional description:
05ED 1018 ;
05ED 1019 ; This driver is for a multiple-unit controller, and the
05ED 1020 ; two interrupt service routines (PR_INTERRUPT and PT_INTERRUPT)
05ED 1021 ; are used to distinguish which unit caused the interrupt.
05ED 1022 ; Common checking for solicited interrupts occurs after
05ED 1023 ; entry point INTCOM, where control is returned back to the
05ED 1024 ; driver after the appropriate WFIKPC macro.
05ED 1025 ;
05ED 1026 ; Inputs:
05ED 1027 ;
05ED 1028 ; 0(SP) - pointer to the address of the IDB
05ED 1029 ; 4(SP) - saved R0
05ED 1030 ; 8(SP) - saved R1
05ED 1031 ; 12(SP) - saved R2
05ED 1032 ; 16(SP) - saved R3
05ED 1033 ; 20(SP) - saved R4
05ED 1034 ; 24(SP) - saved R5
05ED 1035 ; 28(SP) - saved PSL (program status longword)
05ED 1036 ; 32(SP) - saved PC
05ED 1037 ;
05ED 1038 ; The IDB contains the CSR address and the UCB address.
05ED 1039 ;
05ED 1040 ; Outputs:
05ED 1041 ;
05ED 1042 ; The routine must preserve all registers except R0-R5.
05ED 1043 ;
05ED 1044 ;--
05ED 1045 PT_INTERRUPT:
53 9E D0 05ED 1046 MOVL @ (SP)+,R3 ; Service punch interrupt
05ED 1047 ; Get address of IDB and remove
55 1C A3 D0 05F0 1048 MOVL IDB$L_UCBLST+4(R3),R5 ; pointer from stack.
07 11 05F4 1049 BRB INTCOM ; Punch is unit # 1 (PC1:)
05ED 1050 PR_INTERRUPT: ; Go to common code
53 9E D0 05F6 1051 MOVL @ (SP)+,R3 ; Service reader interrupt
55 18 A3 D0 05F9 1052 MOVL IDB$L_UCBLST(R3),R5 ; Get IDB address
54 63 D0 05FD 1053 INTCOM: MOVL IDB$L_CSR(R3),R4 ; Reader is unit # 0 (PC0:)
01 E5 0600 1054 BBCC #UCB$V_INT,- ; Get CSR address
64 A5 0602 1055 UCB$W_STS(R5),- ; If device does not expect
07 0604 1056 UNSOL_INTERRUPT ; interrupt, dismiss it.
0605 1057 ; This is a solicited interrupt. Restore control to the main driver.
53 10 A5 D0 0605 1058 RESTORE_DRIVER: ; Jump to main driver code.
OC B5 16 0609 1059 MOVL UCB$L_FR3(R5),R3 ; Restore driver's R3
060C 1060 JSB @UCB$L_FPC(R5) ; Call driver at interrupt
060C 1061 ; wait address.
060C 1062 ; Dismiss the interrupt.
50 8E 7D 060C 1063 UNSOL_INTERRUPT: ; Dismiss unsolicited interrupt.
52 8E 7D 060F 1064 MOVQ (SP)+,R0 ; Restore R0-R5
54 8E 7D 0612 1065 MOVQ (SP)+,R2 ; This is faster than a
02 0615 1066 MOVQ (SP)+,R4 ; POPR #AM<R0,R1,R2,R3,R4,R5>
0616 1067 REI ; Return from interrupt.
0616 1068 .SBTTL PT_CANCEL, Cancel I/O routine
```

```
0616 1070
0616 1071 ;++
0616 1072 ; PT_CANCEL, Cancels an I/O operation in progress
0616 1073 ;
0616 1074 ; Functional description:
0616 1075 ;
0616 1076 ; This routine tests to see if the punch is being deallocated
0616 1077 ; or deassigned by checking the reference count field in the
0616 1078 ; UCB. It checks to see if a trailer needs to be punched,
0616 1079 ; and if so, then a trailer is punched (possibly with an
0616 1080 ; EOF byte). Also, regardless of whether a
0616 1081 ; header has been punched, the default number of null bytes to
0616 1082 ; punch in a header/trailer is restored to NULL_BYTES. The
0616 1083 ; flag indicating that a header has not been punched is cleared
0616 1084 ; for the next I/O operation, as is the bit indicating that
0616 1085 ; the current $QIO is the first $QIO for the user (since the
0616 1086 ; channel was assigned to the device). Other flags are
0616 1087 ; reset to their initial state for the next user.
0616 1088 ;
0616 1089 ; This routine calls IOC$CANCELIO to set the cancel bit in the
0616 1090 ; UCB status word if:
0616 1091 ;
0616 1092 ; the device is busy,
0616 1093 ; the IRP's process ID matches the cancel process ID,
0616 1094 ; the IRP channel matches the cancel channel.
0616 1095 ;
0616 1096 ; The routine then does device-dependent cancel I/O fixups by
0616 1097 ; clearing the interrupt enable bit for either the reader or
0616 1098 ; the punch, as appropriate.
0616 1099 ;
0616 1100 ; Inputs:
0616 1101 ;
0616 1102 ; R2 - negated value of the channel index number
0616 1103 ; R3 - address of the current IRP (I/O request packet)
0616 1104 ; R4 - address of the PCB (process control block) for the
0616 1105 ; process canceling I/O
0616 1106 ; R5 - address of the UCB (unit control block)
0616 1107 ;
0616 1108 ; Outputs:
0616 1109 ;
0616 1110 ; The routine must preserve all registers except R0-R3.
0616 1111 ;
0616 1112 ; The routine sets the UCB$M_CANCEL bit in UCB$W_STS.
0616 1113 ;
0616 1114 ; The routine restores UCB fields to their original state.
0616 1115 ;
0616 1116 ;--
0616 1117
0616 1118 PT_CANCEL ; Cancel an I/O operation
0616 1119 .ENABL LSB
00A8 C5 5C A5 B1 0616 1120 CMPW UCB$W_REFC(R5),UCB$W_MY_REFCNT(R5) ; Deal or deassign?
5D 13 061C 1121 BEQL 10$ ; No, if eq1
061E 1122 ; Note that the count is not decremented..
061E 1123 ; Therefore the user will not get two
061E 1124 ; trailers if he allocates the device and
061E 1125 ; assigns a channel; the deallocate will not
061E 1126 ; match the count field (like a $CANCEL).
```

```
00A0 C5 54 D0 061E 1127 MOVL R4,UCB$L_SAV_R4(R5) ; Save R4 since not CSR
00 68 A5 06 E4 0623 1128 BBSC #FRST_QIO,UCB$W_DEVSTS(R5),3$ ; Clear read header bit
68 A5 08 B3 0628 1129 3$: BITW #PNCH,UCB$W_DEVSTS(R5) ; Trailer needed?
12 062C 1130 BNEQ 4$ ; Yes, if neq
34 38 A5 1A E0 062E 1131 BBS #DEV$V_IDV,UCB$L_DEVCHAR(R5),8$ ; Reader?, go away
68 A5 01 B3 0633 1132 BITW #BIT0,UCB$W_DEVSTS(R5) ; User want EOF byte?
2E 13 0637 1133 BEQL 8$ ; No, if eq
44 A5 01 9A 0639 1134 MOVZBL #1,UCB$L_DEVDEPEND(R5) ; Punching only EOF byte
68 A5 14 A8 063D 1135 BISW #END!START,UCB$W_DEVSTS(R5) ; Flag punching trailer tape
50 24 A5 D0 0641 1136 MOVL UCB$L_CRB(R5),R0 ; Get CRB
54 2C B0 D0 0645 1137 MOVL @CRB$L_INTD+VEC$L_IDB(R0),R4 ; Get CSR address
04 A4 0040 8F A8 0649 1138 BISW #PT_STS_M_IE,PT_PPS(R4) ; Set int. enable bit
FCA8 31 064F 1139 BRW PUNEEOF ; Punch EOF byte
68 A5 14 A8 0652 1140 4$: BISW #END!START,UCB$W_DEVSTS(R5) ; Flag punching trailer tape
00 68 A5 01 E4 0656 1141 BBSC #PUNCH_HDR,UCB$W_DEVSTS(R5),5$ ; Enable trailer punching
FC3A 31 065B 1142 5$: BRW PNCH_HDR ; Punch trailer tape
68 A5 08 AA 065E 1143 INCAN: BICW #PNCH,UCB$W_DEVSTS(R5) ; Clear flag for next operation
54 00A0 C5 D0 0662 1144 MOVL UCB$L_SAV_R4(R5),R4 ; Restore R4 if changed
00 68 A5 01 E4 0667 1145 8$: BBSC #PUNCH_HDR,UCB$W_DEVSTS(R5),9$ ; Clear for next op.
68 A5 0311 8F AA 066C 1146 9$: BICW #BIT0!BIT8!BIT9!END,UCB$W_DEVSTS(R5) ; Clear for next user
42 A5 32 B0 0672 1147 MOVW #NULL_BYTES,UCB$W_DEVBUFSIZ(R5) ; Restore def nullbytes
00 68 A5 07 E4 0676 1148 BBSC #STARTING,UCB$W_DEVSTS(R5),10$ ; Clear for next oper.
00000000 GF 16 067B 1149 10$: JSB G^IOC$CANCELIO ; Set cancel bit if appropriate.
1A 64 A5 03 E1 0681 1150 BBC #UCB$V_CANCEL,UCB$W_STS(R5),OKAY ; If clear, return
0686 1151 .DSABL LSB
0686 1152
0686 1153 ;
0686 1154 ; Device-dependent cancel operations go next. The appropriate int. ena.
0686 1155 ; bit is cleared. This subroutine is called from several timeout and
0686 1156 ; powerfail recovery routines; it must be very careful NOT to issue
0686 1157 ; a macro (or call on a system routine) that returns control to the
0686 1158 ; caller's caller (since the stack contains "local" return info).
0686 1159 ;
51 24 A5 D0 0686 1160 SETOFF: MOVL UCB$L_CRB(R5),R1 ; Get address of CRB
51 2C B1 D0 068A 1161 MOVL @CRB$L_INTD+VEC$L_IDB(R1),R1 ; Get address of CSR
08 38 A5 1A E0 068E 1162 BBS #DEV$V_IDV,UCB$L_DEVCHAR(R5),5$ ; Test for reader
04 A1 0040 8F AA 0693 1163 BICW #PT_STS_M_IE,PT_PPS(R1) ; Turn off punch int. ena. bit
05 11 0699 1164 BRB OKAY ; And finish up
61 0040 8F AA 069B 1165 5$: BICW #PT_CSR_M_IE,PT_PPS(R1) ; Turn off reader int. ena. bit
06A0 1166 OKAY:
05 06A0 1167 RSB ; Return
06A1 1168 .SBTTL PT_REG_DUMP, Device register dump routine
```

