

```

EEEEEEEEEEEEEEEEEE DDDDDDDDDDDDD TTTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEEEE DDDDDDDDDDDDD TTTTTTTTTTTTTTTT
EEEEEEEEEEEEEEEEEE DDDDDDDDDDDDD TTTTTTTTTTTTTTTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEEEEEEEEEEEEE DDD DDD TTT
EEEEEEEEEEEEEE DDD DDD TTT
EEEEEEEEEEEEEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEE DDD DDD TTT
EEEEEEEEEEEEEEEEEE DDDDDDDDDDDDD TTT
EEEEEEEEEEEEEEEEEE DDDDDDDDDDDDD TTT
EEEEEEEEEEEEEEEEEE DDDDDDDDDDDDD TTT

```

[illegible]

```
IIIIII  000000  MM  MM  000000  DDDDDDDDD
IIIIII  000000  MM  MM  000000  DDDDDDDDD
II      00      00  MMMM  MMMM  00      00  DD      DD
II      00      00  MMMM  MMMM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
II      00      00  MM  MM  MM  00      00  DD      DD
IIIIII  000000  MM  MM  000000  DDDDDDDDD
IIIIII  000000  MM  MM  000000  DDDDDDDDD
                                     ....
                                     ....
                                     ....
```

```
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLLLL IIIIII  SSSSSSSS
LLLLLLLLLLLL IIIIII  SSSSSSSS
```

```
0001 0 %TITLE 'EDT$IOMOD - I/O for VAX/VMS'
0002 0 MODULE EDT$IOMOD (
0003 0 IDENT = 'V04-000'
0004 0 ) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1 FACILITY: EDT -- The DEC Standard Editor
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 This module contains all the I/O interface code for VAX/VMS
0037 1 systems.
0038 1
0039 1 ENVIRONMENT: VAX/VMS only
0040 1
0041 1 AUTHOR: Bob Kushlis, CREATION DATE: 28-JUL-1979
0042 1
0043 1 MODIFIED BY:
0044 1
0045 1 Dan Szymanski, 04-DEC-80, 02
0046 1 FABSM_MXV added in EDT$$OPN_OFIDEF so that the output file if created with
0047 1 a maximized version number.
0048 1
0049 1 Dan Szymanski, 16-DEC-80, 03
0050 1 Trailing : removed during recursive translation of SYSS$INPUT in EDT$$TI_OPN .
0051 1
0052 1 John Sauter, 19-Dec-1980, 04
0053 1 Added calls to trace package under conditional compilation.
0054 1
0055 1 2-005 - Regularized the headers. JBS 17-Feb-1981
0056 1 2-006 - Make PIC, get the event flag numbers dynamicly, carry protection to
0057 1 output files from the first input file, and manipulate storage using
```

```
58 0058 1 LIB$GET_VM and LIB$FREE_VM. JBS 17-Feb-1981
59 0059 1 2-007 - Add a new entry point to close the work file, and removed some globals
60 0060 1 made unnecessary by it. JBS 18-Feb-1981
61 0061 1 2-008 - Make EDT$$FATAL_IOERR global, to avoid calls to SY_EXIT. JBS 19-Feb-1981
62 0062 1 2-009 - Add EDT$$G_IOFI_NFND, remove IO_FNF and make IO_ST5 not
63 0063 1 global. JBS 19-Feb-1981
64 0064 1 2-010 - Fix module name. JBS 06-Mar-1981
65 0065 1 2-011 - Fix remaining IO entry points. JBS 30-Mar-1981
66 0066 1 2-012 - Only get event flags if we don't already have them. JBS 02-Apr-1981
67 0067 1 2-013 - Use the ASSERT macro. JBS 01-Jun-1981
68 0068 1 2-014 - Put the work file on SYS$SCRATCH, and use EDTWORK as a logical name
69 0069 1 for it. JBS 01-Jun-1981
70 0070 1 2-015 - Fix a typo that caused buckets to never be marked dirty. JBS 04-Jun-1981
71 0071 1 2-016 - In response to a request from the VMS group, make all terminals $QIOs be
72 0072 1 $QIOs, for compatibility with the VMS 3.0 terminal driver, which will
73 0073 1 not always use buffered I/O. Maybe later we can find time to smarten up
74 0074 1 the code. JBS 06-Jun-1981
75 0075 1 2-017 - Arrange to allocate the event flags before entering change mode, if necessary.
76 0076 1 JBS 06-Jun-1981
77 0077 1 2-018 - Fix up the protection. The work file needs only system delete, owner read,
78 0078 1 write and delete. Other output files have the same protection as the primary
79 0079 1 input file. JBS 08-Jul-1981
80 0080 1 2-019 - Use the new message codes. JBS 05-Aug-1981
81 0081 1 2-020 - Add a missing call to $$TRACE, to pick up a prompt. JBS 06-Aug-1981
82 0082 1 2-021 - Correct a call to $$TRACE_READ, which was clobbering TT_CHAN. JBS 07-Aug-1981
83 0083 1 2-022 - Use EDT$$ALO_HEAP and EDT$$DEA_HEAP instead of
84 0084 1 LIB$GET_VM and LIB$FREE_VM. JBS 07-Aug-1981
85 0085 1 2-023 - Correct a bug in reading trace files. JBS 19-Aug-1981
86 0086 1 2-024 - When reading a trace file, don't interact with the terminal. This will allow
87 0087 1 EDT running in a batch job to read trace files. JBS 20-Aug-1981
88 0088 1 2-025 - If the terminal type is unknown, but it is an ANSI-mode scope terminal, assume
89 0089 1 it is a VT100. This is needed because the terminal type distinguishes the
90 0090 1 VT100 from the VT101, VT102, VT125, etc which are compatible with it.
91 0091 1 JBS 16-Nov-1981
92 0092 1 2-026 - Put a test in EDT$$TI_OPN so it is a noop if called twice. This is needed because
93 0093 1 in tracing EDT it is called from startup as well as init. JBS 17-Dec-1981
94 0094 1 2-027 - Set default terminal width to system's terminal width. SMB 23-Dec-1981
95 0095 1 2-028 - make file i/o routines pass back status to caller. STS 06-Jan-1982
96 0096 1 2-029 - Run through PRETTY. JBS 12-Jan-1982
97 0097 1 2-030 - Pass error information from opn_ifi back to fileio. STS 13-Jan-1982
98 0098 1 2-031 - Pass error info from opn_ofi back to fileio. STS 13-Jan-1982
99 0099 1 2-032 - Pass error info from read and write routines back. STS 15-Jan-1982
100 0100 1 2-033 - Pass error info from delete journal file. STS 18-Jan-1982
101 0101 1 2-034 - Make sure NAM$B_RSL is non-zero. STS 19-Jan-1982
102 0102 1 2-035 - Remove an extra dot in RHB handling. JBS 21-Jan-1982
103 0103 1 2-036 - Add check for VFC file when opening and getting. STS 22-Jan-1982
104 0104 1 2-037 - Don't call a PRN file VFC. JBS 30-Jan-1982
105 0105 1 2-038 - Pass workfile bucket back in descriptor. STS 02-Feb-1982
106 0106 1 2-039 - Delete edt$$get_fnam. STS 12-Feb-1982
107 0107 1 2-040 - Add descriptor to parameters passed to edt$$wf_wr and edt$$wf_rd.
108 0108 1 STS 15-Feb-1982
109 0109 1 2-041 - Track the new terminal driver's data structures. JBS 21-Feb-1982
110 0110 1 2-042 - Initialize record attribute for primary file to CR. STS 05-Mar-1982
111 0111 1 2-043 - Don't signal on exceeded disk if ctx bit not set. STS 18-Mar-1982
112 0112 1 2-044 - Add a parameter to control use of related file name. JBS 24-Mar-1982
113 0113 1 2-045 - Only remember input file name and attributes on the first input open,
114 0114 1 even if it fails. JBS 26-Mar-1982
```

115	0115	1	2-046	- Add a flag for non-standard input file. JBS 26-Mar-1982
116	0116	1	2-047	- Add a rename module. SMB 31-Mar-1982
117	0117	1	2-048	- Change CLOSE to delete an output file or journal file. SMB 08-Apr-1982
118	0118	1	2-049	- Make the FORWARD declaration of CLOSE agree with the body. JBS 09-Apr-1982
119	0119	1	2-050	- Remove EDT\$\$GET_IFIATTR, no longer used. JBS 09-Apr-1982
120	0120	1	2-051	- Don't do 0-length terminal QIOs. JBS 09-Apr-1982
121	0121	1	2-052	- Make EDT\$\$WF_CLS return a value and use the new message. JBS 12-Apr-1982
122	0122	1	2-053	- Add the parse routine. SMB 13-Apr-1982
123	0123	1	2-054	- Get device type during parse of output file. SMB 26-Apr-1982
124	0124	1	2-055	- Fix bug in RENAMING given an explicit file version number. SMB 04-May-1982
125	0125	1	2-056	- Check for Process Permanent File on the Parse of output file. SMB 11-May-1982
126	0126	1	2-057	- Make minor modifications based on code review input. SMB 25-May-1982
127	0127	1	2-058	- Don't check for record too long--check in EDT\$WFREAINP instead. JBS 02-Jun-1982
128	0128	1	2-059	- Change file attributes on EXIT with sequenced file. SMB 07-Jun-1982
129	0129	1	2-060	- Pass back STV on parse of output file. SMB 14-Jun-1982
130	0130	1	2-061	- Revise handling of file name defaulting. JBS 16-Jun-1982
131	0131	1	2-062	- Pass back STV on flush of output buffer. STS 01-Jul-1982
132	0132	1	2-063	- Use an action routine with \$PUTMSG so we can control access to the screen.
133	0133	1		JBS 06-Jul-1982
134	0134	1	2-064	- Make the status and stv vector global. STS 20-Jul-1982
135	0135	1	2-065	- Save and restore all terminal characteristics. SMB 28-Jul-1982
136	0136	1	2-066	- Correctly report terminal read errors. STS 06-Aug-1982
137	0137	1	2-067	- Add EDT\$\$INIT_IO. JBS 27-Aug-1982
138	0138	1	2-068	- Add EDT\$\$G_TI_SCROLL and start on long typeahead. JBS 31-Aug-1982
139	0139	1	2-069	- Add EDT\$\$G_TI_EDIT; if the terminal is not a VT100,
140	0140	1		don't use autorepeat. JBS 02-Sep-1982
141	0141	1	2-070	- Use escape mode to minimize the number of QIOs required. JBS 03-Sep-1982
142	0142	1	2-071	- Don't read the typeahead buffer in escape mode. JBS 07-Sep-1982
143	0143	1	2-072	- Don't use escape mode when either tracing is active or a keypad
144	0144	1		prompt is specified. This is to keep the QA system working,
145	0145	1		since we don't implement escape mode on the PDP-11s. JBS 08-Sep-1982
146	0146	1	2-073	- Add another case of not using escape mode when there is a keypad
147	0147	1		prompt. JBS 09-Sep-1982
148	0148	1	2-074	- In trace mode, initialize with autorepeat enabled, for PDP-11 compatibility.
149	0149	1		JBS 10-Sep-1982
150	0150	1	2-075	- Add EDT\$\$G_TI_DUMB. JBS 16-Oct-1982
151	0151	1	2-076	- Try to reduce the number of buffered I/Os. JBS 10-Nov-1982
152	0152	1	2-077	- Don't do copies to and from workfile - but pass addresses. 11-Nov-1982
153	0153	1	2-078	- Allocate cache only as needed, and open workfile only after
154	0154	1		cache is exhausted. STS 16-Nov-1982
155	0155	1	2-079	- VT125s should default to AUTOREPEAT. JBS 01-Dec-1982
156	0156	1	2-080	- Signal any errors from opening the terminal. STS 13-Dec-1982
157	0157	1	2-081	- Signal errors reading from terminal and add more detail
158	0158	1		to error messages when opening terminal fails. STS 15-Dec-1982
159	0159	1	2-082	- Add routine edt\$\$ti_deas to deassign the terminal channel. STS 21-Dec-1982
160	0160	1	2-083	- In EDT\$\$TI_DEAS, set OWN storage to its initial state and signal any errors. JBS 28-Dec-1982
161	0161	1	2-084	- Don't deassign the terminal channel if tracing. JBS 29-Dec-1982
162	0162	1	2-085	- Allocate window blocks from the end to speed up FIND_WINDOW, and do some code bumming in FIND_WIND
163	0163	1		to the same end. JBS 03-Jan-1983
164	0164	1	2-086	- Revise the logic for handling control Y so that we loop waiting for a character. JBS 19-Jan-1983
165	0165	1	2-087	- Add EDT\$\$TERM_RCC. JBS 26-Jan-1983
166	0166	1	2-088	- Remove reference to EDT\$\$OUT_ERRMSG, not used. JBS 22-Mar-1983
167	0167	1	2-089	- Don't do fancy output if the file type is .TMP. JBS 01-Apr-1983
168	0168	1	2-090	- On exit, set application keypad mode or numeric keypad mode, as the user requests. JBS 01-Apr-1983
169	0169	1	2-091	- Don't force maximum version number unless requested. JBS 04-Apr-1983
170	0170	1	2-092	- Improve the appearance of the listing. JBS 14-Jun-1983
171	0171	1	2-093	- Fix a typo caused by edit 2-092. JBS 16-Jun-1983

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS

N 11
16-Sep-1984 00:35:33 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:23:19 DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1 Page 4
(1)

```
: 172      0172 1 2-094 - Defend the terminal QIOs against SYSSINPUT not being a terminal. JBS 21-Jun-1983
: 173      0173 1 2-095 - Try improving performance by extending files less often. JBS 22-Aug-1983
: 174      0174 1 2-096 - More performance improvements based on Chris Saether's advice. JBS 29-Aug-1983
: 175      0175 1 2-097 - Add test for VT55 and treat it as a VT52. REM 3-Nov-1983
: 176      0176 1 2-098 - Support EDT$$G_DEC CRT. REM 2-Mar-1984
: 177      0177 1 2-099 - Add logic to make TT2$V_APP_KEYPAD & TT2$V_DECCRT2 always return false
: 178      0178 1 if they are not defined in STARLET. REM 6-Mar-1984
: 179      0179 1 2-100 - Removed macros for the previous fix and inserted conditional compila-
: 180      0180 1 tion around the usage of the cells in question. REM 19-Mar-1984
: 181      0181 1 2-101 - Added logic to cause the rename function (that renames the temp file
: 182      0182 1 to be the new output file) to propagate the ACL from the input file.
: 183      0183 1 This compiles only if XAB$B_PROT_OPT is defined. REM 21-Mar-1984
: 184      0184 1 2-102 - Fixed conditional compilation so that VT102s etc. get EDT$$G_DEC_CRT
: 185      0185 1 properly set. REM 1-May-1984
: 186      0186 1 --
: 187      0187 1
```

```
: 189 0188 1 %SBTTL 'Declarations'
: 190 0189 1
: 191 0190 1 : TABLE OF CONTENTS:
: 192 0191 1 :
: 193 0192 1
: 194 0193 1 REQUIRE 'EDT$SRC:TRAROUNAM';
: 195 0632 1
: 196 0633 1 FORWARD ROUTINE
: 197 0634 1 EDT$$IO_ERRMSG : NOVALUE,
: 198 0635 1 PUTMSG ACTION,
: 199 0636 1 EDT$$FATAL_IOERR : NOVALUE,
: 200 0637 1 EDT$$INIT_IO : NOVALUE,
: 201 0638 1 EDT$$WF_OPN,
: 202 0639 1 EDT$$WF_CLS,
: 203 0640 1 WF_OPN,
: 204 0641 1 FIND_WINDOW,
: 205 0642 1 EDT$$WF_WR : NOVALUE,
: 206 0643 1 WRITE_WINDOW : NOVALUE,
: 207 0644 1 REPLACE_WINDOW,
: 208 0645 1 GET_WF : NOVALUE,
: 209 0646 1 EDT$$WF_RD : NOVALUE,
: 210 0647 1 EDT$$PAR_FNAME,
: 211 0648 1 EDT$$OPN_INOUT,
: 212 0649 1 EDT$$OPN_IFIDEF,
: 213 0650 1 EDT$$OPN_OFIDEF,
: 214 0651 1 EDT$$CLS_FI,
: 215 0652 1 EDT$$RD_IFI,
: 216 0653 1 EDT$$WR_OFI,
: 217 0654 1 EDT$$REN_FI,
: 218 0655 1 EDT$$FLUSH_OBUF,
: 219 0656 1 CC_ENABLE,
: 220 0657 1 CC_AST : NOVALUE,
: 221 0658 1 EDT$$TERM_RCC : NOVALUE,
: 222 0659 1 EDT$$TI_OPN,
: 223 0660 1 EDT$$TI_RES,
: 224 0661 1 EDT$$TI_DEAS : NOVALUE,
: 225 0662 1 EDT$$RD_CMDLN,
: 226 0663 1 EDT$$TI_WRLN : NOVALUE,
: 227 0664 1 EDT$$TI_GETCH : NOVALUE,
: 228 0665 1 EDT$$TI_WSTR : NOVALUE,
: 229 0666 1 EDT$$TI_RDTYAHED,
: 230 0667 1 EDT$$TI_RDSTR : NOVALUE,
: 231 0668 1 EDT$$TI_ENTERCHM : NOVALUE,
: 232 0669 1 EDT$$TI_LEAVECHM : NOVALUE;
: 233 0670 1
: 234 0671 1 :
: 235 0672 1 : INCLUDE FILES:
: 236 0673 1 :
: 237 0674 1
: 238 0675 1 REQUIRE 'EDT$SRC:EDTREQ';
: 239 0810 1
: 240 0811 1 REQUIRE 'EDT$SRC:SYSSYM';
: 241 0841 1
: 242 0842 1 REQUIRE 'TRACESEL';
: 243 0873 1
: 244 0874 1 REQUIRE 'EDT$SRC:TRACEMAC';
: 245 1101 1
```