```

06A1 1170
06A1 1171 ;++
06A1 1172 ; PT_REG_DUMP, Dumps the contents of device registers to a buffer
06A1 1173 ;
06A1 1174 ; Functional description:
06A1 1175 ;
06A1 1176 ;         Writes the number of device registers, and their current
06A1 1177 ;         contents into a diagnostic or error buffer. Reader and
06A1 1178 ;         punch both have two registers (a status register, and a
06A1 1179 ;         data register).
06A1 1180 ;
06A1 1181 ; Inputs:
06A1 1182 ;
06A1 1183 ;         R0         - address of the output buffer
06A1 1184 ;         R4         - address of the CSR (controller status register)
06A1 1185 ;         R5         - address of the UCB (unit control block)
06A1 1186 ;
06A1 1187 ; Outputs:
06A1 1188 ;
06A1 1189 ;         The routine must preserve all registers except R1-R3.
06A1 1190 ;
06A1 1191 ;         The output buffer contains the current contents of the device
06A1 1192 ;         registers. R0 contains the address of the next empty longword in
06A1 1193 ;         the output buffer.
06A1 1194 ;
06A1 1195 ;--
06A1 1196
06A1 1197 PT_REG_DUMP:
80      02      9A 06A1 1198      MOVZBL #PT_NUM_REGS,(R0)+ ; Dump device registers
80      00A4 C5 3C 06A4 1199      MOVZWL UCB$W_PT_PPS(R5),(R0)+ ; Store device register count
80      00A7 C5 9A 06A9 1200      MOVZBL UCB$B_PT_PPB(R5),(R0)+ ; Store status register
05      06AE 1201      RSB ; Store data register
06AF 1202 ; Return
06AF 1203 ;++
06AF 1204 ; Label that marks the end of the driver
06AF 1205 ;--
06AF 1206
06AF 1207 PT_END: ; Last location in driver
06AF 1208      .END

```

\$\$\$	= 00000020	R	02	IDB\$_CSR	= 00000000		
\$\$OP	= 00000002			IDB\$_UCBLST	= 00000018		
ACP_FDT	00000147	R	03	INCAN	0000065E	R	03
AFTINT	00000396	R	03	INTCOM	000005FD	R	03
AT\$_UBA	= 00000001			IO\$_FCODE	= 00000006		
BACK	00000331	R	03	IO\$_VFCODE	= 00000000		
BIT0	= 00000001			IO\$_ACPCONTROL	= 00000038		
BIT8	= 00000100			IO\$_CREATE	= 00000033		
BIT9	= 00000200			IO\$_DEACCESS	= 00000034		
BUF_OVR_HD	= 00000000			IO\$_DELETE	= 00000035		
CANCEL	000004C4	R	03	IO\$_MODIFY	= 00000036		
COMMON	0000046D	R	03	IO\$_MOUNT	= 00000039		
COMPLETE_IO	000003D9	R	03	IO\$_READBLK	= 00000021		
COPY	00000447	R	03	IO\$_READPBLK	= 0000000C		
CRB\$_INTD	= 00000024			IO\$_READPROMPT	= 00000037		
CRB\$_INTD2	= 00000048			IO\$_READVBLK	= 00000031		
ddb\$_DDT	= 0000000C			IO\$_VIRTUAL	= 0000003F		
DEV\$_AVL	= 00040000			IO\$_WRITEBLK	= 00000020		
DEV\$_ELG	= 00400000			IO\$_WRITEOF	= 00000028		
DEV\$_IDV	= 04000000			IO\$_WRITEPBLK	= 0000000B		
DEV\$_ODV	= 08000000			IO\$_WRITEVBLK	= 00000030		
DEV\$_REC	= 00000001			IOC\$CANCELIO	*****	X	03
DEV\$_V_IDV	= 0000001A			IOC\$MNTVER	*****	X	03
DEV\$_V_ODV	= 0000001B			IOC\$REQCOM	*****	X	03
DON	00000444	R	03	IOC\$RETURN	*****	X	03
DPT\$_C_LENGTH	= 00000038			IOC\$WFIKPC	*****	X	03
DPT\$_C_VERSION	= 00000004			IPL\$_POWER	= 0000001F		
DPT\$_INITAB	00000038	R	02	IPL\$_SYNCH	= 00000008		
DPT\$_REINITAB	00000059	R	02	IRP\$_MEDIA	= 00000038		
DPT\$_TAB	00000000	R	02	IRP\$_SVAPTE	= 0000002C		
DSABL	00000519	R	03	IRP\$_FCODE	= 00000006		
DYN\$_C_CRB	= 00000005			IRP\$_VFCODE	= 00000000		
DYN\$_C_DDB	= 00000006			IRP\$_V_FUNC	= 00000001		
DYN\$_C_DPT	= 0000001E			IRP\$_W_BCNT	= 00000032		
DYN\$_C_UCB	= 00000010			IRP\$_W_BOFF	= 00000030		
EMB\$_L_DV_REGS	= 0000004E			IRP\$_W_FUNC	= 00000020		
END	= 00000010			IRP\$_W_STS	= 0000002A		
EOF	= 0000001A			JIB\$_L_BYTCNT	= 00000020		
ERBFSIZE	= 0000000C			LOOP_CNT	= 0000000F		
ERL\$_DEVICERR	*****	X	03	MAIN	0000051D	R	03
ERL\$_DEVICTMO	*****	X	03	MAINLP	0000036A	R	03
EXE\$_ABORTIO	*****	X	03	MASKH	= 00B80000		
EXE\$_ALLOCBUF	*****	X	03	MASKL	= 00000000		
EXE\$_BUFRQUOTA	*****	X	03	MSG\$_DEVOFFLIN	*****	X	03
EXE\$_FINISHIOC	*****	X	03	MY_CLASS	= 00000064		
EXE\$_IOFORK	*****	X	03	MY_TYPE	= 00000064		
EXE\$_QIOACPPKT	*****	X	03	NO_PRIV	0000015C	R	03
EXE\$_QIODRVPKT	*****	X	03	NT_MNT	00000157	R	03
EXE\$_READCHK	*****	X	03	NT_WRT	00000177	R	03
EXE\$_SNDEVMMSG	*****	X	03	NUL	00000310	R	03
EXE\$_WRITECHK	*****	X	03	NULL	= 00000000		
FINISH	000003D0	R	03	NULLP	00000315	R	03
FIX	00000511	R	03	NULL_BYTES	= 00000032		
FOR_PNCH	00000183	R	03	OKAY	000006A0	R	03
FRST_QIO	= 00000006			P1	= 00000000		
FUNCTAB_LEN	= 00000064			P2	= 00000004		
GO	00000552	R	03	P3	= 00000008		



PCDRIVER  
Symbol table

- VAX/VMS PAPER TAPE READER/PUNCH DRIVER 12-SEP-1984 17:59:27 VAX-11 Macro V03-01 Page 29  
22-AUG-1984 20:06:00 WORK2:[COURSE.DRIVER]PCDRIVER.MAR;(1)

P4	= 0000000C			SKIP1	000005E2 R	03
P5	= 00000010			SKIP2	0000045C R	03
P6	= 00000014			SS\$_CANCEL	= 00000830	
PC\$DDT	= 00000000	RG	03	SS\$_DEVNOTMOUNT	= 0000007C	
PCB\$_JIB	= 00000080			SS\$_DRVERR	= 0000008C	
PCB\$_Q_PRIV	= 00000084			SS\$_ENDOFFILE	= 00000870	
PNCH	= 00000008			SS\$_ILLIOFUNC	= 000000F4	
PNCH_HDR	00000298	R	03	SS\$_NOPRIV	= 00000024	
POWER_FAIL	000003FC	R	03	SS\$_NORMAL	= 00000001	
POW_FAIL	000003E8	R	03	SS\$_TAPEPOSLOST	= 00000224	
PR\$_IPL	*****	X	03	SS\$_TIMEOUT	= 0000022C	
PRV\$_MOUNT	= 00000011			START	= 00000004	
PR_FDT_ROUTINE	000001A4	R	03	STARTING	= 00000007	
PR_INTERRUPT	000005F6	R	03	SY\$_GL_OPRMBX	*****	X 03
PR_POWFAIL	000003F9	R	03	TO_ACP	00000173	R 03
PR_SPERR	0000042C	R	03	UCB\$_DEVCLASS	= 00000040	
PR_START	000004D7	R	03	UCB\$_DEVTYPE	= 00000041	
PR_TIMEOUT	00000451	R	03	UCB\$_DIPL	= 0000005E	
PT_CANCEL	00000616	R	03	UCB\$_FIPL	= 0000000B	
PT_CONTROL_INIT	0000009C	R	03	UCB\$_PT_OFLCNT	000000A6	
PT_CSR_M_BUSY	= 00000800			UCB\$_PT_PPB	000000A7	
PT_CSR_M_DONE	= 00000080			UCB\$_PT_UCBLEN	000000AA	
PT_CSR_M_ERROR	= 00008000			UCB\$_CRB	= 00000024	
PT_CSR_M_IE	= 00000040			UCB\$_DEVCHAR	= 00000038	
PT_CSR_M_RDENA	= 00000001			UCB\$_DEVDEPEND	= 00000044	
PT_END	000006AF	R	03	UCB\$_DPC	= 0000009C	
PT_ERROR	00000411	R	03	UCB\$_FPC	= 0000000C	
PT_FDT_ROUTINE	000000BC	R	03	UCB\$_FR3	= 00000010	
PT_FUNC_TABLE	00000038	R	03	UCB\$_IRP	= 00000058	
PT_INTERRUPT	000005ED	R	03	UCB\$_SAV_R4	000000A0	
PT_NUM_REGS	= 00000002			UCB\$_SVAPTE	= 00000078	
PT_PPB	00000006			UCB\$_VCB	= 00000034	
PT_PPS	00000004			UCB\$_M_ONLINE	= 00000010	
PT_PRB	00000002			UCB\$_V_CANCEL	= 00000003	
PT_PRS	00000000			UCB\$_V_INT	= 00000001	
PT_PUNCH	0000028B	R	03	UCB\$_V_MOUNTING	= 00000009	
PT_REG_DUMP	000006A1	R	03	UCB\$_V_POWER	= 00000005	
PT_RESTART	000003E5	R	03	UCB\$_V_BCNT	= 0000007E	
PT_START	00000216	R	03	UCB\$_V_DEVBUFSIZ	= 00000042	
PT_STS_M_ERROR	= 00008000			UCB\$_V_DEVSTS	= 00000068	
PT_STS_M_IE	= 00000040			UCB\$_V_FUNC	= 0000009A	
PT_STS_M_READY	= 00000080			UCB\$_V_MY_REFCNT	000000A8	
PT_TIMEOUT	0000045F	R	03	UCB\$_V_PT_PPS	000000A4	
PT_TIMEOUT_SEC	= 0000000A			UCB\$_V_REFC	= 0000005C	
PT_UNIT_INIT	0000009D	R	03	UCB\$_V_STS	= 00000064	
PUNCH_HDR	= 00000001			UCB\$_V_UNIT	= 00000054	
PUNEOF	000002FA	R	03	UNSOL_INTERRUPT	0000060C	R 03
PUN_OT	000004A0	R	03	VCB\$_TRANS	= 0000000C	
Q_TO_ACP	00000162	R	03	VEC\$_IDB	= 00000008	
RD_HDR	= 00000020			VEC\$_INITIAL	= 0000000C	
READ	00000545	R	03	WAIT	00000324	R 03
RESTORE_DRIVER	00000605	R	03			
SETOFF	00000686	R	03			
SET_FDT	00000199	R	03			
SHORT_WAIT	= 00000002					
SIZ...	= 00000001					
SKIP	000005DF	R	03			

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	000000AA ( 170.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	0000006E ( 110.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	000006AF ( 1711.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	122	00:00:00.33	00:00:01.51
Command processing	137	00:00:00.46	00:00:01.04
Pass 1	1976	00:00:31.64	00:00:38.03
Symbol table sort	82	00:00:03.71	00:00:03.81
Pass 2	819	00:00:05.70	00:00:06.79
Symbol table output	31	00:00:00.24	00:00:00.37
Psect synopsis output	7	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	3180	00:00:42.11	00:00:51.58

The working set limit was 500 pages.  
146981 bytes (288 pages) of virtual memory were used to buffer the intermediate code.  
There were 130 pages of symbol table space allocated to hold 2469 non-local and 50 local symbols.  
1208 source lines were read in Pass 1, producing 20 object records in Pass 2.  
45 pages of virtual memory were used to define 42 macros.

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

Macro library name	Macros defined
SYS\$SYSROOT:[SYSLIB]LIB.MLB;1	29
SYS\$SYSROOT:[SYSLIB]STARLET.MLB;1	10
TOTALS (all libraries)	39

2705 GETS were required to define 39 macros.

There were no errors, warnings or information messages.

MAC/LIS PCDRIVER+SYS\$LIBRARY:LIB/LIB



```
100 $!
200 $!
300 $SET VERIFY
400 $MAC/LIS PCDRIVER+SYS$LIBRARY:LIB/LIB
500 $LINK/NOTRACE/MAP PCDRIVER/OPT,PCDRIVER,SYS$SYSTEM:SYS.STB/SEL
600 $SET NOVERIFY
```

PCDRIVER.COM

100 !  
200 BASE=0

PCDRIVER.OPT

```
100 $!  
200 $!  
300 $MCR SYSGEN  
400 LOAD SYS$SYSDISK:[COURSE.DRIVER]PCDRIVER  
500 CONNECT PCA0/ADAPTER=3-  
600 /NUMVEC=2-  
700 /VECTOR=%070-  
800 /CSR=%0777550-  
900 /DRIVER=PCDRIVER  
1000 CONNECT PCA1/ADAPTER=3-  
1100 /VECTOR=%074-  
1200 /DRIVER=PCDRIVER  
1300 SHO /DEV=PC  
1400 EXIT
```