! Turn tracing on or off

! Define \$\$TRACE macro

```
246 1102 1 |
247 1103 1 | MACROS:          None
248 1104 1 |
249 1105 1 |
250 1106 1 | EQUATED SYMBOLS:
251 1107 1 |
252 1108 1 |
253 1109 1 | FIELD
254 1110 1 |     ITEM_FIELDS =           ! Item list fields
255 1111 1 |     SET
256 1112 1 |         BF_LEN = [0, 0, 16, 0], ! Buffer length
257 1113 1 |         I_CODE = [0, 16, 16, 0], ! Item code
258 1114 1 |         BOF_ADR = [1, 0, 32, 0], ! Buffer address
259 1115 1 |         RLEN_ADR = [2, 0, 32, 0], ! Return length address
260 1116 1 |     TES;
261 1117 1 |
262 1118 1 | FIELD
263 1119 1 |     SETM_FIELDS =           ! SETMODE fields
264 1120 1 |     SET
265 1121 1 |         S_CLASS = [0, 0, 8, 0], ! Class
266 1122 1 |         S_TYPE = [0, 8, 8, 0], ! Type
267 1123 1 |         S_WID = [0, 16, 16, 0], ! Terminal width
268 1124 1 |         S_CHAR = [1, 0, 32, 0], ! Characteristics
269 1125 1 |     TES;
270 1126 1 |
271 1127 1 | LITERAL
272 1128 1 |     EDT$K_FAC_NO = 133,
273 1129 1 |     WINDOW_SIZE = 11,
274 1130 1 |     NUM_WINDOW = 100;
275 1131 1 |
276 1132 1 |
277 1133 1 | OWN STORAGE:
278 1134 1 |
279 1135 1 |
280 1136 1 | OWN
281 1137 1 |     CACHE_ADDR : VECTOR [NUM_WINDOW], ! The start address of the heap storage for this window
282 1138 1 |     CACHE_TIMER, ! Counter incremented for each reference
283 1139 1 |     CACHE_WINDOW : VECTOR [NUM_WINDOW], ! The number of the window in a slot
284 1140 1 |     CACHE_REF : VECTOR [NUM_WINDOW], ! The timer at the last reference to this window
285 1141 1 |     CACHE_DIRTY : VECTOR [NUM_WINDOW, BYTE]; ! The window dirty flag.
286 1142 1 |
287 1143 1 | OWN
288 1144 1 |     WF_CACHE_FULL,
289 1145 1 |     WF_OPEN,
290 1146 1 |     MYBUF : BLOCK [CH$ALLOCATION (256)],
291 1147 1 |     MYBUF2 : BLOCK [CH$ALLOCATION (256)],
292 1148 1 |     INPUT_NAME : BLOCK [NAM$C_MAXRSS, BYTE],
293 1149 1 |     INPUT_NAME_LEN,
294 1150 1 |     INPUT_RAT,
295 1151 1 |     INPUT_RFM,
296 1152 1 |     INPUT_PROT,
297 1153 1 |     INPUT_MRS,
298 1154 1 |     ATTRIBUTES_SET : INITIAL (0), ! 1 = input attributes have been set
299 1155 1 |     ATTRIBUTES_DEFAULTED : INITIAL (0), ! 1 = input attributes = default attributes
300 1156 1 |     JOURNAL_NAM : $NAM DECL, ! NAM block for the journal file (so we can delete it)
301 1157 1 |     NAM_BLOCK : $NAM DECL, ! NAM block for last output file opened (so we can delete it)
302 1158 1 |     TERM_IN_RAB : $RAB DECL,
```

```

303 1159 1 TERM_OUT_RAB : $RAB DECL,
304 1160 1 IOSB : VECTOR [4, WORD],
305 1161 1 TT_CHAN,
306 1162 1 TT_OPEN : INITIAL (0),
307 1163 1 T_DEVCHAR : BLOCK [4, BYTE],
308 1164 1 t_devclass : BLOCK [4, BYTE],
309 1165 1 T_DEVDEP : BLOCK [4, BYTE],
310 1166 1 t_devextdep : BLOCK [4, BYTE],
311 1167 1 t_devtype : BLOCK [4, BYTE],
312 1168 1 T_DEVWID : BLOCK [4, BYTE],
313 1169 1 INPUT_EFN,
314 1170 1 OUTPUT_EFN,
315 1171 1 HAVE_EFNS : INITIAL (0),
316 1172 1 WF_IFI,
317 1173 1 IO_WF_RAB : $RAB DECL,
318 1174 1 IO_MVEC : INITIAL (2),
319 1175 1 IO_STS,
320 1176 1 IO_STV;
321 1177 1
322 1178 1
323 1179 1 ! EXTERNAL REFERENCES:
324 1180 1
325 1181 1
326 1182 1 EXTERNAL ROUTINE
327 1183 1 EDT$FMT_STR,
328 1184 1 EDT$FMT_CRLF,
329 1185 1 EDT$FMT_MSG,
330 1186 1 EDT$SC_RESET,
331 1187 1 EDT$MSG_TOSTR,
332 1188 1 EDT$ALO_HEAP,
333 1189 1 EDT$DEA_HEAP : NOVALUE;
334 1190 1
335 1191 1 EXTERNAL
336 1192 1 EDT$G_TI_PLEN,
337 1193 1 EDT$G_TI_RESET,
338 1194 1 EDT$G_J00_IFI,
339 1195 1 EDT$G_EDIT_MOD,
340 1196 1 EDT$G_TI_TYP,
341 1197 1 EDT$G_TI_WID,
342 1198 1 EDT$G_CC : VOLATILE,
343 1199 1 EDT$G_DEC_CRT,
344 1200 1 EDT$G_EIGHT_BIT,
345 1201 1 EDT$T_FMT_BOF,
346 1202 1 EDT$A_FMT_CUR,
347 1203 1 EDT$T_RDARED : BLOCK [CH$ALLOCATION (K_RDAHED LEN)],
348 1204 1 EDT$G_RDAHED,
349 1205 1 EDT$G_TI_SCROLL,
350 1206 1 EDT$G_TI_EDIT,
351 1207 1 EDT$G_ENB_AUTRPT,
352 1208 1 EDT$G_TI_DUMB,
353 1209 1 EDT$T_PMT_KPD : VECTOR [, BYTE];
354 1210 1
355 1211 1 OWN
356 1212 1 VFCHD : BLOCK [CH$ALLOCATION (255)];
357 1213 1
358 1214 1 GLOBAL
359 1215 1 EDT$G_IO_MVEC : INITIAL (2),
```

! 1 = terminal open
! Terminal independent characteristics
! Device class
! Terminal dependent characteristics
! Terminal extended characteristics
! Terminal type
! Terminal width

! 1 = event flags allocated

! Message to print from EDT\$\$IO_ERRMSG

! Terminate a line

! Read-ahead buffer
! Number of characters in read-ahead buffer
! 1 = terminal has scrolling regions
! 1 = terminal has editing features (ICM, DCH, IL, DL)
! 1 = OK to use Autorepeat
! 1 = terminal driver needs CR to avoid wrapping lines
! Counted string of keypad prompt

! Sequence no. in VFC files.

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
Declarations

E 12
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1 Page 8
(2)

```
: 360      1216 1      EDT$G_IO_STS;
: 361      1217 1      EDT$G_IO_STV;
: 362      1218 1
: 363      1219 1      %IF edt$Str_act
: 364      1220 1      %THEN
: 365      1221 1
: 366      1222 1      GLOBAL
: 367      1223 1      EDT$L_TR_INFLG;                ! 1 iff input from trace file
: 368      1224 1
: 369      1225 1      %FI
: 370      1226 1
: 371      1227 1      EXTERNAL ROUTINE
: 372      1228 1      STR$COPY_R,
: 373      1229 1      LIB$GET_EF,
: 374      1230 1      LIB$FREE_EF;
: 375      1231 1
: 376      1232 1      !+
: 377      1233 1      ! This module uses the following message codes:
: 378      1234 1      !-
: 379      1235 1      MESSAGES ((EDITORABO, OPNINTRM, OPNOUTTRM, ERRINPTRM, INSMEM, IOERRWRK, WORFILCLO));
```

```

381 1236 1 %SBTTL 'EDT$$IO_ERRMSG - print an I/O error message'
382 1237 1
383 1238 1 GLOBAL ROUTINE EDT$$IO_ERRMSG ! Print an I/O error message
384 1239 1 : NOVALUE =
385 1240 1
386 1241 1 ++
387 1242 1 FUNCTIONAL DESCRIPTION:
388 1243 1
389 1244 1 Put out an I/O (RMS) error message. The status and stv have already
390 1245 1 been placed in the message vector by fiohndlr.
391 1246 1
392 1247 1 FORMAL PARAMETERS:
393 1248 1
394 1249 1 NONE
395 1250 1
396 1251 1 IMPLICIT INPUTS:
397 1252 1
398 1253 1 EDT$$G_IO_MVEC
399 1254 1
400 1255 1 IMPLICIT OUTPUTS:
401 1256 1
402 1257 1 NONE
403 1258 1
404 1259 1 ROUTINE VALUE:
405 1260 1
406 1261 1 NONE
407 1262 1
408 1263 1 SIDE EFFECTS:
409 1264 1
410 1265 1 Sends text to the screen.
411 1266 1
412 1267 1 --
413 1268 1
414 1269 2 BEGIN
415 1270 2 $PUTMSG (MSGVEC = EDT$$G_IO_MVEC, ACTRTN = PUTMSG_ACTION);
416 1271 1 END; ! of routine EDT$$IO_ERRMSG
```

.TITLE EDT\$IOMOD EDT\$IOMOD - I/O for VAX/VMS
.IDENT \V04-000\

.PSECT _EDT\$DATA,NOEXE, PIC,2

00000	CACHE_ADDR:	
	.BLKB	400
00190	CACHE_TIMER:	
	.BLKB	4
00194	CACHE_WINDOW:	
	.BLKB	400
00324	CACHE_REF:	
	.BLKB	400
00484	CACHE_DIRTY:	
	.BLKB	100
00518	WF_CACHE_FULL:	
	.BLKB	4
0051C	WF_OPEN:	.BLKB
		4
00520	MYBUF:	.BLKB
		256

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$IO_ERRMSG - print an I/O error message

G 12
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742 Page 10
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1 (3)

EDTS
V04-

	00620	MYBUF2:	.BLKB	256
	00720	INPUT_NAME:		
			.BLKB	255
	0081F		.BLKB	1
	00820	INPUT_NAME_LEN:		
			.BLKB	4
	00824	INPUT_RAT:		
			.BLKB	4
	00828	INPUT_RFM:		
			.BLKB	4
	0082C	INPUT_PROT:		
			.BLKB	4
	00830	INPUT_MRS:		
			.BLKB	4
00000000	00834	ATTRIBUTES_SET:		
			.LONG	0
00000000	00838	ATTRIBUTES_DEFAULTED:		
			.LONG	0
	0083C	JOURNAL_NAM:		
			.BLKB	96
	0089C	NAM_BLOCK:		
			.BLKB	96
	008FC	TERM_IN_RAB:		
			.BLKB	68
	00940	TERM_OUT_RAB:		
			.BLKB	68
	00984	IOSB:	.BLKB	8
	0098C	TT_CHAN:	.BLKB	4
00000000	00990	TT_OPEN:	.LONG	0
	00994	T_DEVCHAR:		
			.BLKB	4
	00998	T_DEVCLASS:		
			.BLKB	4
	0099C	T_DEVDEP:		
			.BLKB	4
	009A0	T_DEVEXTDEP:		
			.BLKB	4
	009A4	T_DEVTYPE:		
			.BLKB	4
	009A8	T_DEVWID:		
			.BLKB	4
	009AC	INPUT_EFN:		
			.BLKB	4
	009B0	OUTPUT_EFN:		
			.BLKB	4
00000000	009B4	HAVE_EFNS:		
			.LONG	0
	009B8	WF_IF1:	.BLKB	4
	009BC	IO_WF_RAB:		
			.BLKB	68
00000002	00A00	IO_MVEC:	.LONG	2
	00A04	IO_STS:	.BLKB	4
	00A08	IO_STV:	.BLKB	4
	00A0C	VFCHD:	.BLKB	256
00000002	00B0C	EDT\$\$\$IO_MVEC:		
			.LONG	2
	00B10	EDT\$\$\$IO_STS:		

[illegible]

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$IO_ERRMSG - print an I/O error message

H 12
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1 Page 11
(3)

EDT\$
V04-

; Ro

; 7

00B14 EDT\$\$G_IO STV::
.BLKB 4
.BLKB 4

.EXTRN EDT\$\$FMT_STR, EDT\$\$FMT_CRLF
.EXTRN EDT\$\$FMT_MSG, EDT\$\$SC_RESET
.EXTRN EDT\$\$MSG_TOSTR, EDT\$\$ALO_HEAP
.EXTRN EDT\$\$DEA_HEAP, EDT\$\$G_TI_PLEN
.EXTRN EDT\$\$G_TI_RESET
.EXTRN EDT\$\$G_JOB_IFI, EDT\$\$G_EDIT_MOD
.EXTRN EDT\$\$G_TI_TYP, EDT\$\$G_TI_WID
.EXTRN EDT\$\$G_CC, EDT\$\$G_DEC_CRT
.EXTRN EDT\$\$G_EIGHT_BIT
.EXTRN EDT\$\$T_FMT_BOF, EDT\$\$A_FMT_CUR
.EXTRN EDT\$\$T_RDAED, EDT\$\$G_RDAED
.EXTRN EDT\$\$G_TI_SCROLL
.EXTRN EDT\$\$G_TI_EDIT, EDT\$\$G_ENB_AUTRPT
.EXTRN EDT\$\$G_TI_DUMB, EDT\$\$T_PMT_KPD
.EXTRN STRSCOPY_R, LIB\$GET_EF
.EXTRN LIB\$FREE_EF, EDT\$EDITORABO
.EXTRN EDT\$OPNINTRM, EDT\$OPNOUTTRM
.EXTRN EDT\$ERRINPTRM, EDT\$INSMEM
.EXTRN EDT\$IOERRWRK, EDT\$_QORFILCLO
.EXTRN SYSS\$PUTMSG