```
100 C
200 C
300 EXTERNAL IO$_READVBLK
400 CHARACTER*36 ALPHA
500 INTEGER*4 SYSS$ASSIGN,SYSS$QIOW,IOSB(2)
600 INTEGER*2 IOSBW(4)
700 EQUIVALENCE (IOSB(1),IOSBW(1))
800 10 FORMAT(' COMPLETED QIOW ',I2,' -- ',I4,' BYTES READ')
900 20 FORMAT(' ERROR ID = ',I4,' FOR SS$_VALUE = ',I4)
1000 25 FORMAT(' IOSBW(2) VALUE = ',I2)
1100 30 FORMAT(' BEFORE ASSIGNMENT MADE')
1200 40 FORMAT(' ASSIGNMENT SUCCESSFUL')
1300 50 FORMAT(' DATA READ = ',A)
1400 WRITE (6,30)
1500 J=0
1600 II=SYSS$ASSIGN('PCA0:',IC,,)
1700 IF(.NOT.II)GOTO 200
1800 WRITE (6,40)
1900 DO 100 I=1,4
2000 J=1
2100 II=SYSS$QIOW(,%VAL(IC),IO$_READVBLK,IOSB,,,%REF(ALPHA),
2200 1%VAL(36),,,,,)
2300 IF(.NOT.II)GOTO 200
2400 J=2
2500 IF(IOSBW(1).NE.1)GOTO 200
2600 WRITE (6,10)I,IOSBW(2)
2700 WRITE (6,50)ALPHA
2800 100 CONTINUE
2900 GOTO 300
3000 200 WRITE (6,20)J,II
3100 WRITE (6,25)IOSBW(2)
3200 IF(.NOT.II)CALL SYS$EXIT(%VAL(II))
3300 CALL SYS$EXIT(%VAL(IOSBW(1)))
3400 300 STOP
3500 END
```

```

100 C
200 C
300 EXTERNAL IO$ WRITEVBLK,IO$ READVBLK
400 CHARACTER*36 ALPHA,BETA,GAMMA(18)
500 INTEGER*4 SYS$ASSIGN,SYS$QIOW,IOSB(2),SYS$QIO,IOSB1(2)
600 INTEGER*2 IOSBW(4),IOSBW1(4)
700 EQUIVALENCE (IOSB(1),IOSBW(1)),(IOSB1(1),IOSBW1(1))
800 10 FORMAT(' IOSBW(2) AND IOSBW1(2) = ',I2,' ',I2)
900 15 FORMAT(' COMPLETED READ ',I2)
1000 20 FORMAT(' ERROR ID = ',I4)
1100 25 FORMAT(' DATA READ = ',A)
1200 30 FORMAT(' BEFORE ASSIGNMENTS MADE')
1300 40 FORMAT(' ASSIGNMENTS SUCCESSFUL')
1400 ALPHA='ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
1500 WRITE (6,30)
1600 J=0
1700 II=SYS$ASSIGN('PCA0:',IC,,)
1800 IF(.NOT.II)GOTO 200
1900 J=5
2000 II=SYS$ASSIGN('PCA1:',IC2,,)
2100 IF(.NOT.II)GOTO 200
2200 WRITE (6,40)
2300 J=1
2400 IP=3
2500 IR=18
2600 C IF CHANGE IR, CHANGE GAMMA(IR) TOO IN CHAR STATEMENT
2700 IRINIT=IR
2800 K=0
2900 95 II=SYS$QIO(%VAL(1),%VAL(IC2),IO$ WRITEVBLK,IOSB,,,%REF(ALPHA),
3000 1%VAL(36),,,,,)
3100 IF(.NOT.II)GOTO 200
3200 J=2
3300 IF (K .NE. 0) GOTO 99
3400 K=1
3500 98 II=SYS$QIO(,%VAL(IC),IO$ READVBLK,IOSB1,,,%REF(BETA),
3600 1%VAL(36),,,,,)
3700 IF(.NOT.II) GOTO 200
3800 J=4
3900 99 IF (IOSBW1(1).NE.1)GOTO 100
4000 INDX=IRINIT+1-IR
4100 GAMMA(INDX)=BETA
4200 IR=IR-1
4300 IF(IR .GT. 0) GOTO 98
4400 IOSBW1(1)=0
4500 GOTO 150
4600 100 IF (IOSBW(1) .NE. 1) GOTO 99
4700 IP=IP-1
4800 IF (IP .GT. 0) GOTO 95
4900 IOSBW(1)=0
5000 150 IF ((IP .EQ. 0) .AND. (IR .EQ. 0)) GOTO 300
5100 IF ((IP .LT. 0) .OR. (IR .LT. 0)) GOTO 400
5200 GOTO 99
5300 200 WRITE (6,20)J
5400 WRITE (6,10)IOSBW(2),IOSBW1(2)
5500 IF(.NOT.II)CALL SYS$EXIT(%VAL(II))
5600 CALL SYS$EXIT(%VAL(IOSBW(1)))
5700 300 DO 350 I=1,IRINIT-1
5800 WRITE (6,15) I
5900 WRITE (6,25) GAMMA(I)

```



```
6000 350 CONTINUE
6100 STOP
6200 400 WRITE (6,20) IP
6300 WRITE (6,20) IR
6400 END
```

```

100 C
200 C
300 EXTERNAL IOS$_WRITEVBLK,IOS$_MODIFY,IOS$_DELETE,IOS$_DEACCESS
400 CHARACTER*36 ALPHA
500 INTEGER*4 SYS$ASSIGN,SYS$QIOW,IOSB(2)
600 INTEGER*2 IOSBW(4)
700 EQUIVALENCE (IOSB(1),IOSBW(1))
800 10 FORMAT(' COMPLETED QIOW ',I2,' -- ',I4,' BYTES PUNCHED')
900 20 FORMAT(' ERROR ID = ',I4)
1000 25 FORMAT(' NEW MODE VALUE = ',I4)
1100 30 FORMAT(' BEFORE ASSIGNMENT MADE')
1200 40 FORMAT(' ASSIGNMENT SUCCESSFUL')
1300 ALPHA='ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
1400 WRITE (6,30)
1500 J=0
1600 II=SYS$ASSIGN('PCAL:',IC,,)
1700 IF(.NOT.II)GOTO 200
1800 WRITE (6,40)
1900 J=3
2000 II=SYS$QIOW(,%VAL(IC),IOS$_DEACCESS,IOSB,,,,,,,,)
2100 IF (.NOT. II) GOTO 200
2200 WRITE (6,25) IOSB(2)
2300 J=4
2400 II=SYS$QIOW(,%VAL(IC),IOS$_MODIFY,,,,,%VAL(70),,,,,)
2500 IF (.NOT. II) GOTO 200
2600 J=5
2700 II=SYS$QIOW(,%VAL(IC),IOS$_DEACCESS,IOSB,,,,,,,,)
2800 IF (.NOT. II) GOTO 200
2900 WRITE (6,25) IOSB(2)
3000 DO 100 I=1,4
3100 J=1
3200 II=SYS$QIOW(,%VAL(IC),IOS$_WRITEVBLK,IOSB,,,%REF(ALPHA),
3300 1,%VAL(36),,,,,)
3400 IF(.NOT.II)GOTO 200
3500 J=2
3600 IF(IOSBW(1).NE.1)GOTO 200
3700 WRITE (6,10)I,IOSBW(2)
3800 100 CONTINUE
3900 J=5
4000 II=SYS$QIOW(,%VAL(IC),IOS$_DELETE,,,,,,,,)
4100 IF (.NOT. II) GOTO 200
4200 J=6
4300 II=SYS$QIOW(,%VAL(IC),IOS$_DEACCESS,IOSB,,,,,,,,)
4400 IF (.NOT. II) GOTO 200
4500 WRITE (6,25) IOSB(2)
4600 GOTO 300
4700 200 WRITE (6,20)J
4800 IF(.NOT.II)CALL SYS$EXIT(%VAL(II))
4900 CALL SYS$EXIT(%VAL(IOSBW(1)))
5000 300 STOP
5100 END

```

**PCDRIVER.LIS  
PCDRIVER.COM  
PCDRIVER.OPT  
PCLOAD.COM  
ATTN.FOR**

(1)	40	External and local symbol definitions
(1)	96	Standard tables
(1)	165	PC_CONTROL_INIT, Controller initialization routine
(1)	194	PC_UNIT_INIT, Unit initialization routine
(1)	231	CSR_FDT_ROUTINE, Return CSR Address
(1)	268	SET_AST_FDT, Reader Attn. AST
(1)	312	PUNC_AST_FDT, Punch Attn. AST
(1)	356	PC_START, Start I/O Routine
(1)	366	ASTINT, Interrupt service routines
(1)	419	PC_END, End of driver

```
0000 1 .TITLE PCDRIVER - VAX/VMS PAPER TAPE READER/PUNCH DRIVER
0000 2 .IDENT /V02/
0000 3
0000 4 ;++
0000 5 ;
0000 6 ; FACILITY:
0000 7 ;
0000 8 ; VAX/VMS Paper Tape Reader/Punch driver
0000 9 ;
0000 10 ; ABSTRACT:
0000 11 ;
0000 12 ; This driver allows the user to control the paper tape reader/
0000 13 ; punch by using the set attention AST scheme. The following
0000 14 ; function codes are accepted:
0000 15 ;
0000 16 ; IO$_SENSECHAR--takes no arguments, and returns the address
0000 17 ; of the CSR in the high-order longword of the IOSB.
0000 18 ;
0000 19 ; IO$_SETMODE----takes up to three parameters:
0000 20 ;
0000 21 ; P1 = address of AST routine to be entered when the paper
0000 22 ; tape reader interrupts (P1 is required).
0000 23 ; P2 = parameter to be passed to the AST routine
0000 24 ; specified in P1 (P2 is optional).
0000 25 ; P3 = access mode in which the AST routine specified in
0000 26 ; P1 is to run (P3 is optional).
0000 27 ;
0000 28 ; IO$_SETCHAR----identical to IO$_SETMODE except that the AST
0000 29 ; routine specified in P1 will be entered if the punch
0000 30 ; generates an interrupt (instead of the reader).
0000 31 ;
0000 32 ; AUTHOR:
0000 33 ;
0000 34 ; Vik Muiznieks July--1979
0000 35 ;
0000 36 ; History
0000 37 ; Fred Marsh August 1894
0000 38 ; Added $DYNDEF to the symbol definitions
0000 39 ;--
0000 40 .SBTTL External and local symbol definitions
```

```

0000 42
0000 43 ;
0000 44 ; External symbols
0000 45 ;
0000 46
0000 47 $CRBDEF ; Channel request block
0000 48 $DCDEF ; Device classes and types
0000 49 $DDBDEF ; Device data block
0000 50 $DEVDEF ; Device characteristics
0000 51 $IDBDEF ; Interrupt data block
0000 52 $IODEF ; I/O function codes
0000 53 $IPLDEF ; Hardware IPL definitions
0000 54 $IRPDEF ; I/O request packet
0000 55 $SSDEF ; System status codes
0000 56 $UCBDEF ; Unit control block
0000 57 $VECDEF ; Interrupt vector block
0000 58 $DYNDEF ;
0000 59 ;
0000 60 ; Local symbols
0000 61 ;
0000 62
0000 63 ;
0000 64 ; Argument list (AP) offsets for device-dependent $QIO parameters
0000 65 ; (Only P1, P2, and P3 have meaning for this driver.)
0000 66 ;
0000 67
00000000 0000 68 P1 = 0 ; First $QIO parameter
00000004 0000 69 P2 = 4 ; Second $QIO parameter
00000008 0000 70 P3 = 8 ; Third $QIO parameter
0000000C 0000 71 P4 = 12 ; Fourth $QIO parameter
00000010 0000 72 P5 = 16 ; Fifth $QIO parameter
00000014 0000 73 P6 = 20 ; Sixth $QIO parameter
0000 74
0000 75 ;
0000 76 ; Other constants
0000 77 ;
0000 78
000000FC 0000 79 MY_CLASS=252 ; My own device class number
0000 80
0000 81 ;
0000 82 ; Definitions that follow the standard UCB fields
0000 83 ;
0000 84
0000 85 $DEFINI UCB ; Start of UCB definitions
0000 86
00000090 0000 87 .=UCB$K_LENGTH ; Position to end of UCB
0090 88
0090 89 $DEF UCB$L_ASTHEAD .BLKL 1 ; Listhead for reader attn. AST
0094 90
0094 91 $DEF UCB$L_PUNCHD .BLKL 1 ; Listhead for punch attn. AST
0098 92
0098 93 $DEF UCB$K_PC_UCBLEN ; Length of extended UCB
0098 94
0098 95 $DEFEND UCB ; End of UCB definitions
0000 96 .SBTTL Standard tables

```

```
0000 98
0000 99 ;
0000 100 ; Driver prologue table
0000 101 ;
0000 102
0000 103 DPTAB - ; DPT-creation macro
0000 104 END=PC_END,- ; End of driver label
0000 105 ADAPTER=UBA,- ; Adapter type
0000 106 UCBSIZE=<UCB$K_PC_UCBLEN>,- ; Length of UCB
0000 107 NAME=PCDRIVER ; Driver name
0038 108 DPT_STORE INIT ; Start of load
0038 109 ; initialization table
0038 110 DPT_STORE UCB,UCB$B_FIPL,B,8 ; Device fork IPL
003C 111 DPT_STORE UCB,UCB$B_DIPL,B,20 ; Device int. IPL (BR4)
0040 112 DPT_STORE UCB,UCB$$_DEVCHAR,L,<- ; Device characteristics
0040 113 DEV$M_AVL!- ; e.g., dev. available
0040 114 DEV$M_REC!- ; record oriented
0040 115 DEV$M_IDV!- ; input device
0040 116 DEV$M_ODV> ; output device
0047 117 DPT_STORE UCB,UCB$B_DEVCLASS,B,MY_CLASS ; store my device class
004B 118
004B 119 DPT_STORE REINIT ; Start of reload
004B 120 ; initialization table
004B 121 DPT_STORE DDB,DOB$_DDT,D,PC$DDT ; Address of DDT
0050 122 DPT_STORE CRB,CRB$_INTD+4,D,- ; Address of reader int.
0050 123 ASTINT ; service routine
0055 124 DPT_STORE CRB,CRB$_INTD2+4,D,- ; Address of punch int.
0055 125 PUNC_INT ; service routine
005A 126 DPT_STORE CRB,- ; Address of controller
005A 127 CRB$_INTD+VEC$_INITIAL,- ; initialization routine
005A 128 D,PC_CONTROL_INIT
005F 129 DPT_STORE CRB,- ; Address of device
005F 130 CRB$_INTD+VEC$_UNITINIT,- ; unit initialization
005F 131 D,PC_UNIT_INIT ; routine
0064 132
0064 133 DPT_STORE END ; End of initialization
0000 134 ; tables
0000 135
0000 136 ;
0000 137 ; Driver dispatch table
0000 138 ;
0000 139
0000 140 DDTAB - ; DDT-creation macro
0000 141 DEVNAM=PC,- ; Name of device
0000 142 START=PC_START,- ; Start I/O routine
0000 143 FUNCTB=PC_FUNCTABLE,- ; FDT address
0000 144 CANCEL=+IOC$CANCELIO ; Cancel I/O routine
0038 145
0038 146 ;
0038 147 ; Function dispatch table
0038 148 ;
0038 149
0038 150 PC_FUNCTABLE: ; FDT for driver
0038 151 FUNCTAB ,- ; Valid I/O functions
0038 152 <SENSECHAR,- ; Get CSR address func.
0038 153 SETCHAR,- ; Punch attn. AST func.
0038 154 SETMODE> ; Reader attn. AST func
```

0040	155	FUNCTAB	,-	; Buffered functions
0040	156		<SENSECHAR,-	; Get CSR address
0040	157		SETCHAR,-	; Punch attn. AST
0040	158		SETMODE>	; Reader attn. AST
0048	159	FUNCTAB	CSR_FDT_ROUTINE,-	; FDT get CSR routine
0048	160		<SENSECHAR>	
0054	161	FUNCTAB	PUNC_AST_FDT,-	; FDT punch attn.
0054	162		<SETCHAR>	; AST routine
0060	163	FUNCTAB	SET_AST_FDT,-	; FDT reader attn.
0060	164		<SETMODE>	; AST routine
006C	165	.SBTTL	PC_CONTROL_INIT,	Controller initialization routine



```
006C 167
006C 168 ;++
006C 169 ; PC_CONTROL_INIT, Readies controller for I/O operations
006C 170 ;
006C 171 ; Functional description:
006C 172 ;
006C 173 ;     The operating system calls this routine in 3 places:
006C 174 ;
006C 175 ;             at system startup
006C 176 ;             during driver loading and reloading
006C 177 ;             during recovery from a power failure
006C 178 ;
006C 179 ; Inputs:
006C 180 ;
006C 181 ;     R4     - address of the CSR (controller status register)
006C 182 ;     R5     - address of the IDB (interrupt data block)
006C 183 ;     R6     - address of the DDB (device data block)
006C 184 ;     R8     - address of the CRB (channel request block)
006C 185 ;
006C 186 ; Outputs:
006C 187 ;
006C 188 ;     The routine must preserve all registers except R0-R3.
006C 189 ;
006C 190 ;--
006C 191
006C 192 PC_CONTROL_INIT: ; Initialize controller
05 006C 193     RSB ; Return
006D 194     .SBTTL PC_UNIT_INIT, Unit initialization routine
```

```
006D 196
006D 197 ;++
006D 198 ; PC_UNIT_INIT, Readies unit for I/O operations
006D 199 ;
006D 200 ; Functional description:
006D 201 ;
006D 202 ; The operating system calls this routine after calling the
006D 203 ; controller initialization routine:
006D 204 ;
006D 205 ; at system startup
006D 206 ; during driver loading
006D 207 ; during recovery from a power failure
006D 208 ;
006D 209 ; This routine sets the on-line bit in the UCB, and
006D 210 ; loads the owner field of the IDB with the UCB address.
006D 211 ;
006D 212 ; Inputs:
006D 213 ;
006D 214 ; R4 - address of the CSR (controller status register)
006D 215 ; R5 - address of the UCB (unit control block)
006D 216 ;
006D 217 ; Outputs:
006D 218 ;
006D 219 ; The routine must preserve all registers except R0-R3.
006D 220 ;
006D 221 ;--
006D 222
006D 223 PC_UNIT_INIT:
006D 224 BISW #UCB$M_ONLINE,UCB$W_STS(R5) ; Initialize unit
64 A5 10 A8 006D 224 BISW #UCB$M_ONLINE,UCB$W_STS(R5) ; Set unit online
50 24 A5 D0 0071 225 MOVL UCB$L_CRB(R5),R0 ; Get address of CRB
50 2C A0 D0 0075 226 MOVL CRB$L_INTD+VEC$L_IDB(R0),R0 ; Get address of IDB
04 A0 55 D0 0079 227 MOVL R5,IDB$L_OWNER(R0) ; Set address of dev UCB
007D 228 ; as OWNER--will not
007D 229 ; change
05 007D 230 RSB ; Return
007E 231 .SBTTL CSR_FDT_ROUTINE, Return CSR Address
```

```
007E 233
007E 234 ;++
007E 235 ; CSR_FDT_ROUTINE
007E 236 ;
007E 237 ; Functional description:
007E 238 ;
007E 239 ; This FDT routine returns the address of the device's CSR in
007E 240 ; the high-order longword of the IOSB. The CSR is found in the
007E 241 ; first longword of the IDB. R1 is placed into the high-order
007E 242 ; longword of the IOSB by IOPOST.
007E 243 ;
007E 244 ; Inputs:
007E 245 ;
007E 246 ; R0-R2 - scratch registers
007E 247 ; R3 - address of the IRP (I/O request packet)
007E 248 ; R4 - address of the PCB (process control block)
007E 249 ; R5 - address of the UCB (unit control block)
007E 250 ; R6 - address of the CCB (channel control block)
007E 251 ; R7 - bit number of the I/O function code
007E 252 ; R8 - address of the FDT table entry for this routine
007E 253 ; R9-R11 - scratch registers
007E 254 ; AP - address of the 1st function dependent $QIO parameter
007E 255 ;
007E 256 ; Outputs:
007E 257 ;
007E 258 ; The routine must preserve all registers except R0-R2, and
007E 259 ; R9-R11.
007E 260 ;
007E 261 ;--
007E 262
007E 263 CSR_FDT_ROUTINE:
50 24 A5 DO 007E 264 MOVL UCB$_CRB(R5),R0 ; Get CRB address
51 2C B0 DO 0082 265 MOVL @CRB$_INTD+VEC$_IDB(R0),R1 ; Store CSR in IOSB(2)
50 01 3C 0086 266 MOVZWL #SS$_NORMAL,R0 ; Return normal success
00000000'GF 17 0089 267 JMP G^EXE$FINISHIO ; All done
008F 268 .SBTTL SET_AST_FDT, Reader Attn. AST
```

```
008F 270
008F 271 ;++
008F 272 ; SET_AST_FDT
008F 273 ;
008F 274 ; Functional description:
008F 275 ;
008F 276 ; This FDT routine queues an attention AST for the process
008F 277 ; issuing the SETMODE $QIO. P1 and P2 are checked for
008F 278 ; accessibility.
008F 279 ;
008F 280 ; Inputs:
008F 281 ;
008F 282 ; R0-R2 - scratch registers
008F 283 ; R3 - address of the IRP (I/O request packet)
008F 284 ; R4 - address of the PCB (process control block)
008F 285 ; R5 - address of the UCB (unit control block)
008F 286 ; R6 - address of the CCB (channel control block)
008F 287 ; R7 - bit number of the I/O function code
008F 288 ; R8 - address of the FDT table entry for this routine
008F 289 ; R9-R11 - scratch registers
008F 290 ; AP - address of the 1st function dependent $QIO parameter
008F 291 ;
008F 292 ; Outputs:
008F 293 ;
008F 294 ; The routine must preserve all registers except R0-R2, and
008F 295 ; R9-R11.
008F 296 ;
008F 297 ;--
008F 298
008F 299 SET_AST_FDT:
50 6C D0 008F 300 MOVL P1(AP),R0 ; Get ast address
09 13 0092 301 BEQL NOP1 ; 0 means disable ast
51 04 9A 0094 302 MOVZBL #4,R1 ; It's a longword
00000000'GF 16 0097 303 JSB G^EXE$WRITECHK ; Can user read it
009D 304 ; No return on no access
50 04 AC D0 009D 305 NOP1: MOVL P2(AP),R0 ; Get ast par. if there
06 13 00A1 306 BEQL NOP2 ; No P2 parameter if eq1
00000000'GF 16 00A3 307 JSB G^EXE$WRITECHK ; Can user read it?
00A9 308 ; No return on no access
57 0090 C5 DE 00A9 309 NOP2: MOVAL UCB$L_ASTHEAD(R5),R7 ; Get listhead
00000000'GF 16 00AE 310 JSB G^COM$SETATTNAST ; Set up ast request
00000000'GF 17 00B4 311 JMP G^EXE$FINISHIOC ; Complete SETMODE $QIO
00BA 312 .SBTTL PUNC_AST_FDT, Punch Attn. AST
```