.PSECT _EDT\$CODE, NOWRT, SHR, PIC, 2

.ENTRY EDT\$\$IO_ERRMSG, Save nothing
CLRQ -(SP)
PUSHAB PUTMSG_ACTION
PUSHAB EDT\$\$G_IO_MVEC
CALLS #4, SYSS\$PUTMSG
RET

: 1238
: 1270
:
:
:
: 1271

0000 00000
7E 7C 00002
CF 9F 00004
EF 9F 00008
04 FB 0000E
04 00015
00000000G 00 00000000'

; Routine Size: 22 bytes, Routine Base: _EDT\$CODE + 0000

; 417 1272 1

```

419 1273 1 XSBTTL 'PUTMSG_ACTION - Dispose of $PUTMSG's text'
420 1274 1 ROUTINE PUTMSG_ACTION ( ! Dispose of $PUTMSG's text
421 1275 1 TEXT_DESC ! The text to be printed
422 1276 1 ) =
423 1277 1
424 1278 1 ++
425 1279 1 FUNCTIONAL DESCRIPTION:
426 1280 1
427 1281 1 This routine is called by $PUTMSG for each line to output.
428 1282 1 Output the text through EDT's formatting routines, and tell
429 1283 1 $PUTMSG to not output it.
430 1284 1
431 1285 1 FORMAL PARAMETERS:
432 1286 1
433 1287 1 TEST_DESC A descriptor for the line of text to be output
434 1288 1
435 1289 1 IMPLICIT INPUTS:
436 1290 1
437 1291 1 NONE
438 1292 1
439 1293 1 IMPLICIT OUTPUTS:
440 1294 1
441 1295 1 NONE
442 1296 1
443 1297 1 ROUTINE VALUE:
444 1298 1
445 1299 1 Always 0, to tell $PUTMSG not to output the text itself
446 1300 1
447 1301 1 SIDE EFFECTS:
448 1302 1
449 1303 1 Sends text to the screen.
450 1304 1
451 1305 1 --
452 1306 1
453 1307 2 BEGIN
454 1308 2
455 1309 2 MAP
456 1310 2 TEXT_DESC : REF BLOCK [8, BYTE];
457 1311 2
458 1312 2 +
459 1313 2 Send the string to the format buffer.
460 1314 2 -
461 1315 2 EDT$$FMT_STR (.TEXT_DESC [DSC$A_POINTER], .TEXT_DESC [DSC$W_LENGTH]);
462 1316 2 +
463 1317 2 Terminate the line.
464 1318 2 -
465 1319 2 EDT$$FMT_CRLF ();
466 1320 2 +
467 1321 2 Tell $PUTMSG to not output the line.
468 1322 2 -
469 1323 2 RETURN (0);
470 1324 1 END; ! of routine PUTMSG_ACTION

```

```
EDT$IOMOD - I/O for VAX/VMS
PUTMSG_ACTION - Dispose of $PUTMSG's text
```

J 12
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

```
VAX-11 Bliss-32 V4.0-742
DISK$VMSMASTER:[EDT.SRC]IOMOD.B32;1
```

Page 13
(4)

EDTS
V04-

	50	04	AC	D0	00002
	7E		60	3C	00006
		04	A0	DD	00009
000000000G	00		02	FB	0000C
000000000G	00		00	FB	00013
			50	D4	0001A
				04	0001C

```

        .WORD      Save nothing
        MOVL       TEXT_DESC, R0
        MOVZWL     (R0), -(SP)
        PUSHL      4(R0)
        CALLS      #2, EDT$$FMT_STR
        CALLS      #0, EDT$$FMT_CRLF
        CLRL       R0
        RET

```

: 1274
 : 1315
 :
 :
 :
 :
 :
 : 1319
 : 1323
 : 1324

```
; Routine Size: 29 bytes,    Routine Base: _EDT$CODE + 0016
```

 $\epsilon = R_c$

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS

EDT\$\$FATAL_IOERR - put out an I/O error messag

K 12

16-Sep-1984 00:35:33

14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 14

(5)

```

472 1325 1 %SBTTL 'EDT$$FATAL_IOERR - put out an I/O error message and abort'
473 1326 1
474 1327 1 GLOBAL ROUTINE EDT$$FATAL_IOERR (          ! Put out an I/O error message and abort
475 1328 1      N                                ! Not used
476 1329 1      ) : NOVALUE =
477 1330 1
478 1331 1      ++
479 1332 1      FUNCTIONAL DESCRIPTION:
480 1333 1
481 1334 1          Put out an I/O error message and abort.
482 1335 1
483 1336 1      FORMAL PARAMETERS:
484 1337 1
485 1338 1          N                Not used.
486 1339 1
487 1340 1      IMPLICIT INPUTS:
488 1341 1
489 1342 1          EDT$$G_EDIT_MOD
490 1343 1
491 1344 1      IMPLICIT OUTPUTS:
492 1345 1
493 1346 1          NONE
494 1347 1
495 1348 1      ROUTINE VALUE:
496 1349 1
497 1350 1          NONE
498 1351 1
499 1352 1      SIDE EFFECTS:
500 1353 1
501 1354 1          Calls EDT$$SC_RESET and EDT$$IO_ERRMSG .
502 1355 1          Never returns to its caller.
503 1356 1      --
504 1357 1
505 1358 2      BEGIN
506 1359 2      +
507 1360 2      Reset any screen modes.
508 1361 2      -
509 1362 2
510 1363 2      IF (.EDT$$G_EDIT_MOD EQL CHANGE_MODE) THEN EDT$$SC_RESET ();
511 1364 2
512 1365 2      EDT$$IO_ERRMSG ();
513 1366 2      SIGNAL_STOP (EDT$_EDITORABO);
514 1367 1      END;                                     ! of routine EDT$$FATAL_IOERR
```

		0000	00000	.ENTRY	EDT\$\$FATAL_IOERR, Save nothing	: 1327
		00000000G	00 D5 00002	TSTL	EDT\$\$G_EDIT_MOD	: 1363
			07 12 00008	BNEQ	1\$	
00000000G	00		00 FB 0000A	CALLS	#0, EDT\$\$SC_RESET	
B8	AF		00 FB 00011	CALLS	#0, EDT\$\$IO_ERRMSG	: 1365
		00000000G	8F DD 00015	PUSHL	#EDT\$_EDITORABO	: 1366
00000000G	00		01 FB 0001B	CALLS	#1, LTB\$STOP	
			04 00022	RET		: 1367

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$FATAL_IOERR - put out an I/O error messag

L 12
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1 Page 15
(5)

; Routine Size: 35 bytes, Routine Base: _EDT\$CODE + 0033

; 515 1368 1

EDT\$
V04-

; Ro

; 8

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$INIT_IO - Initialize the I/O system

M 12
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 16
(6)

EDT\$
V04-

```
: 517 1369 1 %SBTTL 'EDT$$INIT_IO - Initialize the I/O system'
: 518 1370 1
: 519 1371 1 GLOBAL ROUTINE EDT$$INIT_IO ! Initialize the I/O system
: 520 1372 1 : NOVALUE =
: 521 1373 1
: 522 1374 1 !++
: 523 1375 1 FUNCTIONAL DESCRIPTION:
: 524 1376 1
: 525 1377 1 Do any initialization needed before any devices are opened.
: 526 1378 1 On VMS, this is none. It must remain none because of callable EDT.
: 527 1379 1
: 528 1380 1 FORMAL PARAMETERS:
: 529 1381 1
: 530 1382 1 NONE
: 531 1383 1
: 532 1384 1 IMPLICIT INPUTS:
: 533 1385 1
: 534 1386 1 NONE
: 535 1387 1
: 536 1388 1 IMPLICIT OUTPUTS:
: 537 1389 1
: 538 1390 1 NONE
: 539 1391 1
: 540 1392 1 ROUTINE VALUE:
: 541 1393 1
: 542 1394 1 NONE
: 543 1395 1
: 544 1396 1 SIDE EFFECTS:
: 545 1397 1
: 546 1398 1 NONE
: 547 1399 1
: 548 1400 1 --
: 549 1401 1
: 550 1402 2 BEGIN
: 551 1403 2 RETURN;
: 552 1404 1 END; ! of routine EDT$$INIT_IO
```

0000 00000
04 00002

.ENTRY EDT\$\$INIT_IO, Save nothing
RET

: 1371
: 1404

; Routine Size: 3 bytes, Routine Base: _EDT\$CODE + 0056

; 553 1405 1

```

555 1406 1 %SBTTL 'WF_OPN - Open work file'
556 1407 1 ROUTINE WF_OPN                                ! Open work file
557 1408 1 =
558 1409 1
559 1410 1 ++
560 1411 1 FUNCTIONAL DESCRIPTION:
561 1412 1
562 1413 1     Open the work file. Abort if it fails.
563 1414 1     The work file is opened in SYS$SCRATCH. The cache area is allocated from virtual memory.
564 1415 1
565 1416 1 FORMAL PARAMETERS:
566 1417 1
567 1418 1     NONE
568 1419 1
569 1420 1 IMPLICIT INPUTS:
570 1421 1
571 1422 1     NONE
572 1423 1
573 1424 1 IMPLICIT OUTPUTS:
574 1425 1
575 1426 1     IO_STS
576 1427 1     IO_STV
577 1428 1     WF_IFI
578 1429 1     CACHE_ADDR
579 1430 1
580 1431 1 ROUTINE VALUE:
581 1432 1
582 1433 1     1
583 1434 1
584 1435 1 SIDE EFFECTS:
585 1436 1
586 1437 1     Opens the work file. Prints an error message and does not
587 1438 1     return if the open fails.
588 1439 1     Calls EDT$ALO_HEAP to get virtual memory to hold the cache.
589 1440 1
590 1441 1 --
591 1442 1
592 1443 2 BEGIN
593 1444 2
594 1445 2 LOCAL
595 1446 2     XABPRO : $XABPRO_DECL,
596 1447 2     WORK_FAB : $FAB_DECL;
597 1448 2
598 1449 2 ++
599 1450 2 Set up the FAB, RAB and XAB for the open.
600 1451 2
601 P 1452 2 $FAB_INIT (FAB = WORK_FAB, FAC = <GET, PUT, BIO>, FOP = <TMD>, ALQ = WINDOW_SIZE, RAT = CR, RFM = UDF,
602 1453 2     RTV = 20, SHR = NIL, XAB = XABPRO);
603 1454 2 WORK_FAB [FAB$FNA] = UPLIT (BYTE ('EDTWORK'));
604 1455 2 WORK_FAB [FAB$B_FNS] = %CHARCOUNT ('EDTWORK');
605 1456 2 WORK_FAB [FAB$F_DNA] = UPLIT (BYTE ('SYS$SCRATCH:TMP'));
606 1457 2 WORK_FAB [FAB$B_DNS] = %CHARCOUNT ('SYS$SCRATCH:TMP');
607 1458 2 $RAB_INIT (RAB = IO_WF_RAB, ROP = <BIO>, FAB = WORK_FAB, CTX = 0);
608 1459 2 $XABPRO_INIT (XAB = XABPRO, PRO = <D, RWD>);
609 1460 2
610 1461 3 IF (IO_STS = $RMS_CREATE (FAB = WORK_FAB))
611 1462 2 THEN
```

```
! of routine WF_OPN
```

[illegible]

EDT\$IOMOD - I/O for VAX/VM's
WF_OPN - Open work file

C 13
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

```
VAX-11 Bliss-32 V4.0-742
DISK$VMMASTER:[EDT.SRC]IOMOD.B32;1
```

Page 19
(7)

EDTS
V04-

	66		50	D0	0007D	MOVL	R0, IO_STS	:	
	1C		50	E9	00080	BLBC	R0, 1\$-	:	
B4	A6	02	AE	3C	00083	MOVZWL	WORK FAB+2, WF_IFI	:	1464
		B8	A6	9F	00088	PUSHAB	IO_WF_RAB	:	1466
00000000G	00		01	FB	0008B	CALLS	#1, SYS\$CONNECT	:	
	66		50	D0	00092	MOVL	R0, IO_STS	:	
	0C		50	E8	00095	BLBS	R0, 2\$-	:	
04	A6	C4	A6	D0	00098	MOVL	IO_WF_RAB+12, IO_STV	:	
			05	11	0009D	BRB	2\$-	:	1461
04	A6	0C	AE	D0	0009F	MOVL	WORK FAB+12, IO_STV	:	1470
	12		66	E8	000A4	BLBS	IO_STS, 3\$	:	1476
	7E		66	7D	000A7	MOVQ	IO_STS, -(SP)	:	
			7E	D4	000AA	CLRL	-(SP)	:	
		00000000G	8F	DD	000AC	PUSHL	#EDT\$ IOERRWRK	:	
00000000G	00		04	FB	000B2	CALLS	#4, LIB\$STOP	:	
FB18	C6		01	D0	000B9	MOVL	#1, WF_OPEN	:	1478
CO	A6		01	D0	000BE	MOVL	#1, IO_WF_RAB+8	:	1479
	50		01	D0	0C0C2	MOVL	#1, R0	:	1480
			04	000C5	RET			:	1481

```
; Routine Size: 198 bytes,    Routine Base: _EDT$CODE + 0074
```

[illegible]

```

: 632 1482 1 %SBTTL 'EDT$$WF_OPN - Open work file'
: 633 1483 1
: 634 1484 1 GLOBAL ROUTINE EDT$$WF_OPN . Open work file
: 635 1485 1 =
: 636 1486 1
: 637 1487 1 !++
: 638 1488 1 FUNCTIONAL DESCRIPTION:
: 639 1489 1
: 640 1490 1 Initialize work system
: 641 1491 1
: 642 1492 1 FORMAL PARAMETERS:
: 643 1493 1
: 644 1494 1 NONE
: 645 1495 1
: 646 1496 1 IMPLICIT INPUTS:
: 647 1497 1
: 648 1498 1 NONE
: 649 1499 1
: 650 1500 1 IMPLICIT OUTPUTS:
: 651 1501 1
: 652 1502 1 The work file cache.
: 653 1503 1
: 654 1504 1 ROUTINE VALUE:
: 655 1505 1
: 656 1506 1 Always 1; signals all errors.
: 657 1507 1
: 658 1508 1 SIDE EFFECTS:
: 659 1509 1
: 660 1510 1 NONE
: 661 1511 1
: 662 1512 1 --
: 663 1513 1
: 664 1514 2 BEGIN
: 665 1515 2
: 666 1516 2 DECR I FROM NUM_WINDOW - 1 TO 0 DO
: 667 1517 3 BEGIN
: 668 1518 3 CACHE_WINDOW [.I] = -1;
: 669 1519 3 CACHE_REF [.I] = 0;
: 670 1520 3 CACHE_DIRTY [.I] = 0;
: 671 1521 3 CACHE_ADDR [.I] = 0;
: 672 1522 2 END;
: 673 1523 2
: 674 1524 2 WF_OPEN = 0;
: 675 1525 2 WF_CACHE_FULL = 0;
: 676 1526 2 CACHE_TIMER = 0;
: 677 1527 2 RETURN (1);
: 678 1528 1 END;
! of routine EDT$$WF_OPN
```

```

0004 00000
52 00000000' EF 9E 00002
50 63 8F 9A 00009
6240 01 CE 0000D 1$:
0190 C240 D4 00011
```

```

.ENTRY EDT$$WF_OPN, Save R2
MOVAB CACHE_WINDOW, R2
MOVZBL #99, I
MNEGL #1, CACHE_WINDOW[I]
CLRL CACHE_REF[I]
```

```

: 1484
:
: 1516
: 1518
: 1519
```

[illegible]

```

681      1530 1 %SBTTL 'EDT$$WF_CLS -close the work file'
682      1531 1
683      1532 1 GLOBAL ROUTINE EDT$$WF_CLS ! Close the work file
684      1533 1 =
685      1534 1
686      1535 1 !++
687      1536 1 FUNCTIONAL DESCRIPTION:
688      1537 1
689      1538 1 Close the work file.
690      1539 1
691      1540 1 FORMAL PARAMETERS:
692      1541 1
693      1542 1 NONE
694      1543 1
695      1544 1 IMPLICIT INPUTS:
696      1545 1
697      1546 1 CACHE_ADDR
698      1547 1
699      1548 1 IMPLICIT OUTPUTS:
700      1549 1
701      1550 1 NONE
702      1551 1
703      1552 1 ROUTINE VALUE:
704      1553 1
705      1554 1 Always 1; signals all errors.
706      1555 1
707      1556 1 SIDE EFFECTS:
708      1557 1
709      1558 1 Errors are signalled.
710      1559 1 Releases all the heap storage in the work file cache.
711      1560 1
712      1561 1 --
713      1562 1
714      1563 2 BEGIN
715      1564 2
716      1565 2 LOCAL
717      1566 2 WORK_FAB : $FAB_DECL;
718      1567 2
719      1568 2 IF .WF_OPEN
720      1569 2 THEN
721      1570 3 BEGIN
722      1571 3 !+
723      1572 3 Disconnect the RAB
724      1573 3 -
725      1574 3 IO_STS = $RMS_DISCONNECT (RAB = IO_WF_RAB);
726      1575 3 IO_STV = IO_WF_RAB [RAB$L_STV];
727      1576 3
728      1577 3 IF .IO_STS
729      1578 3 THEN
730      1579 4 BEGIN
731      1580 4 !+
732      1581 4 Move the IFI into the FAB
733      1582 4 -
734      1583 4 $FAB_INIT (FAB = WORK_FAB);
735      1584 4 WORK_FAB [FAB$W_IFI] = .WF_IFI;
736      1585 4 !+
737      1586 4 And close the file.