```
00BA 314
00BA 315 ;++
00BA 316 ; PUNC_AST_FDT
00BA 317 ;
00BA 318 ; Functional description:
00BA 319 ;
00BA 320 ; This FDT routine queues an attention AST for the process
00BA 321 ; issuing the SETCHAR $QIO. P1 and P2 are checked for
00BA 322 ; accessibility.
00BA 323 ;
00BA 324 ; Inputs:
00BA 325 ;
00BA 326 ; R0-R2 - scratch registers
00BA 327 ; R3 - address of the IRP (I/O request packet)
00BA 328 ; R4 - address of the PCB (process control block)
00BA 329 ; R5 - address of the UCB (unit control block)
00BA 330 ; R6 - address of the CCB (channel control block)
00BA 331 ; R7 - bit number of the I/O function code
00BA 332 ; R8 - address of the FDT table entry for this routine
00BA 333 ; R9-R11 - scratch registers
00BA 334 ; AP - address of the 1st function dependent $QIO parameter
00BA 335 ;
00BA 336 ; Outputs:
00BA 337 ;
00BA 338 ; The routine must preserve all registers except R0-R2, and
00BA 339 ; R9-R11.
00BA 340 ;
00BA 341 ;--
00BA 342
00BA 343 PUNC_AST_FDT:
50 6C D0 00BA 344 MOVL P1(AP),R0 ; Get ast address
09 13 00BD 345 BEQL NO_P1 ; 0 means disable ast
51 04 9A 00BF 346 MOVZBL #4,R1 ; It's a longword
00000000'GF 16 00C2 347 JSB G^EXE$WRITECHK ; Can user read it
00C8 348 ; No return on no access
50 04 AC D0 00C8 349 NO_P1: MOVL P2(AP),R0 ; Get ast par. if there
06 13 00CC 350 BEQL NO_P2 ; No P2 parameter if eql
00000000'GF 16 00CE 351 JSB G^EXE$WRITECHK ; Can user read it?
00D4 352 ; No return on no access
57 0094 C5 DE 00D4 353 NO_P2: MOVAL UCB$L_PUNCHD(R5),R7 ; Get listhead
00000000'GF 16 00D9 354 JSB G^COM$SETATTNAST ; Set up ast request
00000000'GF 17 00DF 355 JMP G^EXE$FINISHIOC ; Complete SETMODE $QIO
00E5 356 .SBTTL PC_START, Start I/O Routine
```

```
00E5 358 ; PC_START
00E5 359 ;
00E5 360 ;      The driver really has no start I/O entry point.
00E5 361 ;      This is a dummy routine which should never be
00E5 362 ;      called by this driver.
00E5 363 ;
00E5 364 PC_START:
05 00E5 365      RSB
00E6 366      .SBTTL  ASTINT, Interrupt service routines
```

```

00E6 368 ;++
00E6 369 ; ASTINT
00E6 370 ;
00E6 371 ; Functional description:
00E6 372 ;
00E6 373 ;           The driver is for a single-unit controller, and
00E6 374 ;           the unit initialization code has stored the
00E6 375 ;           address of the UCB in the owner field of the IDB.
00E6 376 ;
00E6 377 ;           These routines simply restore the address of the UCB
00E6 378 ;           in R5, place the appropriate AST listhead in R4, and
00E6 379 ;           call a system routine to queue pending AST's. Note
00E6 380 ;           that no check is made to see if the UCB$V_INT bit is
00E6 381 ;           set (in UCB$W_STS), since the routines that set up the
00E6 382 ;           attention AST's do not set the bit.
00E6 383 ;
00E6 384 ; Inputs:
00E6 385 ;
00E6 386 ;           0(SP) - pointer to the address of the IDB (interrupt data
00E6 387 ;           block)
00E6 388 ;           4(SP) - saved R0
00E6 389 ;           8(SP) - saved R1
00E6 390 ;           12(SP) - saved R2
00E6 391 ;           16(SP) - saved R3
00E6 392 ;           20(SP) - saved R4
00E6 393 ;           24(SP) - saved R5
00E6 394 ;           28(SP) - saved PSL (program status longword)
00E6 395 ;           32(SP) - saved PC
00E6 396 ;
00E6 397 ;           The IDB contains the CSR address and the UCB address.
00E6 398 ;
00E6 399 ; Outputs:
00E6 400 ;
00E6 401 ;           The routine must preserve all registers except R0-R5.
00E6 402 ;
00E6 403 ;--
00E6 404
00E6 405 ASTINT:
53 9E DO 00E6 406          MOVL    @(SP)+,R3          ; Reader interrupted
55 04 A3 DO 00E9 407          MOVL    IDB$$_OWNER(R3),R5      ; Get IDB
54 0090 C5 DE 00ED 408        MOVAL   UCB$$_ASTHEAD(R5),R4    ; Get OWNER UCB
00000000 GF 16 00F2 409 DELIV: JSB    GACOM$DELATTNAST      ; Get reader listhead
50 8E 7D 00F8 410          MOVQ    (SP)+,R0                ; Deliver attn. ast's
52 8E 7D 00FB 411          MOVQ    (SP)+,R2                ; Dismiss interrupt
54 8E 7D 00FE 412          MOVQ    (SP)+,R4                ; After restoring
02 0101 413          REI                                ; Registers
0102 414 PUNC_INT:
53 9E DO 0102 415          MOVL    @(SP)+,R3          ; Punch interrupted
55 04 A3 DO 0105 416          MOVL    IDB$$_OWNER(R3),R5      ; Get IDB
54 0094 C5 DE 0109 417        MOVAL   UCB$$_PUNCHD(R5),R4    ; Get OWNER UCB
E2 11 010E 418          BRB      DELIV                    ; Get punch listhead
0110 419          .SBTTL  PC_END, End of driver          ; Deliver attn. ast's

```

PCDRIVER  
V02

- VAX/VMS PAPER TAPE READER/PUNCH DRIVER 12-SEP-1984 18:01:38 VAX-11 Macro V03-01 Page 12  
PC\_END, End of driver 22-AUG-1984 21:15:21 WORK2:[COURSE.DRIVER.RT]PCDRIVER.M(1)

```
0110 421  
0110 422 ;++  
0110 423 ; Label that marks the end of the driver  
0110 424 ;--  
0110 425  
0110 426 PC_END: ; Last location in driver  
0110 427 .END
```



PCDRIVER  
Symbol table

- VAX/VMS PAPER TAPE READER/PUNCH DRIVER 12-SEP-1984 18:01:38 VAX-11 Macro V03-01 Page 13  
22-AUG-1984 21:15:21 WORK2:[COURSE.DRIVER.RT]PCDRIVER.M(1)

\$\$\$	= 00000020	R	02	SET_AST_FDT	0000008F	R	03
\$\$OP	= 00000002			SS\$_NORMAL	= 00000001		
ASTINT	= 000000E6	R	03	UCB\$_DEVCLASS	= 00000040		
AT\$_UBA	= 00000001			UCB\$_DIPL	= 0000005E		
COM\$DELATTNAST	*****	X	03	UCB\$_FIPL	= 0000000B		
COM\$SETATTNAST	*****	X	03	UCB\$_LENGTH	= 00000090		
CRB\$_INTD	= 00000024			UCB\$_PC_UCLEN	00000098		
CRB\$_INTD2	= 00000048			UCB\$_ASTHEAD	00000090		
CSR_FDT_ROUTINE	= 0000007E	R	03	UCB\$_CRB	= 00000024		
DDB\$_L_DDT	= 0000000C			UCB\$_DEVCHAR	= 00000038		
DELIV	= 000000F2	R	03	UCB\$_PUNCHD	00000094		
DEV\$_M_AVL	= 00040000			UCB\$_M_ONLINE	= 00000010		
DEV\$_M_IDV	= 04000000			UCB\$_W_STS	= 00000064		
DEV\$_M_ODV	= 08000000			VEC\$_L_IDB	= 00000008		
DEV\$_M_REC	= 00000001			VEC\$_L_INITIAL	= 0000000C		
DPT\$_C_LENGTH	= 00000038			VEC\$_L_UNITINIT	= 00000018		
DPT\$_C_VERSION	= 00000004						
DPT\$_INITAB	= 00000038	R	02				
DPT\$_REINITAB	= 0000004B	R	02				
DPT\$_TAB	= 00000000	R	02				
DYN\$_C_CRB	= 00000005						
DYN\$_C_DDB	= 00000006						
DYN\$_C_DPT	= 0000001E						
DYN\$_C_UCB	= 00000010						
EXE\$_FINISHIO	*****	X	03				
EXE\$_FINISHIOC	*****	X	03				
EXE\$_WRITECHK	*****	X	03				
FUNCTAB_LEN	= 00000034						
IDB\$_L_OWNER	= 00000004						
IO\$_SENSECHAR	= 0000001B						
IO\$_SETCHAR	= 0000001A						
IO\$_SETMODE	= 00000023						
IO\$_VIRTUAL	= 0000003F						
IOC\$_CANCELIO	*****	X	03				
IOC\$_MNTVER	*****	X	03				
IOC\$_RETURN	*****	X	03				
MASKH	= 00000008						
MASKL	= 00000000						
MY_CLASS	= 000000FC						
NOP1	= 0000009D	R	03				
NOP2	= 000000A9	R	03				
NO_P1	= 000000C8	R	03				
NO_P2	= 000000D4	R	03				
P1	= 00000000						
P2	= 00000004						
P3	= 00000008						
P4	= 0000000C						
P5	= 00000010						
P6	= 00000014						
PC\$_DDT	= 00000000	RG	03				
PC_CONTROL_INIT	= 0000006C	R	03				
PC_END	= 00000110	R	03				
PC_FUNCTABLE	= 00000038	R	03				
PC_START	= 000000E5	R	03				
PC_UNIT_INIT	= 0000006D	R	03				
PUNC_AST_FDT	= 000000BA	R	03				
PUNC_INT	= 00000102	R	03				

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

PSECT name	Allocation	PSECT No.	Attributes
. ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	00000098 ( 152.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	00000065 ( 101.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	00000110 ( 272.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	100	00:00:00.33	00:00:01.16
Command processing	129	00:00:00.46	00:00:01.12
Pass 1	1101	00:00:21.36	00:00:36.45
Symbol table sort	29	00:00:02.85	00:00:05.31
Pass 2	418	00:00:03.19	00:00:03.99
Symbol table output	12	00:00:00.11	00:00:00.14
Psect synopsis output	7	00:00:00.05	00:00:00.19
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1802	00:00:28.37	00:00:48.37

The working set limit was 500 pages.  
96378 bytes (189 pages) of virtual memory were used to buffer the intermediate code.  
There were 100 pages of symbol table space allocated to hold 1819 non-local and 0 local symbols.  
427 source lines were read in Pass 1, producing 16 object records in Pass 2.  
27 pages of virtual memory were used to define 24 macros.

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

Macro library name	Macros defined
SYS\$SYSROOT:[SYSLIB]LIB.MLB;1	14
SYS\$SYSROOT:[SYSLIB]STARLET.MLB;1	8
TOTALS (all libraries)	22

2058 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MAC/LIS PCDRIVER+SYS\$LIBRARY:LIB/LIB

```
100 $!
200 $!
300 $SET VERIFY
400 $MAC/LIS PCDRIVER+SYS$LIBRARY:LIB/LIB
500 $LINK/NOTRACE/MAP PCDRIVER/OPT,PCDRIVER,SYSS$SYSTEM:SYS.STB/SEL
600 $SET NOVERIFY
```

PCDRIVER.COM

100 !  
200 BASE=0

PCDRIVER.OPT

```
100 $!  
200 $!  
300 $MCR SYSGEN  
400 LOAD SYSSYSDISK:[COURSE.DRIVER.RT]PCDRIVER  
500 CONNECT PCA0/ADAPTER=3-  
600 /NUMVEC=2-  
700 /VECTOR=%070-  
800 /CSR=%0777550-  
900 /DRIVER=PCDRIVER  
1000 SHO /DEV=PC  
1100 EXIT
```

```

100 C
200 C                               ATTN.FOR
300 C
400 PROGRAM MAIN
500 C
600 C   This program causes the subroutine SUB to entered
700 C   in kernel mode.  The subroutine will service
800 C   interrupts from the paper tape reader, using the
900 C   set attention ast mechanism.
1000 C
1100 INTEGER*4 SYS$CMKRNL,SYS$EXIT
1200 EXTERNAL SUB,SUBTWO
1300 COMMON ICSR,K,L,ICHAN
1400 COMMON /CHAR/ ALPHA
1410 COMMON /DEBUG/ ICODE,ISTAT
1500 INTEGER*2 ICHAN
1600 CHARACTER*36 ALPHA
1610 ICODE=0
1700 I=SYS$CMKRNL(SUB,)
1800 C
1900 C   Check for error in system service call
2000 C
2100 IF (.NOT. I) CALL SYS$EXIT(%VAL(I))
2110 IF (ICODE .EQ. 0) GOTO 40
2120     WRITE (6,30) ICODE
2130 30     FORMAT (' ERROR ID = ',I4)
2140     CALL SYS$EXIT(%VAL(ISTAT))
2150 40     WRITE (6,50) ALPHA(1:L)
2175 50     FORMAT(' AFTER HIBER, ALPHA = ',A)
2200 C
2300 C   Now enter subroutine SUBTWO in kernel mode to punch some
2400 C   characters on paper tape
2500 C
2600 I=SYS$CMKRNL(SUBTWO,)
2700 IF (.NOT. I) CALL SYS$EXIT(%VAL(I))
2710 WRITE (6,60)
2720 60     FORMAT(' PUNCHING COMPLETE')
2800 END
2900
3000 C   ***** SUB *****
3100
3200 SUBROUTINE SUB
3300 C
3400 C   This routine assigns a channel to the paper tape
3500 C   reader, retrieves the address of the paper tape
3600 C   CSR in IOSB(2) (via a SENSECHAR QIO), and then requests
3700 C   an attention ast (when the paper tape reader interrupts)
3800 C   via a SETMODE QIO (in subroutine SETAST).  The paper
3900 C   tape reader is actually started (and interrupts are
4000 C   enabled) by subroutine INTENA.
4100 C
4200 IMPLICIT INTEGER*4(S)
4300 EXTERNAL READAST,SETAST,INTENA,IO$_SENSECHAR
4400 C
4500 C   Note that integer values and character strings must
4600 C   be placed in separate common blocks (a FORTRAN
4700 C   restriction).
4800 C
4900 COMMON ICSR,K,L,ICHAN

```

```

5000      COMMON /CHAR/ ALPHA
5010      COMMON /DEBUG/ ICODE, ISTAT
5100      INTEGER*4 IOSB(2)
5200      INTEGER*2 ICHAN, IOSBW(4)
5300      CHARACTER*36 ALPHA
5400      EQUIVALENCE (IOSB(1), IOSBW(1))
5500      C
5600      C      J is an error indicator flag
5700      C
5800      C      J=0
5900      C
6000      C      Assign a channel to the paper tape reader
6100      C      and check for system service error.
6200      C
6300      C      I=SYSS$ASSIGN('PCA0:', ICHAN,,)
6400      C      IF (.NOT. I) GOTO 200
6500      C      J=1
6600      C
6700      C      K indicates position in character string for next
6800      C      character read
6900      C      L indicates number of characters to be read into
7000      C      the string (note that the character statements
7100      C      in the subroutines need to be changed to reflect
7200      C      the maximum size that L may take on (set to 36)).
7300      C      Also, byte array (BYTE) in subroutine READAST must
7400      C      be increased to the same size as L.
7500      C
7600      C      K=1
7700      C      L=36
7800      C      I=SYSS$QIOW(,%VAL(ICHAN), IOS$_SENSECHAR, IOSB,,,,,,,,)
7900      C      ICSR=IOSB(2)
8000      C      IF (.NOT. I) GOTO 200
8100      C      J=2
8200      C      IF (IOSBW(1) .NE. 1) GOTO 200
8300      C
8400      C      Enable attention ast (which is actually handled by
8500      C      subroutine READAST).
8600      C      Note that the %VAL function is used to pass the
8700      C      address of the CSR. In the subroutine, the address
8800      C      of the subroutine parameter becomes the address of
8900      C      the CSR, and the subroutine can directly reference
9000      C      the CSR by referencing the subroutine parameter.
9100      C
9200      C      CALL SETAST(0)
9300      C
9400      C      Enable paper tape reader interrupts, and start the
9500      C      paper tape reader (to read the next character).
9600      C
9700      C      CALL INTENA(%VAL(ICSR))
9800      C      J=3
9900      C
10000     C      Wait for all characters to be read (determined by
10100     C      subroutine READAST, who issues the WAKE system service).
10200     C
10300     C      I=SYSS$HIBER()
10400     C      IF (.NOT. I) GOTO 200
10600     C      RETURN
10700     C
10800     C      Enter here if error condition occurs in a system
10900     C      service (J flags where error occurred).

```

```

11000 C
11100 200 ICODE = J
11110 ISTAT = IOSBW(1)
11120 IF (.NOT. I) ISTAT = I
11200 RETURN
11600 END
11700
11800 C ***** READAST *****
11900
12000 SUBROUTINE READAST(MYCSR)
12100 IMPLICIT INTEGER*4(S)
12200 CHARACTER*36 ALPHA
12300 EXTERNAL SETAST,INTENA
12400 INTEGER*2 ICHAN,MYCSR(4)
12500 BYTE DATA(36)
12600 COMMON ICSR,K,L,ICHAN
12700 COMMON /CHAR/ ALPHA
12800 C
12900 C Note that the only valid (FORTRAN) equivalence for
13000 C a character string is to a BYTE variable.
13100 C
13200 EQUivalence (ALPHA(1:),DATA(1))
13300 C
13400 C The data buffer register for the paper tape reader
13500 C follows the paper tape reader CSR. Note that only
13600 C the low order byte contains data. Note also, refer-
13700 C ences to device registers must reference those
13800 C locations as bytes or words only, not as longwords.
13900 C
14000 DATA(K)=MYCSR(2)
14100 C
14200 C Increment index into string, and see if all done
14300 C reading characters by comparing index to number of
14400 C characters desired.
14500 C
14600 K=K+1
14700 IF (K .LE. L) GOTO 75
14800 C
14900 C If all done, awaken main subroutine (SUB).
15000 C
15100 CALL SYS$WAKE(,)
15200 RETURN
15300 C
15400 C Must reenale attention ast for next character since
15500 C attention ast only good for one interrupt. Note that
15600 C a separate subroutine is required since the QIO is
15700 C not allowed to specify the same ast routine address as the
15800 C routine from which the QIO is being issued. Interrupts
15900 C are also reenaled, and the next character is read from
16000 C the paper tape reader (via subroutine INTENA).
16100 C
16200 75 CALL SETAST(0)
16300 CALL INTENA(%VAL(ICSR))
16400 RETURN
16500 END
16600
16700 C ***** SETAST *****
16800
16900 SUBROUTINE SETAST(ITYPE)
17000 IMPLICIT INTEGER*4(S)

```



```

17100 CHARACTER*36 ALPHA
17200 EXTERNAL IO$ SETMODE, READAST, IO$ SETCHAR, PNCHAST
17300 INTEGER*2 ICHAN, IOSBW(4)
17400 INTEGER*4 IOSB(2)
17500 COMMON ICSR, K, L, ICHAN
17600 COMMON /CHAR/ ALPHA
17700 EQUIVALENCE (IOSBW(1), IOSB(1))
17800 C
17900 C Check to see if requesting a reader attention ast (ITYPE=0),
18000 C or a punch attention ast (ITYPE .NE. 0).
18100 C
18200 IF (ITYPE .NE. 0) GOTO 98
18300 C
18400 C Request reader attention ast, specifying as the routine
18500 C to service the ast subroutine READAST, and specifying
18600 C as the ast parameter the CSR of the paper tape reader.
18700 C
18800 I=SYS$QIOW(, %VAL(ICHAN), IO$ SETMODE, IOSB, , , READAST,
18900 1%VAL(ICSR), , , ,)
19000 GOTO 99
19100 C
19200 C Request punch attention ast, specifying as the routine to
19300 C service the ast subroutine PNCHAST, and specifying as the ast
19400 C parameter the CSR of the paper tape reader.
19500 C
19600 98 I=SYS$QIOW(, %VAL(ICHAN), IO$ SETCHAR, IOSB, , , PNCHAST,
19700 1%VAL(ICSR), , , ,)
19800 C
19900 C Check on system service error
20000 C
20100 99 IF (.NOT. I) CALL SYS$EXIT(%VAL(I))
20200 IF (IOSBW(1) .NE. 1) CALL SYS$EXIT(%VAL(IOSBW(1)))
20300 RETURN
20400 END
20500
20600 C ***** INTENA *****
20700
20800 SUBROUTINE INTENA(MYCSR)
20900 INTEGER*2 GO, MYCSR(4)
21000 C
21100 C DECIMAL 65 = HEX 41 = BINARY 01000001 which corresponds
21200 C to the READER and INTERRUPT ENABLE bits in the paper
21300 C tape reader control and status register. These bits
21400 C must be set to enable interrupts and to allow the
21500 C reading of the next character on the paper tape.
21600 C
21700 DATA GO /65/
21800 C
21900 C Set the appropriate bits in the paper tape CSR
22000 C
22100 MYCSR(1) = (MYCSR(1) .OR. GO)
22200 RETURN
22300 END
22400
22500 C ***** PNCHENA *****
22600
22700 SUBROUTINE PNCHENA(MYCSR)
22800 INTEGER*2 GO, MYCSR(4)
22900 C
23000 C DECIMAL 64 = HEX 40 = BINARY 01000000 which corresponds to the

```

```

23100 C      INTERRUPT ENABLE bit in the paper tape punch's control and
23200 C      status register. This bit must be set to allow the punching of
23300 C      characters. The bit needs to be set only once.
23400 C
23500      DATA GO /64/
23600 C
23700 C      Set the appropriate bits in the punch's CSR
23800 C
23900      MYCSR(3) = (MYCSR(3) .OR. GO)
24000      RETURN
24100      END
24200
24300 C      ***** PNCHDAT *****
24400
24500      SUBROUTINE PNCHDAT(MYCSR)
24600      IMPLICIT INTEGER*4(S)
24700      CHARACTER*36 ALPHA
24800      CHARACTER*1 NEXT
24900      EXTERNAL SETAST,IO$ SETCHAR
25000      INTEGER*2 ICHAN,MYCSR(4),IOSBW(4)
25100      INTEGER*4 IOSB(2)
25200      BYTE DATA
25300      COMMON ICSR,K,L,ICHAN
25400      COMMON /CHAR/ ALPHA
25500 C
25600 C      Note that the only valid equivalence for a character string
25700 C      is to a BYTE variable.
25800 C
25900      EQUIVALENCE (NEXT,DATA),(IOSBW(1),IOSB(1))
26000 C
26100 C      Reenable attention ast for next character.
26200 C
26300      CALL SETAST(1)
26400 C
26500 C      The paper tape punch is started by writing a character in
26600 C      the punch's data buffer (in this case, MYCSR(4)). Only the
26700 C      low order byte contains data.
26800 C
26900      NEXT = ALPHA(K:K)
27000      MYCSR(4) = DATA
27100 C
27200 C      Increment index into string, and see if all done punching
27300 C      characters by comparing index (K) to number desired (L).
27400 C
27500      K=K+1
27600      IF (K .LE. L) GOTO 650
27700 C
27800 C      If all done, awaken main subroutine (SUBTWO)
27900 C      after disabling the last attention ast requested
28000 C
28100      I=SYS$QIOW(,%VAL(ICHAN),IO$ SETCHAR,IOSB,,,,,,)
28200      IF (.NOT. I) CALL SYS$EXIT(%VAL(I))
28300      IF (IOSBW(1) .NE. 1) CALL SYS$EXIT(%VAL(IOSBW(1)))
28400      CALL SYS$WAKE(,)
28500 650      RETURN
28600      END
28700
28800 C      ***** SUBTWO *****
28900
29000      SUBROUTINE SUBTWO

```

```

29100 C
29200 C This subroutine will punch (L) characters from the character
29300 C string alpha. K serves as an index into the string. If
29400 C more than (36) characters are desired, the character statements
29500 C have to be edited to the appropriate maximum length in this
29600 C subroutine, and all subroutines called by this subroutine.
29700 C
29800 IMPLICIT INTEGER*4(S)
29900 EXTERNAL PNCHAST,SETAST,PNCHENA,PNCHDAT
30000 COMMON ICSR,K,L,ICHAN
30100 COMMON /CHAR/ ALPHA
30200 INTEGER*2 ICHAN
30300 CHARACTER*36 ALPHA
30400 C
30500 C It is assumed that a channel has already been assigned to the
30600 C punch (as in subroutine SUB), and that the CSR address has been
30700 C placed into ICSR (via a SENSECHAR QIO -- see subroutine SUB).
30800 C
30900 ALPHA='ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
31000 K=1
31100 L=36
31200 C
31300 C Enable punch interrupts
31400 C
31500 CALL PNCHENA(%VAL(ICSR))
31600 C
31700 C Punch first character (will set up attn. ast)
31800 C
31900 CALL PNCHDAT(%VAL(ICSR))
32000 C
32100 C Wait for all characters to be punched (determined by
32200 C subroutine PNCHAST, who issues the WAKE system service).
32300 C
32400 I=SYSSHIBER()
32500 IF (.NOT. I) CALL SYS$EXIT(%VAL(I))
32700 RETURN
32900 END
33000
33100 C ***** PNCHAST *****
33200
33300 SUBROUTINE PNCHAST(MYCSR)
33400 EXTERNAL PNCHDAT
33500 COMMON ICSR,K,L,ICHAN
33600 COMMON /CHAR/ ALPHA
33700 INTEGER*2 ICHAN,MYCSR(4)
33800 CHARACTER*36 ALPHA
33900 C
34000 C This subroutine passes control to subroutine PNCHDAT. It is
34100 C included so PNCHDAT doesn't specify itself as the ast routine.
34200 C
34300 CALL PNCHDAT(%VAL(ICSR))
34400 RETURN
34500 END

```



**PRINTER.LIS  
PRINTER.COM  
CINLOAD.COM**

PRINTER  
Table of contents

(1)	2697	Local Data
(1)	9402	System Mapped Data
(1)	10702	System Mapped Routines
(1)	17102	Main Program
(1)	27702	AST Routine For Each Printer Interrupt