```

```

738 1587 4 !-
739 1588 4      IO_STS = $RMS_CLOSE (FAB = WORK_FAB);
740 1589 4      IO_STV = .WORK_FAB [FAB$L_STV];
741 1590 3      END;
742 1591 3
743 1592 3      IF ( NOT .IO_STS) THEN SIGNAL_STOP (EDT$_WORFILCLO, 0, .IO_STS, .IO_STV);
744 1593 3
745 1594 2      END;
746 1595 2
747 1596 2 !+
748 1597 2 !- Now release the cache.
749 1598 2 !-
750 1599 2
751 1600 2      DECR I FROM NUM_WINDOW - 1 TO 0 DO
752 1601 2
753 1602 2      IF (.CACHE_ADDR [I] NEQA 0) THEN EDT$$DEA_HEAP (%REF (WINDOW_SIZE*512), CACHE_ADDR [I]);
754 1603 2
755 1604 2      RETURN (1);
756 1605 1      END;

```

! of routine EDT\$\$WF_CLS

```

                                .EXTRN  SYS$DISCONNECT, SYS$CLOSE
                                .ENTRY   EDT$$WF_CLS, Save R2,R3,R4,R5,R6
                                MOVAB    IO_STS, R6
                                MOVAB    -84(SP), SP
                                BLBC     WF_OPEN, 2$
                                PUSHAB   IO_WF_RAB
                                CALLS    #1, SYS$DISCONNECT
                                MOVL     R0, IO_STS
                                MOVAB    IO_WF_RAB+12, IO_STV
                                BLBC     IO_STS, 1$
                                MOVCS    #0, (SP), #0, #80, $RMS_PTR

                                MOVW     #20483, $RMS_PTR
                                MOVB     #2, $RMS_PTR+22
                                MOVB     #2, $RMS_PTR+31
                                MOVW     WF_IF1, WORK_FAB+2
                                PUSHAB   WORK_FAB
                                CALLS    #1, SYS$CLOSE
                                MOVL     R0, IO_STS
                                MOVL     WORK_FAB+12, IO_STV
                                BLBS     IO_STS, 2$
                                MOVQ     IO_STS, -(SP)
                                CLRL     -(SP)
                                PUSHL    #EDT$_WORFILCLO
                                CALLS    #4, LIB$STOP
                                MOVZBL   #99, I
                                TSTL     CACHE_ADDR[I]
                                BEQL     4$
                                PUSHAL   CACHE_ADDR[I]
                                MOVZWL   #5632, 4(SP)
                                PUSHAB   4(SP)
                                CALLS    #2, EDT$$DEA_HEAP
                                SOBGEQ   I, 3$
                                MOVL     #1, R0

```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$WF_CLS - close the work file

H 13
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1 Page 24
(9)

04 00090

RET

; 1605

; Routine Size: 145 bytes, Routine Base: _EDT\$CODE + 0168

; 757 1606 1

EDT\$
V04-

; Rc

```

759 1607 1 XSBTTL 'FIND_WINDOW - Find the window containing a bucket'
760 1608 1 ROUTINE FIND_WINDOW (
761 1609 1     BUKT_NUM,
762 1610 1     ADDR
763 1611 1 ) =
764 1612 1
765 1613 1 ++
766 1614 1 FUNCTIONAL DESCRIPTION:
767 1615 1
768 1616 1     Locate the window which contains a specified bucket in the cache. If it
769 1617 1     is not in the cache return -1; otherwise the window number is returned.
770 1618 1
771 1619 1 FORMAL PARAMETERS:
772 1620 1
773 1621 1     BUKT_NUM           The bucket we are trying to locate.
774 1622 1
775 1623 1     ADDR              Address of a longword to receive the address of the requested
776 1624 1                     bucket.
777 1625 1
778 1626 1 IMPLICIT INPUTS:
779 1627 1
780 1628 1     CACHE_WINDOW
781 1629 1     CACHE_ADDR
782 1630 1
783 1631 1 IMPLICIT OUTPUTS:
784 1632 1
785 1633 1     CACHE_TIMER         Incremented if window found
786 1634 1     CACHE_REF          Set to CACHE_TIMER for the window found
787 1635 1
788 1636 1 ROUTINE VALUE:
789 1637 1
790 1638 1     The number of the window containing this bucket, or -1 if none found.
791 1639 1
792 1640 1 SIDE EFFECTS:
793 1641 1
794 1642 1     NONE
795 1643 1
796 1644 1 --
797 1645 1
798 1646 2 BEGIN
799 1647 2
800 1648 2 LOCAL
801 1649 2     P_CACHE_WINDOW : REF VECTOR [NUM_WINDOW],      ! to improve the code in the loop
802 1650 2     WINDOW_NUM;
803 1651 2
804 1652 2 +
805 1653 2 Compute the window which contains this bucket.
806 1654 2 -
807 1655 2     WINDOW_NUM = (.BUKT_NUM - 1)/WINDOW_SIZE;
808 1656 2 +
809 1657 2 Search the cache to find the window, if found, then compute the address
810 1658 2 of the requested bucket and return it.
811 1659 2 -
812 1660 2     P_CACHE_WINDOW = CACHE_WINDOW [0];
813 1661 2
814 1662 2     DECR I FROM NUM_WINDOW - 1 TO 0 DO
815 1663 2
```

```
! of routine FIND_WINDOW
```

: 12

```
; Routine Size: 70 bytes,    Routine Base: _EDT$CODE + 01F9
```

[illegible]

! of routine EDTSSWF_WR

[illegible]

EDISIOMOD - I/O for VAX/VMS
WRITE_WINDOW - write out a window

M 13
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32:1

Page 29
(12)

EDTS
V04-

```

885 1731 1 XSBTTL 'WRITE_WINDOW - write out a window'
886 1732 1 ROUTINE WRITE_WINDOW (
887 1733 1     WINDOW_NUM
888 1734 1     ) : NOVALUE =
889 1735 1
890 1736 1 ++
891 1737 1 FUNCTIONAL DESCRIPTION
892 1738 1
893 1739 1     Write out an entire window.
894 1740 1
895 1741 1 FORMAL PARAMETERS:
896 1742 1
897 1743 1     WINDOW_NUM           The slot from which the window is written.
898 1744 1
899 1745 1 IMPLICIT INPUTS:
900 1746 1
901 1747 1     CACHE_ADDR
902 1748 1     CACHE_WINDOW
903 1749 1
904 1750 1 IMPLICIT OUTPUTS:
905 1751 1
906 1752 1     CACHE_DIRTY
907 1753 1     IO_WF_RAB
908 1754 1     IO_ST5
909 1755 1     IO_STV
910 1756 1
911 1757 1 ROUTINE VALUE:
912 1758 1
913 1759 1     NONE
914 1760 1
915 1761 1 SIDE EFFECTS:
916 1762 1
917 1763 1     Writes on the work file.
918 1764 1     I/O errors print an error and do not return to the caller.
919 1765 1
920 1766 1 --
921 1767 1
922 1768 2 BEGIN
923 1769 2 +
924 1770 2 Mark the window as clean.
925 1771 2 -
926 1772 2     CACHE_DIRTY [.WINDOW_NUM] = 0;
927 1773 2
928 1774 2     IF ( NOT .WF_OPEN) THEN WF_OPN ();
929 1775 2
930 1776 2 +
931 1777 2 Setup the RAB with the address and length and record number.
932 1778 2 -
933 1779 2     IO_WF_RAB [RAB$R_RBF] = .CACHE_ADDR [.WINDOW_NUM];
934 1780 2     IO_WF_RAB [RAB$W_RSZ] = WINDOW_SIZE*512;
935 1781 2     IO_WF_RAB [RAB$L_BKT] = .CACHE_WINDOW [.WINDOW_NUM]*WINDOW_SIZE + 1;
936 1782 2 +
937 1783 2 Attempt to write it out.
938 1784 2 -
939 1785 2
940 1786 2 WHILE 1 DO
941 1787 2

```

```

: 942      1788 3      IF ( NOT (IO_STS = $RMS_WRITE (RAB = IO_WF_RAB)))
: 943      1789 2      THEN
: 944      1790 2
: 945      1791 3      IF (.IO_STS EQL RMS$_EOF)
: 946      1792 2      THEN
: 947      1793 2      EXITLOOP
: 948      1794 2      ELSE
: 949      1795 3      BEGIN
: 950      1796 3      IO_STV = .IO_WF_RAB [RAB$_STV];
: 951      1797 3      SIGNAL_STOP (EDT$_IOERRWRK, 0, .IO_STS, .IO_STV);
: 952      1798 3      EXITLOOP;
: 953      1799 3      END
: 954      1800 3
: 955      1801 2      ELSE
: 956      1802 2      EXITLOOP;
: 957      1803 2
: 958      1804 1      END;

```

! of routine WRITE_WINDOW

.EXTRN SYSS\$WRITE

```

                                000C 00000 WRITE_WINDOW:
                                .WORD Save R2,R3
                                MOVAB IO_STS, R3
                                MOVL WINDOW_NUM, R2
                                CLRB CACHE_DIRTY[R2]
                                BLBS WF_OPEN, 1$
                                CALLS #0, WF_OPN
                                MOVL CACHE_ADDR[R2], IO_WF_RAB+40
                                MOVW #5632, IO_WF_RAB+32
                                MULL3 #11, CACHE_WINDOW[R2], R0
                                MOVAB 1(R0), IO_WF_RAB+56
                                PUSHAB IO_WF_RAB
                                CALLS #1, SYSS$WRITE
                                MOVL R0, IO_STS
                                BLBS R0, 2$
                                CMPL IO_STS, #98938
                                BEQL 2$
                                MOVL IO_WF_RAB+12, IO_STV
                                MOVQ IO_STS, -(SP)
                                CLRL -(SP)
                                PUSHL #EDT$_IOERRWRK
                                CALLS #4, LIB$STOP
                                RET
00000000G 00 00000000G 04 00065 2$:

```

; Routine Size: 102 bytes, Routine Base: _EDT\$CODE + 0262

; Ro

; 13

```

960 1805 1 %SBTTL 'REPLACE_WINDOW - read a new window into the cache'
961 1806 1 ROUTINE REPLACE_WINDOW (
962 1807 1     BUKT_NUM
963 1808 1 ) =
964 1809 1
965 1810 1 ++
966 1811 1 FUNCTIONAL DESCRIPTION:
967 1812 1
968 1813 1     This routine is called when a new work-file window must be read into the
969 1814 1     cache. We find the least recently used window for replacement, and write
970 1815 1     it out.
971 1816 1
972 1817 1 FORMAL PARAMETERS:
973 1818 1
974 1819 1     BUKT_NUM             the number of the bucket which we need to put in the cache.
975 1820 1
976 1821 1 IMPLICIT INPUTS:
977 1822 1
978 1823 1     CACHE_REF
979 1824 1     CACHE_DIRTY
980 1825 1
981 1826 1 IMPLICIT OUTPUTS:
982 1827 1
983 1828 1     CACHE_TIMER
984 1829 1     CACHE_REF
985 1830 1     CACHE_WINDOW
986 1831 1
987 1832 1 ROUTINE VALUE:
988 1833 1
989 1834 1     The window number the bucket was allocated to.
990 1835 1
991 1836 1 SIDE EFFECTS:
992 1837 1
993 1838 1     May call WRITE_WINDOW
994 1839 1
995 1840 1 --
996 1841 1
997 1842 2 BEGIN
998 1843 2
999 1844 2 LOCAL
1000 1845 2     WINDOW_NUM,
1001 1846 2     OLD_TIME,
1002 1847 2     OLD_NUM;
1003 1848 2
1004 1849 2 +
1005 1850 2 Determine which window the bucket is in.
1006 1851 2 -
1007 1852 2     WINDOW_NUM = (.BUKT_NUM - 1)/WINDOW_SIZE;
1008 1853 2
1009 1854 2     IF (.WINDOW_NUM GEQ NUM_WINDOW) THEN WF_CACHE_FULL = 1;
1010 1855 2
1011 1856 2     IF .WF_CACHE_FULL
1012 1857 2     THEN
1013 1858 3         BEGIN
1014 1859 3
1015 1860 3 +
1016 1861 3 Find the least recently used window in the cache.
1016 1861 3 -
```

```
1017 1862 3 OLD_TIME = 999999999;
1018 1863 3
1019 1864 3 DECR I FROM NUM_WINDOW - 1 TO 0 DO
1020 1865 3
1021 1866 4 IF (.CACHE_REF [.I] LSSU .OLD_TIME)
1022 1867 3 THEN
1023 1868 4 BEGIN
1024 1869 4 OLD_TIME = .CACHE_REF [.I];
1025 1870 4 OLD_NUM = .I;
1026 1871 3 END;
1027 1872 3
1028 1873 3 ASSERT (.OLD_TIME LSSU 999999999);
1029 1874 3
1030 1875 3 + Write the sucker out if it is dirty.
1031 1876 3 -
1032 1877 3
1033 1878 4 IF (.CACHE_DIRTY [.OLD_NUM] NEQ 0)
1034 1879 3 THEN
1035 1880 4 BEGIN
1036 1881 4 WRITE_WINDOW (.OLD_NUM);
1037 1882 3 END;
1038 1883 3
1039 1884 3 END
1040 1885 2 ELSE
1041 1886 2 OLD_NUM = NUM_WINDOW - 1 - .WINDOW_NUM;
1042 1887 2
1043 1888 2 +
1044 1889 2 Now, place the new bucket in the cache.
1045 1890 2 -
1046 1891 2 CACHE_TIMER = .CACHE_TIMER + 1;
1047 1892 2 CACHE_REF [.OLD_NUM] = .CACHE_TIMER;
1048 1893 2 CACHE_WINDOW [.OLD_NUM] = .WINDOW_NUM;
1049 1894 2 RETURN (.OLD_NUM);
1050 1895 1 END;
```

! of routine REPLACE_WINDOW

```
001C 00000 REPLACE_WINDOW:
50      04      AC      00000000' EF 9E 00002 .WORD Save R2,R3,R4 : 1806
53      00000064 8F      01 C3 00009 MOVAB CACHE_REF, R4 : 1852
      01F4      C4      0B C7 0000E SUBL3 #1, BOKT_NUM, R0 : 1854
      3B9AC9FF 8F      53 D1 00012 DIVL3 #11, R0, WINDOW_NUM : 1856
      63      05 19 00019 BLSS 1$ : 1862
      01F4      C4      01 D0 0001B MOVL #1, WF CACHE_FULL : 1866
      3B9AC9FF 8F      04 E9 00020 1$: BLBC WF CACHE_FULL, 5$ : 1869
      63      8F D0 00025 MOVL #999999999, OLD_TIME : 1870
      51      8F 9A 0002C MOVZBL #99, I : 1866
      6440 D1 00030 2$: CMPL CACHE_REF[I], OLD_TIME : 1873
      07 1E 00034 BGEQU 3$ :
      51      6440 D0 00036 MOVL CACHE_REF[I], OLD_TIME :
      52      50 D0 0003A MOVL I, OLD_NUM :
      F0      50 F4 0003D 3$: SOBGEQ I, 2$ :
      3B9AC9FF 8F      51 D1 00040 CMPL OLD_TIME, #999999999 :
      07 1F 00047 BLSSU 4$ :
```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
REPLACE_WINDOW - read a new window into the cac

D 14
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 33
(13)

EDT\$IOMOD
V04-000

00000000G 00	00	FB 00049	CALLS	#0, EDT\$\$INTER ERR	:	1878
	0190 C442	95 00050 4\$:	TSTB	CACHE_DIRTY[OLD_NUM]	:	
		11 13 00055	BEQL	6\$	:	
		52 DD 00057	PUSHL	OLD_NUM	:	1881
FF3C CF		01 FB 00059	CALLS	#1, WRITE_WINDOW	:	
		08 11 0005E	BRB	6\$	:	1856
52 00000063 8F		53 C3 00060 5\$:	SUBL3	WINDOW_NUM, #99, OLD_NUM	:	1886
	FE6C	C4 D6 00068 6\$:	INCL	CACHE_TIMER	:	1891
		C4 D0 0006C	MOVL	CACHE_TIMER, CACHE_REF[OLD_NUM]	:	1892
FE70 C442		53 D0 00072	MOVL	WINDOW_NUM, CACHE_WINDOW[OLD_NUM]	:	1893
50		52 D0 00078	MOVL	OLD_NUM, R0	:	1894
		04 0007B	RET		:	1895

; Routine Size: 124 bytes, Routine Base: _EDT\$CODE + 02C8

```
1052 1896 1 %SBTTL 'GET_WF - find a bucket in the cache, or read it in'
1053 1897 1 ROUTINE GET_WF (
1054 1898 1     BUKT_NUM,
1055 1899 1     BUF_ADDR,
1056 1900 1     READ
1057 1901 1 ) : NOVALUE =
1058 1902 1
1059 1903 1 ++
1060 1904 1 FUNCTIONAL DESCRIPTION:
1061 1905 1
1062 1906 1     This routine finds a bucket in the cache, or reads it in if necessary.
1063 1907 1
1064 1908 1 FORMAL PARAMETERS:
1065 1909 1
1066 1910 1     BUKT_NUM           The number of the bucket we are trying to locate.
1067 1911 1
1068 1912 1     BUF_ADDR          Address of a longword in which the bucket's address is returned.
1069 1913 1
1070 1914 1     READ              Flag indicating that we should read the file from disk if it
1071 1915 1                     is not in the cache.
1072 1916 1
1073 1917 1 IMPLICIT INPUTS:
1074 1918 1
1075 1919 1     CACHE_ADDR
1076 1920 1
1077 1921 1 IMPLICIT OUTPUTS:
1078 1922 1
1079 1923 1     IO_WF_RAB, if READ = 1
1080 1924 1     IO_STS and IO_STV, if READ = 1
1081 1925 1
1082 1926 1 ROUTINE VALUE:
1083 1927 1
1084 1928 1     NONE
1085 1929 1
1086 1930 1 SIDE EFFECTS:
1087 1931 1
1088 1932 1     Calls FIND_WINDOW and REPLACE_WINDOW.
1089 1933 1     Never returns on I/O errors.
1090 1934 1
1091 1935 1 --
1092 1936 1
1093 1937 2 BEGIN
1094 1938 2
1095 1939 2 LOCAL
1096 1940 2     ALLOC STATUS,
1097 1941 2     WINDOW_ADDR,
1098 1942 2     WINDOW_NUM;
1099 1943 2
1100 1944 2 ++
1101 1945 2 First, see if it is in the cache.
1102 1946 2 --
1103 1947 2
1104 1948 2 IF (FIND_WINDOW (.BUKT_NUM, .BUF_ADDR) GEQ 0) THEN RETURN;
1105 1949 2
1106 1950 2 ++
1107 1951 2 It's not, find a block to replace and read a block.
1108 1952 2 --
```

```

      .EXTRN  SYS$READ

```

				007C	00000	GET_WF: .WORD	Save R2,R3,R4,R5,R6	1897			
		56	00000000G	00	9E	00002	MOVAB	EDT\$\$INTER_ERR, R6			
		55	00000000'	EF	9E	00009	MOVAB	CACHE_ADDR, R5			
		5E		04	C2	00010	SUBL2	#4, SP			
		7E		AC	7D	00013	MOVQ	BUKT_NUM, -(SP)	1948		
		FE99	CF	02	FB	00017	CALLS	#2, FIND_WINDOW			
				50	D5	0001C	TSTL	R0			
				01	19	0001E	BLSS	1\$			
					04	00020	RET				
				04	AC	DD	00021	1\$: PUSHL	BUKT_NUM	1953	
		FF5B	CF	01	FB	00024	CALLS	#1, REPLACE_WINDOW			
			52	50	D0	00029	MOVL	R0, WINDOW_NUM			
			53	6542	D0	0002C	MOVL	CACHE_ADDR[WINDOW_NUM], WINDOW_ADDR	1954		
7E	FFFFFFFF	8F	04	01	7A	00030	EMUL	#1, BUKT_NUM, #-1, -(SP)	1970		
54		54		0B	7B	0003A	EDIV	#11, (SP)+, R4, R4			
		54		09	78	0003F	ASHL	#9, R4, R4			
				53	D5	00043	TSTL	WINDOW_ADDR	1956		
				3F	12	00045	BNEQ	5\$			
			35	0518	C5	E8	00047	BLBS	WF CACHE_FULL, 4\$		
				6542	DF	0004C	PUSHAL	CACHE_ADDR[WINDOW_NUM]	1959		
		04	AE	1600	8F	3C	0004F	MOVZWL	#5632, 4(SP)		
				04	AE	9F	00055	PUSHAB	4(SP)		
		00000000G	00	02	FB	00058	CALLS	#2, EDT\$\$ALO_HEAP			
			10	50	E8	0005F	BLBS	ALLOC STATUS, 2\$	1961		
		0518	C5	01	D0	00062	MOVL	#1, WF CACHE_FULL	1964		
				04	AC	DD	00067	PUSHL	BUKT_NUM	1965	
		FF15	CF	01	FB	0006A	CALLS	#1, REPLACE_WINDOW			
			52	50	D0	0006F	MOVL	R0, WINDOW_NUM			
			53	6542	D0	00072	2\$: MOVL	CACHE_ADDR[WINDOW_NUM], WINDOW_ADDR	1968		
				03	12	00076	BNEQ	3\$	1969		
			66	00	FB	00078	CALLS	#0, EDT\$\$INTER_ERR			
08	BC		53	54	C1	0007B	3\$: ADDL3	R4, WINDOW_ADDR, @BUF_ADDR	1970		
					04	00080	RET		1956		
				03	12	00081	4\$: BNEQ	5\$	1974		
			66	00	FB	00083	CALLS	#0, EDT\$\$INTER_ERR			
08	BC		53	54	C1	00086	5\$: ADDL3	R4, WINDOW_ADDR, @BUF_ADDR	1975		
			60	0C	AC	E9	0008B	BLBC	READ, 7\$	1980	
			05	051C	C5	E8	0008F	BLBS	WF_OPEN, 6\$	1984	
		FC97	CF	00	FB	00094	CALLS	#0, WF_OPN			
		09E0	C5	53	D0	00099	6\$: MOVL	WINDOW_ADDR, IO_WF_RAB+36	1986		
		09DC	C5	1600	8F	B0	0009E	MOVW	#5632, IO_WF_RAB+32	1987	
			04	AC	01	C3	000A5	SUBL3	#1, BUKT_NUM, R0	1988	
50			50	0B	C6	000AA	DIVL2	#11, R0			
			50	0B	C4	000AD	MULL2	#11, R0			
		09F4	C5	01	A0	9E	000B0	MOVAB	1(R0), IO_WF_RAB+56		
				09BC	C5	9F	000B6	PUSHAB	IO_WF_RAB	1990	
		00000000G	00	01	FB	000BA	CALLS	#1, SYS\$READ			
		0A04	C5	50	D0	000C1	MOVL	R0, IO_STS			
			26	50	E8	000C6	BLBS	R0, 7\$			
		0001827A	8F	0A04	C5	D1	000C9	CML	IO_STS, #98938	1993	
				1B	13	000D2	BEQL	7\$			
		0A08	C5	09C8	C5	D0	000D4	MOVL	IO_WF_RAB+12, IO_STV	1996	
			7E	0A04	C5	7D	000DB	MOVQ	IO_STS, -(SP)	1997	
				7E	D4	000E0	CLRL	-(SP)			
		00000000G	00	00000000G	8F	DD	000E2	PUSHL	#EDT\$ IOERRWRK		
					04	FB	000E8	CALLS	#4, LIB\$STOP		
					04	000EF	7\$: RET		2004		

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS

GET_WF - find a bucket in the cache, or read it

H 14
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1 Page 37
(14)

EDT\$
V04-

; Routine Size: 240 bytes, Routine Base: _EDT\$CODE + 0344

; R

; 16

: 2007

```
EDT$IOMOD - I/O for VAX/VMS
EDT$SWF_RD - get new current work file bucket
```

J 14
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDIT.SRC]IOMOD.B32;1

Page 39
(15)

EDTS
V04-

	5E		04	C2	00002
			01	DD	00C05
		04	AE	9F	00007
		04	AC	DD	0000A
F E F E	C F		03	F B	0000D
	50	08	A C	D0	00012
04	A0		6E	D0	00016
				04	0001A

```

SUBL2    #4, SP
PUSHL    #1
PUSHAB   BUF_ADDR
PUSHL    BUKT_NUM
CALLS    #3, GET_WF
MOVL     REC_DEST, RO
MOVL     BUF_ADDR, 4(RO)
RET

```

2053
2054
2055

```
; Routine Size: 27 bytes,    Routine Base: _EDT$CODE + 0434
```

; 1213 2056 1

[illegible]

```

1215      2057 1 %SBTTL 'EDT$$PAR_FNAME - parse a file name'
1216      2058 1
1217      2059 1 GLOBAL ROUTINE EDT$$PAR_FNAME (
1218      2060 1     RAB,
1219      2061 1     FILE_DESC,
1220      2062 1     RLF,
1221      2063 1     DSK,
1222      2064 1     STS,
1223      2065 1     STV
1224      2066 1 ) =
1225      2067 1
1226      2068 1
1227      2069 1 ++
1228      2070 1 FUNCTIONAL DESCRIPTION:
1229      2071 1     This routine parses a file name to determine if valid before
1230      2072 1     later attempting an open.
1231      2073 1
1232      2074 1 FORMAL PARAMETERS:
1233      2075 1
1234      2076 1     RAB           The address of the RAB to connect.
1235      2077 1
1236      2078 1     FILE_DESC      The address of a descriptor containing the filename
1237      2079 1
1238      2080 1     RLF           1 = use the input file as a related file name (EXIT rather than WRITE)
1239      2081 1
1240      2082 1     DSK           1 flags renameable file (disk or DECtape). 0 otherwise
1241      2083 1
1242      2084 1     STS           The RMS status returned
1243      2085 1
1244      2086 1     STV           The RMS stv word returned
1245      2087 1
1246      2088 1 IMPLICIT INPUTS:
1247      2089 1
1248      2090 1     NONE
1249      2091 1
1250      2092 1 IMPLICIT OUTPUTS:
1251      2093 1
1252      2094 1     IO_STS
1253      2095 1     IO_STV
1254      2096 1
1255      2097 1 ROUTINE VALUE:
1256      2098 1
1257      2099 1     If successful, 1; if failed, 0.
1258      2100 1
1259      2101 1 SIDE EFFECTS:
1260      2102 1
1261      2103 1     Passes back a '1' in DSK if this file can be renamed
1262      2104 1
1263      2105 1 --
1264      2106 1
1265      2107 2 BEGIN
1266      2108 2
1267      2109 2 MAP
1268      2110 2     FILE_DESC : REF BLOCK [, BYTE];
1269      2111 2
1270      2112 2 LOCAL
1271      2113 2     FNA,

```

```
1272 2114 2      RSA,  
1273 2115 2      RSL,  
1274 2116 2      FNS,  
1275 2117 2      WORK_FAB : $FAB DECL,  
1276 2118 2      RELAT_NAM : $NAM DECL,  
1277 2119 2      NAM_BLOCK : $NAM DECL;  
1278 2120 2  
1279 2121 2      FNA = .FILE_DESC [DSC$A_POINTER];      ! get file name  
1280 2122 2      FNS = .FILE_DESC [DSC$W_LENGTH];      ! and its length  
1281 2123 2  !+  
1282 2124 2  ! Initialize the NAM block.  
1283 2125 2  !-  
1284 2126 2      $NAM_INIT (NAM = NAM_BLOCK, ESA = MYBUF, ESS = MIN (255, NAM$C_MAXRSS));  
1285 2127 2  !+  
1286 2128 2  ! Set up the related name from the input file, if requested.  
1287 2129 2  !-  
1288 2130 2  
1289 2131 2      IF .RLF  
1290 2132 2      THEN  
1291 2133 3      BEGIN  
1292 2134 3          $NAM_INIT (NAM = RELAT_NAM, ESA = MYBUF2, ESS = NAM$C_MAXRSS, RSA = INPUT_NAME, RSS = NAM$C_MAXRSS);  
1293 2135 3          RELAT_NAM [NAM$B_RSL] = .INPUT_NAME_LEN;  
1294 2136 3          $$TRACE (EDT$$TR_OFO, EDT$$TR_SOFO_RLNAM, .RELAT_NAM [NAM$B_RSL], .RELAT_NAM [NAM$L_RSA]);  
1295 2137 3          NAM_BLOCK [NAM$L_RLF] = RELAT_NAM;  
1296 2138 2      END;  
1297 2139 2  
1298 2140 2  !+  
1299 2141 2  ! Setup the FAB  
1300 2142 2  !-  
1301 2143 2      $FAB_INIT (FAB = WORK_FAB, FOP = <OFP>, NAM = NAM_BLOCK);  
1302 2144 2      WORK_FAB [FAB$L_FNA] = .FNA;  
1303 2145 2      WORK_FAB [FAB$B_FNS] = .FNS;  
1304 2146 2  !+  
1305 2147 2  ! Attempt the parse, set up the status and return  
1306 2148 2  !-  
1307 2149 2      IO_STS = $RMS_PARSE (FAB = WORK_FAB);  
1308 2150 2      IO_STV = .WORK_FAB [FAB$L_STV];  
1309 2151 2  
1310 2152 2      IF .IO_STS  
1311 2153 2      THEN  
1312 2154 3      BEGIN  
1313 2155 3  !+  
1314 2156 3  ! Is this a renameable device?  
1315 2157 3  !-  
1316 2158 3  
1317 2159 5      IF (((.WORK_FAB [FAB$L_DEV] AND DEV$M_FOD) NEQ 0)      ! File oriented device  
1318 2160 5          AND ((.WORK_FAB [FAB$L_DEV] AND DEV$M_RND) NEQ 0)      ! Random access device  
1319 2161 5          AND ((.WORK_FAB [FAB$L_DEV] AND DEV$M_IDV) NEQ 0)      ! Open for input  
1320 2162 5          AND ((.WORK_FAB [FAB$L_DEV] AND DEV$M_SQD) EQL 0)      ! Not a magtape  
1321 2163 4          AND ((.NAM_BLOCK [NAM$C_FNB] AND NAM$M_PPF) EQL 0))      ! Not a process permanent file  
1322 2164 3      THEN  
1323 2165 4      BEGIN  
1324 2166 4  !+  
1325 2167 4  ! If the file type is .TMP we shouldn't do the rename, since that will  
1326 2168 4  ! cause the version number to be incremented by two, and won't help  
1327 2169 4  ! the problem of purging a partial file after a crash.  
1328 2170 4  !-
```