```

0000 100      .TITLE  PRINTER
0000 110      .IDENT  /V01/
0000 120 ;
0000 130 ;      CONNECT TO INTERRUPT EXAMPLE
0000 140 ;
0000 200 ;
0000 300 ;
0000 400 ;
0000 500 ;      This program controls the line printer using the connect
0000 600 ;      to interrupt system service scheme. The printer's CSR is
0000 700 ;      ^0777514, it has two registers (CSR first, then data register).
0000 800 ;      The printer is activated by placing a value to be printed in the
0000 900 ;      data register. The printer generates an interrupt after the
0000 1000 ;     character is printed if the interrupt enable bit in the CSR
0000 1100 ;     is set. The program will prompt the user for data,
0000 1200 ;     and then print the data (telling the user each character that
0000 1300 ;     was printed).
0000 1310 ;     AUTHOR:      Viktors J. Muiznieks      25-March-1980
0000 1500 ;
0000 1600 ;     System macros for defining offsets and symbols
0000 1700 ;
0000 1800 ;
0000 1900      $CINDEF                ; connect to interrupt
0000 2000      $CRBDEF                ; channel request block
0000 2100      $IDBDEF                ; interrupt data block
0000 2200      $IODEF                 ; I/O function codes
0000 2300      $IO780DEF              ; 11-780 adapter codes
0000 2400      $UCBDEF                ; unit control block
0000 2500      $VECDEF                ; CRB vector fields
0000 2600 ;
0000 2605 ;
0000 2610 ;      Local Symbol Definitions
0000 2615 ;
0000 2620 ;
0003FF4C 0000 2625 CSR = ^0777514      ; printer CSR register
0000014C 0000 2630 CSR_PAGE_OFFSET = <CSR&^X1FF> ; page offset of CSR
001009FF 0000 2635 PFN = <IO780$AL_UBOSP+CSR>@-9 ; PFN for printer reg.'s
0000      0000 2640                    ; @-9 divides by 512
00000001 0000 2645 PAGE_COUNT = 1      ; reg. fit on 1 page
00000000 0000 2650 STATUS = 0          ; status reg. offset
00000002 0000 2655 DATA = 2          ; data reg. offset
00000040 0000 2660 INT_ENABLE = ^X40  ; int. enable bit
00000080 0000 2665 READY = ^X80       ; printer ready bit
00008000 0000 2670 ERR_BIT = ^X8000    ; printer error bit
000001FB 0000 2675 MAX_TT_BYTES = 507  ; entered by user
00000020 0000 2680 LF_BUFFER_CR = 32   ; TT carriage control
0000000D 0000 2685 CR = 13             ; carriage return
0000000A 0000 2690 LF = 10            ; line feed
00000001 0000 2692 EXPECTED = ^1      ; int. expected bit
0000      0000 2695
0000      0000 2697      .SBTTL  Local Data

```

```

0000 2700 ;
0000 2800 ; Parameters for $ASSIGN and $QIO system services
0000 2900 ;
0000 3100 DEVICE_NAME: ; printer descriptor
3A 30 41 50 4C 5F 00000008'010E0000' 0000 3200 .ASCID /_LPA0:/
000E 3300 TT_NAME: ; terminal descriptor
54 54 00000016'010E0000' 000E 3400 .ASCID /TT/
0018 3500 TT_CHAN: ; terminal channel #
0000 0018 3600 .WORD
001A 3700 DEVICE_CHAN: ; printer channel #
0000 001A 3800 .WORD
00000000 00000000 001C 3900 IOSB: .QUAD ; IOSB for $QIO calls
0024 4100 ;
0024 4200 ; Parameters for $CRMPSC system service to map the Unibus I/O page
0024 4300 ;
00000200' 0024 4500 MAP_ADR: ; for $CRMPSC
00000200' 0024 4600 .ADDRESS ^X200 ; start address (P0)
0028 4700 .ADDRESS ^X200 ; end address (P0)
002C 4800 ;
002C 4900 ;
002C 5000 ; Messages sent to the terminal user
002C 5100 ;
002C 5200 ;
002C 5300 PROMPT_STRING: ; for terminal user
70 20 65 62 20 6F 74 20 61 74 61 44 002C 5400 .ASCII /Data to be printed (<CR> for none): /
3E 52 43 3C 28 20 64 65 74 6E 69 72 0038
20 3A 29 65 6E 6F 6E 20 72 6F 66 20 0044
00000024 0050 5500 PROMPT_LENGTH = . - PROMPT_STRING
0050 5600 AST_STRING: ; sent in AST routine
69 72 70 20 64 65 68 73 69 6E 69 46 0050 5700 .ASCII /Finished printing character /
63 61 72 61 68 63 20 67 6E 69 74 6E 005C
20 72 65 74 0068
006C 5800 CHARACTER:
00 006C 5900 .BYTE
0000001D 006D 6000 AST_LENGTH = . - AST_STRING
006D 6100 ERROR_STRING: ; if print error occurs
72 72 75 63 63 6F 20 72 6F 72 72 45 006D 6200 .ASCII /Error occurred in printing data - program aborting./
69 74 6E 69 72 70 20 6E 69 20 64 65 0079
72 70 20 2D 20 61 74 61 64 20 67 6E 0085
69 74 72 6F 62 61 20 6D 61 72 67 6F 0091
2E 67 6E 009D
00000033 00A0 6300 ERR_LENGTH = . - ERROR_STRING
00A0 6400
00A0 8200 ;
00A0 8300 ; Parameters used by connect to interrupt $QIO system service
00A0 8400 ;
00A0 8500
00A0 8600 ROUTINE_OFFSET_LIST:
00000000 00A0 8700 .LONG 0 ; powerfail recovery
00000000 00A4 8800 .LONG 0 ; start I/O
00000200' 00A8 8900 .LONG LP_INTERRUPT - SHARED_BUFFER_START ; interrupt service
0000022F' 00AC 9000 .LONG LP_CANCEL - SHARED_BUFFER_START ; cancel I/O
00B0 9100 SHARED_BUFFER_DESC:
0000023D' 00B0 9200 .LONG SHARED_BUFFER_END - SHARED_BUFFER_START ; buffer length
00000000' 00B4 9300 .LONG SHARED_BUFFER_START ; buffer starting addr.
00B8 9402 .SBTTL System Mapped Data

```



```
00B8 9500 ;
00B8 9600 ;
00B8 9700 ;
00B8 9800 ;
00B8 9900 ;
00B8 10000 ;
00B8 10100
00000000 10200 .PSECT SHARED BUF,PAGE,WRT
0000 10300
0000 10400 SHARED_BUFFER_START:
00 0000 10500 .BYTE 0 ; used as index
000001FC 0001 10600 TTDATA: .BLKB MAX_TT_BYTES ; max. 507 bytes
000001FE 01FC 10640 .BLKB 2 ; for <CR> <LF>
00 01FE 10680 CURRENT: .BYTE 0 ; data printed
00 01FF 10690 FLAGS: .BYTE 0 ; expected int. flag
0200 10700
0200 10702 .SBTTL System Mapped Routines
```

```

0200 10800 ; Note that since offsets must be specified in the routine
0200 10900 ; offset list above, the following service routines must be
0200 11000 ; placed immediately following the shared buffer data area.
0200 11100 ;
0200 11400 ; The LP_INTERRUPT routine is called (in kernel mode, in system
0200 11500 ; context) after the printer generates an interrupt. The inputs are
0200 11600 ; R2 - address of counted argument list
0200 11700 ; R4 - address of IDB
0200 11800 ; R5 - address of UCB
0200 11900 ;
0200 12000 ; The counted argument list contains the following items:
0200 12100 ; 0(R2) - count of arguments (4)
0200 12200 ; 4(R2) - system mapped address of shared buffer
0200 12300 ; 8(R2) - address of AST parameter
0200 12400 ; 12(R2) - IDB address
0200 12500 ; 16(R2) - UCB address
0200 12600 ;
0200 12700 ; The routine will place the character printed into the CURRENT
0200 12800 ; location, and request that an AST be delivered (by placing
0200 12900 ; a 1 in R0 before issuing an RSB). If any error occurred in
0200 13000 ; the print operation, a 0 is placed into CURRENT. If the
0200 13010 ; interrupt is unexpected, no AST is requested (R0 cleared).
0200 13100 ;
0200 13200 ; The routine must preserve all registers except R0-R4.
0200 13300 ;
0200 13800 LP_INTERRUPT: ; interrupt service
27 F9 AF 50 D4 0200 13810 CLRRL R0 ; assume unexpected int.
01 E5 0202 13820 BBCC #EXPECTED,FLAGS,NO_AST ; dismiss if unexpected
F4 AF 94 0207 13900 CLRBL CURRENT ; assume printer error
54 64 D0 020A 14000 MOVL IDB$CSR(R4),R4 ; get CSR address
64 0080 8F B3 020D 14100 BITW #READY,STATUS(R4) ; printer ready?
15 13 0212 14200 BEQL Q_AST ; error if not
64 8000 8F B3 0214 14300 BITW #ERR_BIT,STATUS(R4) ; printer error?
OE 12 0219 14400 BNEQ Q_AST ; yes if NEQ
53 04 A2 D0 021B 14500 MOVL 4(R2),R3 ; get buffer address
54 63 9A 021F 14600 MOVZBL (R3),R4 ; get index to char.
53 54 C0 0222 14700 ADDL2 R4,R3 ; point to char.
D5 AF 63 90 0225 14800 MOVBL (R3),CURRENT ; specify current data
50 0000 8F 3C 0229 14900 Q_AST: MOVZWL #SS$_NORMAL,R0 ; request AST delivery
05 022E 15000 NO_AST: RSB
022F 15200 ;
022F 15300 ; The LP_CANCEL routine is called (in kernel mode, in system
022F 15400 ; context) when the process is being disconnected from the
022F 15500 ; interrupt. This routine will disable printer interrupts. The
022F 15600 ; inputs to the routine are as follows:
022F 15800 ; R2 - negated value of channel index number
022F 15900 ; R3 - IRP address
022F 16000 ; R4 - PCB address for process cancelling I/O
022F 16100 ; R5 - UCB address
022F 16400 LP_CANCEL: ; cancel I/O routine
50 24 A5 D0 022F 16500 MOVL UCB$CRB(R5),R0 ; get CRB
50 2C B0 D0 0233 16600 MOVL @CRB$_INTD+VEC$_IDB(R0),R0 ; get CSR address
60 0040 8F AA 0237 16700 BICW #INT_ENABLE,STATUS(R0) ; disable interrupts
05 023C 16800 RSB
023D 17000 SHARED_BUFFER_END:
023D 17102 .SBTTL Main Program

```

Main Program

```

023D 17200 ;
023D 17300 ;
023D 17400 ;
023D 17500 ;
023D 17600 ;
023D 17700 ;
00000000 17800 .PSECT CODE,EXE,NOWRT,PAGE
0000 17900
0000 0000 18000 START: .WORD 0 ; null entry mask
0002 18100 ;
0002 18200 ; Assign channels to terminal and printer
0002 18300 ;
0002 18400
03 50 E8 0017 18500 $ASSIGN_S DEVNAM=TT_NAME,CHAN=TT_CHAN
014C 31 001A 18600 BLBS R0,10$
001D 18700 10$: BRW ERROR
03 50 E8 0032 18800 $ASSIGN_S DEVNAM=DEVICE_NAME,CHAN=DEVICE_CHAN
0131 31 0035 18900 BLBS R0,20$
0038 19000 ; BRW ERROR
0038 19100 ;
0038 19200 ; Map Unibus I/O page into process P0 space so device registers
0038 19300 ; are accessible. Flags indicate that section is to be mapped
0038 19400 ; at end of P0 region (which will be expanded to accomodate the
0038 19500 ; new addresses since the SEC$M_EXPREG bit is set); the P0 region
0038 19600 ; is identified because the INADR argument contains addresses in
0038 19700 ; the P0 range. The SEC$M_PFNMAP bit indicates that the mapping
0038 19800 ; should be done by a pfn in the VBN argument (rather than by the
0038 19900 ; default virtual block number mapping scheme). The region
0038 20000 ; mapped can be written to by the program since the SEC$M_WRT
0038 20100 ; bit is also set (i.e., device registers can be written to by
0038 20200 ; the program).
0038 20300 ; Note that only one page (PAGCNT parameter) is mapped. If the
0038 20400 ; device registers 'straddle' a page boundary, then more than
0038 20500 ; one page should be mapped.
0038 20600 20$: $CRMPSC_S INADR = MAP_ADR ,- ; map into P0 region
0038 20700 RETADR= MAP_ADR ,- ; return actual addr.
0038 20800 FLAGS = #SEC$M_WRT!SEC$M_PFNMAP!SEC$M_EXPREG,-
0038 20900 PAGCNT= #PAGE_COUNT ,- ; only need 1 page
0038 21000 VBN = #PFN ; pfn for mapped page
03 50 E8 0066 21100 BLBS R0,30$
00FD 31 0069 21200 BRW ERROR
006C 21300 ;
006C 21400 ; Issue connect to interrupt $QIO
006C 21500 ;
006C 21600 30$: $QIO_S CHAN = DEVICE_CHAN ,- ; specify printer chan
006C 21700 FUNC = #IO$_CONINTWRITE ,- ; allow write buffer
006C 21900 P1 = SHARED_BUFFER_DESC ,- ; shared buffer desc.
006C 22000 P2 = #ROUTINE_OFFSET_LIST ,- ; routines provided
006C 22100 P3=#<<3@16>!CIN$M_EFN!CIN$M_ISR!CIN$M_CANCEL!CIN$M_REPEAT>,-
006C 22200 P4 = #LP_AST ,- ; AST after interrupt
006C 22300 P5 = #0 ,- ; no AST parameter used
006C 22400 P6 = #1 ; AST count of 1
03 50 E8 009F 22500 BLBS R0,40$
00C4 31 00A2 22600 BRW ERROR
00A5 22700 ;
00A5 22800 ; Ask user for data to be printed
00A5 22900 ;

```

```

00A5 23000 40$: $QIOW_S CHAN=TT_CHAN,FUNC=#IO$ READPROMPT,IOSB=IOSB,-
00A5 23100 P1=TTDATA,P2=#MAX_TT_BYTES,P5=#PROMPT_STRING,-
00A5 23200 P6=#PROMPT_LENGTH
03 50 E8 00D6 23300 BLBS R0,50$
008D 31 00D9 23400 BRW ERROR
00DC 23500 ;
00DC 23600 ; Print data user entered, or exit if no data to print
00DC 23700 ;
0000001E'EF B5 00DC 23800 50$: TSTW IOSB+2 ; check byte count
5A 13 00E2 23900 BEQL DONE ; exit if zero
55 0000001E'EF 3C 00E4 24000 MOVZWL IOSB+2,R5 ; save byte count
52 00000001'EF DE 00EB 24100 MOVAL TTDATA,R2 ; point to data
52 55 C0 00F2 24110 ADDL2 R5,R2 ; point to end of data
62 0A0D 8F B0 00F5 24120 MOVW #<CR>+<LF@8>,(R2) ; insert <CR> <LF>
52 55 C2 00FA 24130 SUBL2 R5,R2 ; restore pointer
55 02 C0 00FD 24140 ADDL2 #2,R5 ; account for <CR> <LF>
54 0000014C 8F 00000024'EF C1 0100 24200 ADDL3 MAP_ADR,#CSR_PAGE_OFFSET,R4 ; point to CSR
53 02 A4 3E 010C 24300 MOVAV DATA(R4),R3 ; point to data reg.
00000000'EF 94 0110 24400 CLRB SHARED_BUFFER_START ; initialize index
64 0040 8F AB 0116 24500 BISW #INT_ENABLE,STATUS(R4) ; set int. enable bit
00000000'EF 96 011B 24600 60$: INCB SHARED_BUFFER_START ; adjust byte index
00 000001FF'EF 01 E3 0121 24610 BBSC #EXPECTED,FLAGS,70$ ; indicate int. expected
63 82 90 0129 24700 70$: MOVB (R2)+,(R3) ; print a character
E5 55 F5 0133 24900 $HIBER_S ; wait for AST to wake
64 0040 8F AA 0136 25000 SOBGTR R5,60$ ; loop if more to print
FF67 31 013B 25100 BICW #INT_ENABLE,STATUS(R4) ; disable printer int.
013E 25200 BRW 40$ ; get more user data
013E 25300 ;
013E 25400 ; User doesn't want any more data printed
013E 25500 ;
013E 25600 DONE: $DASSGN_S CHAN=TT_CHAN ; clean up channels
1A 50 E9 014C 25700 BLBC R0,ERROR
014F 25800 $DASSGN_S CHAN=DEVICE_CHAN
09 50 E9 015D 25900 BLBC R0,ERROR
0160 26000 $EXIT_S
0169 26100
0169 26200 ;
0169 26300 ; Error occurred -- report it to user
0169 26400 ;
0169 26500 ERROR: $EXIT_S CODE=R0
0172 26600
0172 27702 .SBTTL AST Routine For Each Printer Interrupt

```

```

0172 27800 ;
0172 27900 ;      This AST routine is entered whenever the device generates
0172 28000 ;      an interrupt, and the process-supplied AST routine places
0172 28100 ;      a success code in R0 before issuing the RSB.
0172 28300 ;      The routine displays for the user the character
0172 28400 ;      printed, and awakens the main procedure to print
0172 28500 ;      the next character (unless there was an error in the
0172 28600 ;      printing of the character, indicated by CURRENT
0172 28700 ;      containing a 0, in which case an error message is displayed
0172 28800 ;      to the user, and the image exits).
0172 28900 ;
0000 0172 29000 LP_AST: .WORD 0 ; null entry mask
000001FE'EF 0D 91 0174 29140 CMPB #CR,CURRENT ; was character a <CR>?
41 13 017B 29180 BEQL 10$ ; if so, no message
000001FE'EF 0A 91 017D 29220 CMPB #LF,CURRENT ; was character a <LF>?
38 13 0184 29260 BEQL 10$ ; if so, no message
000001FE'EF 95 0186 29300 TSTB CURRENT ; 0 ==> printer error
3C 13 018C 29400 BEQL ERR ; report error
0000006C'EF 000001FE'EF 90 018E 29410 MOVB CURRENT,CHARACTER ; save char. printed
0199 29500 $QIOW_S CHAN=TT_CHAN,FUNC=#IO$_WRITEVBLK,- ; tell user that
0199 29600 P1=AST_STRING,P2=#AST_LENGTH,- ; character has been
0199 29700 P4=#LF_BUFFER_CR ; printed
01BE 29800 10$: $WAKE_S ; print next character
04 01C9 29900 RET
01CA 30000 ERR: $QIOW_S CHAN=TT_CHAN,FUNC=#IO$_WRITEVBLK,- ; tell user that
01CA 30100 P1=ERROR_STRING,P2=#ERR_LENGTH,- ; error occurred in
01CA 30200 P4=#LF_BUFFER_CR ; printing
01EF 30300 $EXIT_S CODE=#SS$_DRVERR ; and stop
01FC 30400
01FC 30500 .END START
    
```

PRINTER  
Symbol table

```

$ST1 = 00000001
AST_LENGTH = 0000001D
AST_STRING 00000050 R 01
CHARACTER 0000006C R 01
CIN$M_CANCEL = 00000080
CIN$M_EFN = 00000001
CIN$M_ISR = 00000040
CIN$M_REPEAT = 00000004
CR = 0000000D
CRB$L_INTD = 00000024
CSR = 0003FF4C
CSR_PAGE_OFFSET = 0000014C
CURRENT 000001FE R 03
DATA = 00000002
DEVICE_CHAN 0000001A R 01
DEVICE_NAME 00000000 R 01
DONE 0000013E R 04
ERR 000001CA R 04
ERROR 00000169 R 04
ERROR_STRING 0000006D R 01
ERR_BIT = 00008000
ERR_LENGTH = 00000033
EXPECTED = 00000001
FLAGS 000001FF R 03
IDB$L_CSR = 00000000
INT_ENABLE = 00000040
IO$_CONINTWRITE = 0000003D
IO$_READPROMPT = 00000037
IO$_WRITEVBLK = 00000030
IO780$AL_UBQSP = 20100000
IOSB 0000001C R 01
LF = 0000000A
LF_BUFFER_CR = 00000020
LP_AST 00000172 R 04
LP_CANCEL 0000022F R 03
LP_INTERRUPT 00000200 R 03
MAP_ADR 00000024 R 01
MAX_TT_BYTES = 000001FB
NO_AST 0000022E R 03
PAGE_COUNT = 00000001
PFN = 001009FF
PROMPT_LENGTH = 00000024
PROMPT_STRING 0000002C R 01
Q_AST 00000229 R 03
READY = 00000080
ROUTINE_OFFSET_LIST 000000A0 R 01
SEC$M_EXPREG ***** X 04
SEC$M_PFNMAP ***** X 04
SEC$M_WRT ***** X 04
SHARED_BUFFER_DESC 000000B0 R 01
SHARED_BUFFER_END 0000023D R 03
SHARED_BUFFER_START 00000000 R 03
SS$_DRVERR ***** X 04
SS$_NORMAL ***** X 03
START 00000000 R 04
STATUS = 00000000
SYSS$ASSIGN ***** GX 04

```

```

SYSS$CRMPSC ***** GX 04
SYSS$DASSGN ***** GX 04
SYSS$EXIT ***** GX 04
SYSS$HIBER ***** GX 04
SYSS$QIO ***** GX 04
SYSS$QIOW ***** GX 04
SYSS$WAKE ***** GX 04
TTDATA 00000001 R 03
TT_CHAN 00000018 R 01
TT_NAME 0000000E R 01
UCB$L_CRB = 00000024
VEC$L_IDB = 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 .	000000B8 ( 184.)	01 ( 1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$ABS\$	00000000 ( 0.)	02 ( 2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SHARED BUF	0000023D ( 573.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC PAGE
CODE	000001FC ( 508.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	104	00:00:00.31	00:00:01.24
Command processing	128	00:00:00.50	00:00:02.19
Pass 1	828	00:00:11.02	00:00:14.03
Symbol table sort	14	00:00:01.32	00:00:01.33
Pass 2	207	00:00:01.73	00:00:01.92
Symbol table output	10	00:00:00.08	00:00:00.15
Psect synopsis output	5	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1299	00:00:14.99	00:00:20.89

The working set limit was 500 pages.  
48128 bytes (94 pages) of virtual memory were used to buffer the intermediate code.  
There were 50 pages of symbol table space allocated to hold 852 non-local and 8 local symbols.  
307 source lines were read in Pass 1, producing 17 object records in Pass 2.  
26 pages of virtual memory were used to define 25 macros.

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

Macro library name	Macros defined
SYS\$SYSROOT:[SYSLIB]LIB.MLB;1	6
SYS\$SYSROOT:[SYSLIB]STARLET.MLB;1	16
TOTALS (all libraries)	22

996 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/LIS PRINTER+SYS\$LIBRARY:LIB/LIB

PRINTER.COM

100 \$!  
200 \$!  
300 \$MACRO/LIS PRINTER+SYS\$LIBRARY:LIB/LIB  
400 \$LINK PRINTER,SYS\$SYSTEM:SYS.STB/SEL



```
100 $!  
200 $!  
300 $MCR SYSGEN  
400 CONNECT LPA0/ADAPTER=3-  
500 /VECTOR=%O200-  
600 /CSR=%O777514-  
700 /DRIVER=CONINTERR  
800 SHO /DEV=CON  
900 USE CURRENT  
1000 SHOW REALTIME_SPTS  
1100 EXIT
```