: Ro
: 17

MOVAB	IO STS, R9	:	
MOVAB	-276(SP), SP	:	
MOVL	FILE_DESC, R6	:	2121
MOVL	4(R6), FNA	:	
MOVZWL	(R6), FNS	:	2122
MOVC5	#0, (SP), #0, #96, \$RMS_PTR	:	2126
MOVW	#24578, \$RMS_PTR	:	
MNEGB	#1, \$RMS_PTR+10	:	
MOVAB	MYBUF, \$RMS_PTR+12	:	
BLBC	RLF, 1\$	:	2131
MOVC5	#0, (SP), #0, #96, \$RMS_PTR	:	2134
MOVW	#24578, \$RMS_PTR	:	

2135
2137
2143

2144
2145
2149

2150
2152
2159
2160
2161
2162
2163
2172
2176

2184
2159
2188
2190

2193
2194
2195

2200
2201
2202
2203

; 1362 2204 1

```
1364 2205 1 %SBTTL 'EDT$$OPN_IFIDEF - Open a file for input by name and default name'
1365 2206 1
1366 2207 1 GLOBAL ROUTINE EDT$$OPN_IFIDEF (
1367 2208 1     RAB,
1368 2209 1     FILE_DESC,
1369 2210 1     DNA,
1370 2211 1     DNS,
1371 2212 1     RLF,
1372 2213 1     STS,
1373 2214 1     STV,
1374 2215 1     VFC,
1375 2216 1     NONSTD
1376 2217 1 ) =
1377 2218 1
1378 2219 1 **
1379 2220 1 FUNCTIONAL DESCRIPTION:
1380 2221 1
1381 2222 1     This routine opens a file for input using a name string and default
1382 2223 1     name string.
1383 2224 1
1384 2225 1 FORMAL PARAMETERS:
1385 2226 1
1386 2227 1     RAB           The address of the RAB to connect.
1387 2228 1
1388 2229 1     FILE_DESC     The address of a descriptor containing the filename
1389 2230 1
1390 2231 1     DNA           The address of the default name string.
1391 2232 1
1392 2233 1     DNS           The length of the default name string.
1393 2234 1
1394 2235 1     RLF           1 = use the input file as a related file
1395 2236 1
1396 2237 1     STS           The RMS status returned
1397 2238 1
1398 2239 1     STV           The RMS stv word returned
1399 2240 1
1400 2241 1     VFC           returned as 1 = this file has sequence numbers, 0 = it doesn't
1401 2242 1
1402 2243 1     NONSTD        returned as 1 = this file is not a standard text file, 0 = it is
1403 2244 1
1404 2245 1 IMPLICIT INPUTS:
1405 2246 1
1406 2247 1     NONE
1407 2248 1
1408 2249 1 IMPLICIT OUTPUTS:
1409 2250 1
1410 2251 1     IO_STS
1411 2252 1     IO_STV
1412 2253 1
1413 2254 1 ROUTINE VALUE:
1414 2255 1
1415 2256 1     If successful, the IFI; if the open failed, 0.
1416 2257 1
1417 2258 1 SIDE EFFECTS:
1418 2259 1
1419 2260 1     NONE
1420 2261 1
```

```
1421 2262 1 !--
1422 2263 1
1423 2264 2 BEGIN
1424 2265 2
1425 2266 2 MAP
1426 2267 2 FILE_DESC : REF BLOCK [, BYTE],
1427 2268 2 RAB : REF $RAB_DECL.
1428 2269 2
1429 2270 2 LOCAL
1430 2271 2 FNA,
1431 2272 2 RSA,
1432 2273 2 RSL,
1433 2274 2 FNS,
1434 2275 2 XABPRO : $XABPRO_DECL,
1435 2276 2 WORK_FAB : $FAB_DECL,
1436 2277 2 RELAT_NAM : $NAM_DECL,
1437 2278 2 NAM_BLOCK : $NAM_DECL;
1438 2279 2
1439 2280 2 FNA = .FILE_DESC [DSC$A_POINTER]; ! get file name
1440 2281 2 FNS = .FILE_DESC [DSC$W_LENGTH]; ! and its length
1441 2282 2
1442 2283 2 + Initialize the NAM and XAB blocks.
1443 2284 2 -
1444 P 2285 2 $NAM_INIT (NAM = NAM_BLOCK, ESA = MYBUF2, ESS = MIN (255, NAM$C_MAXRSS), RSA = MYBUF, !
1445 2286 2 RSS = MIN (255, NAM$C_MAXRSS));
1446 2287 2 $XABPRO_INIT (XAB = XABPRO);
1447 2288 2
1448 2289 2 + Setup the FAB and RAB
1449 2290 2 -
1450 2291 2 $$TRACE (EDT$STR_IFO, EDT$STR_SIFO_FNAM, .FNS, .FNA);
1451 P 2292 2 $FAB_INIT (FAB = WORK_FAB, FAC = <GET>, RAT = CR, RFM = VAR, FOP = <SQO, OFP>, MRS = 255, SHR = GET,
1452 2293 2 NAM = NAM_BLOCK, XAB = XABPRO);
1453 2294 2 WORK_FAB [FAB$F_FNA] = .FNA;
1454 2295 2 WORK_FAB [FAB$F_FNS] = .FNS;
1455 P 2296 2 $RAB_INIT (RAB = .RAB, RAC = SEQ, ROP = <LOC, RAH>, UBF = MYBUF, USZ = 255, FAB = WORK_FAB, RHB = VFCHD,
1456 2297 2 CTX = 0);
1457 2298 2
1458 2299 2 + Move in the default name string.
1459 2300 2 -
1460 2301 2 $$TRACE (EDT$STR_IFO, EDT$STR_SIFO_DNAM, .DNS, .DNA);
1461 2302 2 WORK_FAB [FAB$F_DNA] = .DNA;
1462 2303 2 WORK_FAB [FAB$F_DNS] = .DNS;
1463 2304 2
1464 2305 2 + Set up the related name from the input file, if requested.
1465 2306 2 -
1466 2307 2
1467 2308 2 IF .RLF
1468 2309 2 THEN
1469 2310 2 BEGIN
1470 2311 2 $NAM_INIT (NAM = RELAT_NAM, ESA = MYBUF2, ESS = NAM$C_MAXRSS, RSA = INPUT_NAME, RSS = NAM$C_MAXRSS);
1471 2312 2 RELAT_NAM [NAM$B_RSL] = .INPUT_NAME_LEN;
1472 2313 2 $$TRACE (EDT$STR_OFO, EDT$STR_SOFO_RLNAM, .RELAT_NAM [NAM$B_RSL], .RELAT_NAM [NAM$B_RSA]);
1473 2314 2 NAM_BLOCK [NAM$B_RLF] = RELAT_NAM;
1474 2315 2 END;
1475 2316 2
1476 2317 2 +
1477 2318 2 ! Attempt the open and then the connect, setting up the status and STV
```

```
1478 2319 2  ! if either of them fails.
1479 2320 2  !-
1480 2321 2      IO STS = $RMS OPEN (FAB = WORK_FAB);
1481 2322 2      $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_OPN_STS, %UPVAL, WORK_FAB [FAB$$_STS]);
1482 2323 2      $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_OPN_STV, %UPVAL, WORK_FAB [FAB$$_STV]);
1483 2324 2
1484 2325 2      IF .IO_STS
1485 2326 2      THEN
1486 2327 3      BEGIN
1487 2328 3          RSL = .NAM_BLOCK [NAM$$_RSL];
1488 2329 3          STRING_DESC (.FILE_DESC, RSL, MYBUF);
1489 2330 3      END
1490 2331 2      ELSE
1491 2332 3      BEGIN
1492 2333 3
1493 2334 4          IF (.NAM_BLOCK [NAM$$_ESL] NEQ 0)
1494 2335 3          THEN
1495 2336 4          BEGIN
1496 2337 4              RSL = .NAM_BLOCK [NAM$$_ESL];
1497 2338 4              RSA = .NAM_BLOCK [NAM$$_ESA];
1498 2339 4              STRING_DESC (.FILE_DESC, RSL, .RSA);      ! get resultant filename
1499 2340 3          END;
1500 2341 3
1501 2342 2      END;
1502 2343 2
1503 2344 2  !+
1504 2345 2  ! If this is the first time, remember the input file name so we can
1505 2346 2  ! use it as a related name.
1506 2347 2  !-
1507 2348 2
1508 2349 3      IF (( NOT .ATTRIBUTES_SET) AND ( NOT .ATTRIBUTES_DEFAULTED))
1509 2350 2      THEN
1510 2351 3      BEGIN
1511 2352 3          INPUT_NAME_LEN = .FILE_DESC [DSC$$_LENGTH];
1512 2353 3          CH$MOVE (.INPUT_NAME_LEN, .FILE_DESC [DSC$$_POINTER], INPUT_NAME);
1513 2354 2      END;
1514 2355 2
1515 2356 2      IF .IO_STS
1516 2357 2      THEN
1517 2358 3      BEGIN
1518 2359 3          $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_OPN_IFI, 2, WORK_FAB [FAB$$_IFI]);
1519 2360 3          IO STS = $RMS CONNECT (RAB = .RAB);
1520 2361 3          $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_CON_STS, %UPVAL, RAB [FAB$$_STS]);
1521 2362 3          $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_CON_STV, %UPVAL, RAB [FAB$$_STV]);
1522 2363 3
1523 2364 3          IF .IO_STS
1524 2365 3          THEN
1525 2366 3  !+
1526 2367 3  ! Succeeded, return the IFI
1527 2368 3  !-
1528 2369 4          BEGIN
1529 2370 4              $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_RNAM, .NAM_BLOCK [NAM$$_RSL], .NAM_BLOCK [NAM$$_RSA]);
1530 2371 4              $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_CON_ISI, 2, RAB [RAB$$_ISI]);
1531 2372 4  !+
1532 2373 4  ! Get the record format, so we know if it has a VFC field or not.
1533 2374 4  ! If the PRN bit is set, the file is not considered a VFC file, since
1534 2375 4  ! the information in the record prefix is not a line number.
```

```
1535 2376 4 !-
1536 2377 4
1537 2378 5 IF ((.WORK_FAB [FAB$B_RFM] EQL FAB$C_VFC) AND ((.WORK_FAB [FAB$B_RAT] .ND FAB$M_PRN) EQL 0))
1538 2379 4 THEN
1539 2380 4 .VFC = 1 ! This file has sequence numbers
1540 2381 4 ELSE
1541 2382 4 .VFC = 0; ! This file does not have sequence numbers
1542 2383 4
1543 2384 4 !+
1544 2385 4 !- Check the file's attributes to see if it is a "standard" text file.
1545 2386 4 !-
1546 2387 4 .NONSTD = 0;
1547 2388 4
1548 2389 5 IF ((.WORK_FAB [FAB$B_RFM] NEQ FAB$C_VAR) AND (.WORK_FAB [FAB$B_RFM] NEQ FAB$C_VFC))
1549 2390 4 THEN
1550 2391 4 .NONSTD = 1;
1551 2392 4
1552 2393 4 IF (.WORK_FAB [FAB$B_RAT] NEQ FAB$M_CR) THEN .NONSTD = 1;
1553 2394 4
1554 2395 4 IF (.WORK_FAB [FAB$B_ORG] NEQ FAB$C_SEQ) THEN .NONSTD = 1;
1555 2396 4
1556 2397 4 !+
1557 2398 4 !- If this is the first input open, remember the file's attributes.
1558 2399 4 !-
1559 2400 4
1560 2401 5 IF (( NOT .ATTRIBUTES_SET) AND ( NOT .ATTRIBUTES_DEFAULTED))
1561 2402 4 THEN
1562 2403 5 BEGIN
1563 2404 5 ATTRIBUTES_SET = 1;
1564 2405 5 INPUT_PROT = .XABPRO [XAB$W_PRO];
1565 2406 5 !+
1566 2407 5 !- If PRN format, make the output CR format.
1567 2408 5 !-
1568 2409 5
1569 2410 6 IF ((.WORK_FAB [FAB$B_RAT] AND FAB$M_PRN) NEQ 0)
1570 2411 5 THEN
1571 2412 6 BEGIN
1572 2413 6 INPUT_RAT = (.WORK_FAB [FAB$B_RAT] AND NOT FAB$M_PRN) OR FAB$M_CR;
1573 2414 6 INPUT_RFM = FAB$C_VAR;
1574 2415 6 END
1575 2416 5 ELSE
1576 2417 6 BEGIN
1577 2418 6 INPUT_RAT = .WORK_FAB [FAB$B_RAT];
1578 2419 6 INPUT_RFM = .WORK_FAB [FAB$B_RFM];
1579 2420 5 END;
1580 2421 5
1581 2422 5 INPUT_MRS = MIN (.WORK_FAB [FAB$W_MRS], 255);
1582 2423 5
1583 2424 5 IF (.INPUT_MRS EQL 0) THEN INPUT_MRS = 255;
1584 2425 5
1585 2426 4 END;
1586 2427 4
1587 2428 4 .STV = .IO_STV;
1588 2429 4 .STS = .IO_STS;
1589 2430 4 RETURN (.WORK_FAB [FAB$W_IF1]);
1590 2431 4 END
1591 2432 3 ELSE
```

```

: 1592      2433      3      IO_STV = .RAB [RAB$L_STV]
: 1593      2434      3
: 1594      2435      3      END
: 1595      2436      2      ELSE
: 1596      2437      2      IO_STV = .WORK_FAB [FAB$L_STV];
: 1597      2438      2
: 1598      2439      2      !+
: 1599      2440      2      ! Failed, return 0 and set input attributes = default attributes
: 1600      2441      2      !-
: 1601      2442      2      ATTRIBUTES_DEFAULTED = 1;
: 1602      2443      2      .STV = .IO_STV;
: 1603      2444      2      .STS = .IO_STS;
: 1604      2445      2      RETURN (0);
: 1605      2446      1      END;
```

! of EDT\$\$OPN_IFIDEF

.EXTRN SYS\$OPEN

.ENTRY EDT\$\$OPN_IFIDEF, Save R2,R3,R4,R5,R6,R7,R8,-; 2207
R9

```
MOVAB IO_STS, R9
MOVAB -384(SP), SP
MOVL FILE_DESC, R7
MOVL 4(R7), FNA
MOVZWL (R7), FNS
MOVCS #0, (SP), #0, #96, $RMS_PTR
```

```
MOVW #24578, $RMS_PTR
MNEGB #1, $RMS_PTR+2
MOVAB MYBUF, $RMS_PTR+4
MNEGB #1, $RMS_PTR+10
MOVAB MYBUF2, $RMS_PTR+12
MOVCS #0, (SP), #0, #88, $RMS_PTR
```

```
MOVW #22547, $RMS_PTR
MOVCS #0, (SP), #0, #80, $RMS_PTR
```

```
MOVW #20483, $RMS_PTR
MOVL #536870976, $RMS_PTR+4
MOVW #514, $RMS_PTR+22
MOVW #514, $RMS_PTR+30
MOVAB XABPRO, $RMS_PTR+36
MOVAB NAM_BLOCK, $RMS_PTR+40
MOVZBW #255, $RMS_PTR+54
MOVL FNA, WORK_FAB+44
MOVB FNS, WORK_FAB+52
MOVL RAB, R6
MOVCS #0, (SP), #0, #68, (R6)
```

```
MOVW #17409, (R6)
MOVL #66048, 4(R6)
CLRB 30(R6)
MOVZBW #255, 32(R6)
MOVAB MYBUF, 36(R6)
MOVAB VFCHD, 44(R6)
MOVAB WORK_FAB, 60(R6)
```

03FC 00000

```

59 00000000' EF 9E 00002
5E FE94 CE 9E 00009
57 08 AC D0 0000E
58 04 A7 D0 00012
56 67 3C 00016
6E 00 2C 00019
04 04 AE 00020
04 AE 6002 8F B0 00022
06 AE 01 8E 00028
08 AE FB1C C9 9E 0002C
0E AE 01 8E 00032
10 AE FC1C C9 9E 00036
6E 00 2C 0003C
A8 AD 00043
A8 AD 5813 8F B0 00045
6E 00 2C 0004B
FF58 CD 00052
FF58 CD 5003 8F B0 00055
FF5C CD 20000040 8F D0 0005C
FF6E CD 0202 8F B0 00065
FF76 CD 0202 8F B0 0006C
FF7C CD A8 AD 9E 00073
80 AD 04 AE 9E 00079
8E AD FF 8F 9B 0007E
84 AD 58 D0 00083
8C AD 56 90 00087
56 04 AC D0 0008B
6E 00 2C 0008F
66 66 00096
04 66 4401 8F B0 00097
A6 00010200 8F D0 0009C
1E A6 94 000A4
20 A6 FF 8F 9B 000A7
24 A6 FB1C C9 9E 000AC
2C A6 08 A9 9E 000B2
3C A6 FF58 CD 9E 000B7
```

0060	8F	00	88	AD	0C	AC	D0	000BD	MOVL	DNA, WORK_FAB+48	2302
			8D	AD	10	AC	90	000C2	MOVB	DNS, WORK_FAB+53	2303
				2E	14	AC	E9	000C7	BLBC	RLF, 1\$	2308
				6E		00	2C	000CB	MOVCS	#0, (SP), #0, #96, \$RMS_PTR	2311
					64	AE		000D2			
			64	AE	6002	8F	80	000D4	MOVW	#24578, \$RMS_PTR	
			66	AE		01	8E	000DA	MNEGB	#1, \$RMS_PTR+2	
			68	AE	FD1C	C9	9E	000DE	MOVAB	INPUT_NAME, \$RMS_PTR+4	
			6E	AE		01	8E	000E4	MNEGB	#1, \$RMS_PTR+10	
			70	AE	FC1C	C9	9E	000E8	MOVAB	MYBUF2, \$RMS_PTR+12	
			67	AE	FE1C	C9	90	000EE	MOVB	INPUT_NAME_LEN, RELAT_NAM+3	2312
			14	AE	64	AE	9E	000F4	MOVAB	RELAT_NAM, NAM_BLOCK+T6	2314
					FF58	CD	9F	000F9	PUSHAB	WORK_FAB	2321
		00000000G	00			01	FB	000FD	CALLS	#1, SYSSOPEN	
			69			50	D0	00104	MOVL	R0, IO_STS	
			0A			69	E9	00107	BLBC	IO_STS, 2\$	2325
			6E		07	AE	9A	0010A	MOVZBL	NAM_BLOCK+3, RSL	2328
					FB1C	C9	9F	0010E	PUSHAB	MYBUF	2329
						0F	11	00112	BRB	3\$	
					0F	AE	95	00114	TSTB	NAM_BLOCK+11	2334
						16	13	00117	BEQL	4\$	
			6E		0F	AE	9A	00119	MOVZBL	NAM_BLOCK+11, RSL	2337
			50		10	AE	D0	0011D	MOVL	NAM_BLOCK+12, RSA	2338
						50	DD	00121	PUSHL	RSA	2339
					04	AE	9F	00123	PUSHAB	RSL	
						57	DD	00126	PUSHL	R7	
		00000000G	00			03	FB	00128	CALLS	#3, STR\$COPY R	
			13		FE30	C9	E8	0012F	BLBS	ATTRIBUTES_SET, 5\$	2349
			0E		FE34	C9	E8	00134	BLBS	ATTRIBUTES_DEFAULTED, 5\$	
			FE1C			67	3C	00139	MOVZWL	(R7), INPUT_NAME_LEN	2352
FD1C	C9		04		FE1C	C9	28	0013E	MOVCS	INPUT_NAME_LEN, 54(R7), INPUT_NAME	2353
			03			69	E8	00147	BLBS	IO_STS, 6\$	2356
						00B4	31	0014A	BRW	18\$	
						56	DD	0014D	PUSHL	R6	2360
		00000000G	00			01	FB	0014F	CALLS	#1, SYSSCONNECT	
			69			50	D0	00156	MOVL	R0, IO_STS	
			03			69	E8	00159	BLBS	IO_STS, 7\$	2364
						009B	31	0015C	BRW	17\$	
			51		FF77	CD	9A	0015F	MOVZBL	WORK_FAB+31, R1	2378
			03			51	91	00164	CMPB	R1, #3	
						0C	12	00167	BNEQ	8\$	
						02	E0	00169	BBS	#2, WORK_FAB+30, 8\$	
06	FF76		CD			01	D0	0016F	MOVL	#1, @VFC	2380
	20		BC			03	11	00173	BRB	9\$	
					20	BC	D4	00175	CLRL	@VFC	2382
					24	BC	D4	00178	CLRL	@NONSTD	2387
						51	91	0017B	CMPB	R1, #2	2389
			02			09	13	0017E	BEQL	10\$	
			03			51	91	00180	CMPB	R1, #3	
						04	13	00183	BEQL	10\$	
	24		BC			01	D0	00185	MOVL	#1, @NONSTD	2391
			50		FF76	CD	9A	00189	MOVZBL	WORK_FAB+30, R0	2393
			02			50	91	0018E	CMPB	R0, #2	
						04	13	00191	BEQL	11\$	
	24		BC			01	D0	00193	MOVL	#1, @NONSTD	
					FF75	CD	95	00197	TSTB	WORK_FAB+29	2395
						04	13	0019B	BEQL	12\$	

24	BC	01	D0	0019D	MOVL	#1, @NONSTD	:	
45	FE30	C9	E8	001A1	12\$:	BLBS	ATTRIBUTES_SET, 16\$	: 2401
40	FE34	C9	E8	001A6		BLBS	ATTRIBUTES_DEFAULTED, 16\$	:
FE30	C9	01	D0	001AB		MOVL	#1, ATTRIBUTES_SET	: 2404
FE28	C9	B0	AD	3C 001B0		MOVZWL	XABPRO+8, INPUT_PROT	: 2405
50		02	E1	001B6		BBC	#2, R0, 13\$	: 2410
50		04	8A	001BA		BICB2	#4, R0	: 2413
50		02	C9	001BD		BISL3	#2, R0, INPUT_RAT	:
FE24	C9	02	D0	001C3		MOVL	#2, INPUT_RFM	: 2414
		05	11	001C8		BRB	14\$	: 2410
FE20	C9	50	7D	001CA	13\$:	MOVQ	R0, INPUT_RAT	: 2418
50		8E	AD	3C 001CF	14\$:	MOVZWL	WORK_FAB+54, R0	: 2422
00FF	8F	50	B1	001D3		CMPW	R0, #255	:
		04	1B	001D8		BLEQU	15\$	:
50		8F	9A	001DA		MOVZBL	#255, R0	:
FE2C	C9	50	D0	001DE	15\$:	MOVL	R0, INPUT_MRS	:
		06	12	001E3		BNEQ	16\$	: 2424
FE2C	C9	FF	8F	9A 001E5		MOVZBL	#255, INPUT_MRS	:
1C	BC	04	A9	D0 001EB	16\$:	MOVL	IO_STV, @STV	: 2428
18	BC	69	D0	001F0		MOVL	IO_STS, @STS	: 2429
50		FF5A	CD	3C 001F4		MOVZWL	WORK_FAB+2, R0	: 2430
			04	001F9		RET		:
04	A9	0C	A6	D0 001FA	17\$:	MOVL	12(R6), IO_STV	: 2433
			06	11 001FF		BRB	19\$	: 2364
04	A9	FF64	CD	D0 00201	18\$:	MOVL	WORK_FAB+12, IO_STV	: 2437
FE34	C9		01	D0 00207	19\$:	MOVL	#1, ATTRIBUTES_DEFAULTED	: 2442
1C	BC	04	A9	D0 0020C		MOVL	IO_STV, @STV	: 2443
18	BC		69	D0 00211		MOVL	IO_STS, @STS	: 2444
			50	D4 00215		CLRL	R0	: 2445
			04	00217		RET		: 2446

; Routine Size: 536 bytes, Routine Base: _EDT\$CODE + 0551

; 1606 2447 1

; R

; 20

```

1608 2448 1 %SBTTL 'EDT$$OPN_INOUT - Open a file for input and output by name and default name'
1609 2449 1
1610 2450 1 GLOBAL ROUTINE EDT$$OPN_INOUT (
1611 2451 1     RAB,
1612 2452 1     FILE_DESC,
1613 2453 1     DNA,
1614 2454 1     DNS,
1615 2455 1     STS,
1616 2456 1     STV
1617 2457 1 ) =
1618 2458 1
1619 2459 1 ++
1620 2460 1 FUNCTIONAL DESCRIPTION:
1621 2461 1
1622 2462 1     This routine opens a file for input and output using a name string and default
1623 2463 1     name string. It is used only for the journal file.
1624 2464 1
1625 2465 1 FORMAL PARAMETERS:
1626 2466 1
1627 2467 1     RAB           The address of the RAB to connect.
1628 2468 1
1629 2469 1     FILE_DESC     The address of a descriptor containing the filename
1630 2470 1
1631 2471 1     DNA           The address of the default name string.
1632 2472 1
1633 2473 1     DNS           The length of the default name string.
1634 2474 1
1635 2475 1     STS           The RMS status returned
1636 2476 1
1637 2477 1     STV           The RMS stv word returned
1638 2478 1
1639 2479 1 IMPLICIT INPUTS:
1640 2480 1
1641 2481 1     NONE
1642 2482 1
1643 2483 1 IMPLICIT OUTPUTS:
1644 2484 1
1645 2485 1     IO_STS
1646 2486 1     IO_STV
1647 2487 1
1648 2488 1 ROUTINE VALUE:
1649 2489 1
1650 2490 1     If successful, the IFI; if the open failed, 0.
1651 2491 1
1652 2492 1 SIDE EFFECTS:
1653 2493 1
1654 2494 1     NONE
1655 2495 1
1656 2496 1 --
1657 2497 1
1658 2498 2 BEGIN
1659 2499 2
1660 2500 2 MAP
1661 2501 2     FILE_DESC : REF BLOCK [, BYTE],
1662 2502 2     RAB : REF $RAB_DECL;
1663 2503 2
1664 2504 2 LOCAL
```

```

1665 2505 2      FNA,
1666 2506 2      RSA,
1667 2507 2      RSL,
1668 2508 2      FNS,
1669 2509 2      XABPRO : $XABPRO DECL,
1670 2510 2      WORK_FAB : $FAB DECL,
1671 2511 2      RELAT_NAM : $NAM_DECL;
1672 2512 2
1673 2513 2      FNA = .FILE_DESC [DSC$A_POINTER];      ! get file name
1674 2514 2      FNS = .FILE_DESC [DSC$W_LENGTH];      ! and its length
1675 2515 2
1676 2516 2      + Initialize the NAM and XAB blocks.
1677 2517 2      Since only the journal file is opened for I/O, we save the NAM block
1678 2518 2      in case we later delete the journal file.
1679 2519 2
1680 2520 2      $NAM_INIT (NAM = JOURNAL_NAM, ESA = MYBUF2, ESS = NAM$C_MAXRSS, RSA = MYBUF, RSS = NAM$C_MAXRSS);
1681 2521 2      $NAM_INIT (NAM = RELAT_NAM, RSA = INPUT_NAME, RSS = NAM$C_MAXRSS);
1682 2522 2      RELAT_NAM [NAM$B_RSL] = .INPUT_NAME_LEN;
1683 2523 2      $XABPRO_INIT (XAB = XABPRO, PRO = <RD, RWD>);
1684 2524 2
1685 2525 2      + Setup the FAB and RAB
1686 2526 2
1687 2527 2      $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_FNAM, .FNS, .FNA);
1688 P 2528 2      $FAB_INIT (FAB = WORK_FAB, ALQ = 1, DEQ = 5, FAC = <GET, PUT>, FOP = <SQO, DFW, OFP>, SHR = NIL,
1689 2529 2      RAT = CR, RFM = VAR, MRS = 255, NAM = JOURNAL_NAM, XAB = XABPRO);
1690 2530 2      WORK_FAB [FAB$B_FNA] = .FNA;
1691 2531 2      WORK_FAB [FAB$B_FNS] = .FNS;
1692 P 2532 2      $RAB_INIT (RAB = .RAB, RAC = SEQ, ROP = <LOC, RAH, WBH>, UBF = MYBUF, USZ = 255, FAB = WORK_FAB,
1693 2533 2      RHB = VFCHD, CTX = 0);
1694 2534 2
1695 2535 2      + Move in the default name string.
1696 2536 2
1697 2537 2      $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_DNAM, .DNS, .DNA);
1698 2538 2      WORK_FAB [FAB$B_DNA] = .DNA;
1699 2539 2      WORK_FAB [FAB$B_DNS] = .DNS;
1700 2540 2      $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_RLNAM, .RELAT_NAM [NAM$B_RSL], .RELAT_NAM [NAM$B_RSA]);
1701 2541 2      JOURNAL_NAM [NAM$B_RLF] = RELAT_NAM;
1702 2542 2
1703 2543 2      + Attempt the open and then the connect, setting up the status and STV
1704 2544 2      if either of them fails.
1705 2545 2
1706 2546 2      IO_STS = $RMS_OPEN (FAB = WORK_FAB);
1707 2547 2      JOURNAL_NAM [NAM$B_RLF] = 0;
1708 2548 2      $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_OPN_STS, %UPVAL, WORK_FAB [FAB$B_STS]);
1709 2549 2      $$TRACE (EDT$$STR_IFO, EDT$$STR_SIFO_OPN_STV, %UPVAL, WORK_FAB [FAB$B_STV]);
1710 2550 2
1711 2551 2      IF .IO_STS
1712 2552 2      THEN
1713 2553 3      BEGIN
1714 2554 3      RSL = .JOURNAL_NAM [NAM$B_RSL];
1715 2555 3      STRING_DESC (.FILE_DESC, RSL, MYBUF);
1716 2556 3      END
1717 2557 2      ELSE
1718 2558 3      BEGIN
1719 2559 3
1720 2560 4      IF (.JOURNAL_NAM [NAM$B_ESL] NEQ 0)
1721 2561 3      THEN
```

: 21

.ENTRY	EDT\$\$\$OPN_INOUT, Save R2,R3,R4,R5,R6,R7,R8,-	: 2450
	R9	:
MOVAB	10 STS, R9	:
MOVAB	-288(SP), SP	:
MOVL	FILE_DESC, R7	: 2513
MOVL	4(R7), FNA	:
MOVZWL	(R7), FNS	: 2514
MOVC5	#0, (SP), #0, #96, \$RMS_PTR	: 2520
		:
MOVW	#24578, \$RMS_PTR	:
MNEGB	#1, \$RMS_PTR+2	:

0060	8F	00	FE3C C9 FB1C C9 9E 0002F	MOVAB MYBUF, \$RMS_PTR+4	
			FE42 C9 01 8E 00036	MNEGB #1, \$RMS_PTR+10	
			FE44 C9 FC1C C9 9E 0003B	MOVAB MYBUF2, \$RMS_PTR+12	
			6E 00 2C 00042	MOVCS #0, (SP), #0, #96, \$RMS_PTR	2521
			04 AE 00049		
			04 AE 6002 8F B0 0004B	MOVW #24578, \$RMS_PTR	
			06 AE 01 8E 00051	MNEGB #1, \$RMS_PTR+2	
0058	8F	00	08 AE FD1C C9 9E 00055	MOVAB INPUT_NAME, \$RMS_PTR+4	
			07 AE FE1C C9 90 0005B	MOVW INPUT_NAME_LEN, RELAT_NAM+3	2522
			6E 00 2C 00061	MOVCS #0, (SP), #0, #88, \$RMS_PTR	2523
			A8 AD 00068		
0050	8F	00	A8 AD 5813 8F B0 0006A	MOVW #22547, \$RMS_PTR	
			B0 AD FF46 8F B0 00070	MOVW #-186, \$RMS_PTR+8	
			6E 00 2C 00076	MOVCS #0, (SP), #0, #80, \$RMS_PTR	2529
			64 AE 0007D		
			64 AE 5003 8F B0 0007F	MOVW #20483, \$RMS_PTR	
			68 AE 20000060 8F D0 00085	MOVL #536871008, \$RMS_PTR+4	
			74 AE 01 D0 0008D	MOVL #1, \$RMS_PTR+16	
			78 AE 20030005 8F D0 00091	MOVL #537067525, \$RMS_PTR+20	
			0082 CE 0202 8F B0 00099	MOVW #514, \$RMS_PTR+30	
			FF7C CD A8 AD 9E 000A0	MOVAB XABPRO, \$RMS_PTR+36	
			80 AD FE38 C9 9E 000A6	MOVAB JOURNAL_NAM, \$RMS_PTR+40	
			8E AD FF 8F 9B 000AC	MOVZBW #255, \$RMS_PTR+54	
			84 AD 58 D0 000B1	MOVL FNA, WORK_FAB+44	2530
			8C AD 56 90 000B5	MOVW FNS, WORK_FAB+52	2531
0044	8F	00	56 AC D0 000B9	MOVL RAB, R6	2533
			6E 00 2C 000BD	MOVCS #0, (SP), #0, #68, (R6)	
			66 000C4		
			04 66 4401 8F B0 000C5	MOVW #17409, (R6)	
			A6 00010600 8F D0 000CA	MOVL #67072, 4(R6)	
			1E A6 94 000D2	CLRB 30(R6)	
			20 A6 FF 8F 9B 000D5	MOVZBW #255, 32(R6)	
			24 A6 FB1C C9 9E 000DA	MOVAB MYBUF, 36(R6)	
			2C A6 08 A9 9E 000E0	MOVAB VFCHD, 44(R6)	
			3C A6 64 AE 9E 000E5	MOVAB WORK_FAB, 60(R6)	
			88 AD 0C AC D0 000EA	MOVL DNA, WORK_FAB+48	2538
			8D AD 10 AC 90 000EF	MOVW DNS, WORK_FAB+53	2539
			FE48 C9 04 AE 9E 000F4	MOVAB RELAT_NAM, JOURNAL_NAM+16	2541
			64 AE 9F 000FA	PUSHAB WORK_FAB	2546
00000000G		00	01 FB 000FD	CALLS #1, SYS\$OPEN	
		69	50 D0 00104	MOVL R0, IO_STS	
			FE48 C9 D4 00107	CLRL JOURNAL_NAM+16	2547
		0B	69 E9 0010B	BLBC IO_STS, 1\$	2551
		6E	FE3B C9 9A 0010E	MOVZBL JOURNAL_NAM+3, RSL	2554
			FB1C C9 9F 00113	PUSHAB MYBUF	2555
			11 11 00117	BRB 2\$	
		50	FE43 C9 9A 00119 1\$:	MOVZBL JOURNAL_NAM+11, R0	2560
		16	13 0011E	BEQL 3\$	
		6E	50 D0 00120	MOVL R0, RSL	2563
		50	FE44 C9 D0 00123	MOVL JOURNAL_NAM+12, RSA	2564
			50 DD 00128	PUSHL RSA	2565
			04 AE 9F 0012A 2\$:	PUSHAB RSL	
			57 DD 0012D	PUSHL R7	
00000000G		00	03 FB 0012F	CALLS #3, STR\$COPY_R	
		24	69 E9 00136 3\$:	BLBC IO_STS, 5\$	2570
			56 DD 00139	PUSHL R6	2574
00000000G		00	01 FB 0013B	CALLS #1, SYS\$CONNECT	

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$OPN_INOUT - Open a file for input and out

M 15
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 55
(18)

	69		50	D0	00142	MOVL	R0, IO_STS	:	
	0E		69	E9	00145	BLBC	IO_STS, 4\$	:	2578
18	BC	04	A9	D0	00148	MOVL	IO_STV, @STV	:	2584
14	BC		69	D0	0014D	MOVL	IO_STS, @STS	:	2585
	50	66	AE	3C	00151	MOVZWL	WORK_FAB+2, R0	:	2586
				04	00155	RET		:	
04	A9	0C	A6	D0	00156 4\$:	MOVL	12(R6), IO_STV	:	2589
			05	11	0015B	BRB	6\$	:	2578
04	A9	70	AE	D0	0015D 5\$:	MOVL	WORK_FAB+12, IO_STV	:	2593
18	BC	04	A9	D0	00162 6\$:	MOVL	IO_STV, @STV	:	2598
14	BC		69	D0	00167	MOVL	IO_STS, @STS	:	2599
			50	D4	0016B	CLRL	R0	:	2600
			04	0016D	RET			:	2601

; Routine Size: 366 bytes, Routine Base: _EDT\$CODE + 0769

; 1762 2602 1

EDT1
V04-

2578
2584
2585
2586
2589
2578
2593
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047
4048
4049
4050
4051
4052
4053
4054
4055
4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077
4078
4079
4080
4081
4082
4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093
4094
4095
4096
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117
4118
4119
4120
4121
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133
4134
4135
4136
4137
4138
4139
4140
4141
4142
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152
4153
4154
4155
4156
4157
4158
4159
4160
4161
4162
4163
4164
4165
4166
4167
4168
4169
4170
4171
4172
4173
4174
4175
4176
4177
4178
4179
4180
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193
4194
4195
4196
4197
4198
4199
4200
4201
4202
4203
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223
4224
4225
4226
4227
4228
4229
4230
4231
4232
4233
4234
4235
4236
4237
4238
4239
4240
4241
4242
4243
4244
4245
4246
4247
4248
4249
4250
4251
4252
4253
4254
4255
4256
4257
4258
4259
4260
4261
4262
4263
4264
4265
4266
4267
4268
4269
4270
4271
4272
4273
4274
4275
4276
4277
4278
4279
4280
4281
4282
4283
4284
4285
4286
4287
4288
4289
4290
4291
4292
4293
4294
4295
4296
4297
4298
4299
4300
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312
4313
4314
4315
4316
4317
4318
4319
4320
4321
4322
4323
4324
4325
4326
4327
4328
4329
4330
4331
4332
4333
4334
4335
4336
4337
4338
4339
4340
4341
4342
4343
4344
4345
4346
4347
4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360
4361
4362
4363
4364
4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417

```
: 1764 2603 1 %SBTTL 'EDT$$OPN_OFIDEF - open a file for output by name and default name'
: 1765 2604 1
: 1766 2605 1 GLOBAL ROUTINE EDT$$OPN_OFIDEF (
: 1767 2606 1     RAB,                               | Open a file for output by name and default name
: 1768 2607 1     FILE_DESC,                   | Address of the RAB
: 1769 2608 1     DNA,                         | descriptor of filename
: 1770 2609 1     DNS,                         | Address of the default name string
: 1771 2610 1     SEQ,                         | Length of the default name string
: 1772 2611 1     RLF,                         | 1 = VFC format
: 1773 2612 1     DFA,                         | 1 = use input file as related name
: 1774 2613 1     FORCE_MAXV,                   | 1 = use default file attributes, 2 = journal attributes
: 1775 2614 1     STS,                         | 1 = force maximum version number
: 1776 2615 1     STV,                         | rms status word
: 1777 2616 1     ) =                          | rms stv word
: 1778 2617 1
: 1779 2618 1 ++
: 1780 2619 1 FUNCTIONAL DESCRIPTION:
: 1781 2620 1
: 1782 2621 1     This routine opens a file for output using a name string and default
: 1783 2622 1     name string.
: 1784 2623 1
: 1785 2624 1 FORMAL PARAMETERS:
: 1786 2625 1
: 1787 2626 1     RAB                          The address of the RAB to connect.
: 1788 2627 1
: 1789 2628 1     FILE_DESC                     The address of a descriptor containing filename
: 1790 2629 1
: 1791 2630 1     DNA                          The address of the default file name string.
: 1792 2631 1
: 1793 2632 1     DNS                          The length of the default file name string.
: 1794 2633 1
: 1795 2634 1     SEQ                          Flag indicating that the file should be created with VFC format.
: 1796 2635 1
: 1797 2636 1     RLF                          Flag indicating that open should use primary input file as related file
: 1798 2637 1
: 1799 2638 1     DFA                          Flag indicating that the open should use input, journal or default file attributes
: 1800 2639 1
: 1801 2640 1     FORCE_MAXV                     Force maximum version number.
: 1802 2641 1
: 1803 2642 1     STS                          RMS status returned
: 1804 2643 1
: 1805 2644 1     STV                          RMS STV word returned
: 1806 2645 1
: 1807 2646 1 IMPLICIT INPUTS:
: 1808 2647 1
: 1809 2648 1     NONE
: 1810 2649 1
: 1811 2650 1 IMPLICIT OUTPUTS:
: 1812 2651 1
: 1813 2652 1     IO_STS
: 1814 2653 1     IO_STV
: 1815 2654 1
: 1816 2655 1 ROUTINE VALUE:
: 1817 2656 1
: 1818 2657 1     If the file is opened successfully, the IF1; otherwise, 0.
: 1819 2658 1
: 1820 2659 1 SIDE EFFECTS:
```

```
1821 2660 1 |
1822 2661 1 |      NONE
1823 2662 1 |
1824 2663 1 |  --
1825 2664 1 |
1826 2665 2 |      BEGIN
1827 2666 2 |
1828 2667 2 |      MAP
1829 2668 2 |          FILE_DESC : REF BLOCK [, BYTE],
1830 2669 2 |          RAB : REF $RAB_DECL;
1831 2670 2 |
1832 2671 2 |      LOCAL
1833 2672 2 |          RSA,
1834 2673 2 |          RSL,
1835 2674 2 |          FNA,
1836 2675 2 |          FNS,
1837 2676 2 |          XABPRO : $XABPRO_DECL,
1838 2677 2 |          WORK_FAB : $FAB_DECL,
1839 2678 2 |          NAM_BLOCK_PTR : REF $NAM_DECL,
1840 2679 2 |          RELAT_NAM : $NAM_DECL;
1841 2680 2 |
1842 2681 2 |          FNA = .FILE_DESC [DSC$A_POINTER];
1843 2682 2 |          FNS = .FILE_DESC [DSC$W_LENGTH];
1844 2683 2 |  +
1845 2684 2 |  Setup the FAB, RAB and XAB for the open.
1846 2685 2 |  --
1847 2686 2 |      $$TRACE (EDT$STR_OF0, EDT$STR_SOFO_FNAM, .FNS, .FNA);
1848 2687 2 |      $FAB_INIT ( FAB = WORK_FAB
P      2688 2 |                  ,ALQ = 1
P      2689 2 |                  ,DEQ = 0
P      2690 2 |                  ,FAC = <PUT>
P      2691 2 |                  ,FOP = <SQO, DFW, OFP>
1853 2692 2 |                  ,SHR = NIL);
1854 2693 2 |  +
1855 2694 2 |  Maximize version number if requested.
1856 2695 2 |  --
1857 2696 2 |
1858 2697 2 |      IF .FORCE_MAXV THEN WORK_FAB [FAB$L_FOP] = .WORK_FAB [FAB$L_FOP] OR FAB$M_MXV;
1859 2698 2 |
1860 2699 2 |      CASE .DFA FROM 0 TO 2 OF
1861 2700 2 |          SET
1862 2701 2 |
1863 2702 2 |          [0] :                               ! EXIT
1864 2703 3 |              BEGIN
1865 2704 3 |
1866 2705 3 |              IF .ATTRIBUTES_SET
1867 2706 3 |              THEN
1868 2707 4 |                  BEGIN
1869 2708 4 |  +
1870 2709 4 |  Use file attributes from the primary input file.
1871 2710 4 |  --
1872 2711 4 |          $XABPRO_INIT (XAB = XABPRO);
1873 2712 4 |          XABPRO [XAB$W_PRO] = .INPUT_PROT;
1874 2713 4 |          WORK_FAB [FAB$L_XAB] = XABPRO;
1875 2714 4 |  +
1876 2715 4 |  If the input file was VFC sequenced, then a /SEQ on EXIT means give me
1877 2716 4 |  a VFC file with the same attributes as the input file. The
```

```

1878 2717 4 ! absence of /SEQ means give me the default attributes.
1879 2718 4 !-
1880 2719 4
1881 2720 5 IF (.INPUT_RFM EQL FAB$C_VFC)
1882 2721 4 THEN
1883 2722 4
1884 2723 5 IF (.SEQ NEQ 0)
1885 2724 4 THEN
1886 2725 5 BEGIN
1887 2726 5 WORK_FAB [FAB$B_RAT] = .INPUT_RAT;
1888 2727 5 WORK_FAB [FAB$B_RFM] = FAB$C_VFC;
1889 2728 5 WORK_FAB [FAB$W_MRS] = .INPUT_MRS;
1890 2729 5 END
1891 2730 4 ELSE
1892 2731 5 BEGIN
1893 2732 5 WORK_FAB [FAB$B_RAT] = FAB$M_CR;
1894 2733 5 WORK_FAB [FAB$B_RFM] = FAB$C_VAR;
1895 2734 5 WORK_FAB [FAB$W_MRS] = 255;
1896 2735 5 END
1897 2736 5
1898 2737 4 ELSE
1899 2738 5 BEGIN
1900 2739 5 !+
1901 2740 5 ! The input file was not sequenced, so set the record attributes depending
1902 2741 5 ! on whether or not the /SEQ qualifier was specified
1903 2742 5 !-
1904 2743 5 WORK_FAB [FAB$B_RAT] = .INPUT_RAT;
1905 2744 5
1906 2745 6 IF (.SEQ EQL 0)
1907 2746 5 THEN
1908 2747 5 WORK_FAB [FAB$B_RFM] = .INPUT_RFM
1909 2748 5 ELSE
1910 2749 5 WORK_FAB [FAB$B_RFM] = FAB$C_VFC;
1911 2750 5
1912 2751 5 WORK_FAB [FAB$W_MRS] = .INPUT_MRS;
1913 2752 4 END;
1914 2753 4
1915 2754 4 END
1916 2755 3 ELSE
1917 2756 4 BEGIN
1918 2757 4 !+
1919 2758 4 ! Use EDT's default file attributes unless /SEQ is given
1920 2759 4 !-
1921 2760 4 WORK_FAB [FAB$B_RAT] = FAB$M_CR;
1922 2761 4
1923 2762 4 IF (.SEQ EQL 0) THEN WORK_FAB [FAB$B_RFM] = FAB$C_VAR ELSE WORK_FAB [FAB$B_RFM] = FAB$C_VFC;
1924 2763 4
1925 2764 4 WORK_FAB [FAB$W_MRS] = 255;
1926 2765 3 END;
1927 2766 3
1928 2767 3 !+
1929 2768 3 ! Initialize the name blocks.
1930 2769 3 !-
1931 P 2770 3 $NAM INIT (NAM = NAM_BLOCK, ESA = MYBUF2, ESS = MIN (255, NAM$C_MAXRSS), RSA = MYBUF,
1932 2771 3 RSS = MIN (255, NAM$C_MAXRSS));
1933 2772 3 NAM_BLOCK_PTR = NAM_BLOCK;
1934 2773 3 WORK_FAB [FAB$L_NAM] = .NAM_BLOCK_PTR;

```

```
: 1935      2774 2      END;
: 1936      2775 2
: 1937      2776 2      [1] :                               ! WRITE
: 1938      2777 3      BEGIN
: 1939      2778 3      !+
: 1940      2779 3      !- Use EDT's default file attributes unless WRITE /SEQ is given
: 1941      2780 3
: 1942      2781 3      WORK_FAB [FAB$B_RAT] = FAB$M_CR;
: 1943      2782 3
: 1944      2783 3      IF (.SEQ EQL 0) THEN WORK_FAB [FAB$B_RFM] = FAB$C_VAR ELSE WORK_FAB [FAB$B_RFM] = FAB$C_VFC;
: 1945      2784 3
: 1946      2785 3      WORK_FAB [FAB$W_MRS] = 255;
: 1947      2786 3      !+
: 1948      2787 3      !- Initialize the name blocks.
: 1949      2788 3
: 1950      P 2789 3      $NAM_INIT (NAM = NAM_BLOCK, ESA = MYBUF2, ESS = MIN (255, NAM$C_MAXRSS), RSA = MYBUF,
: 1951      2790 3      RSS = MIN (255, NAM$C_MAXRSS));
: 1952      2791 3      NAM_BLOCK_PTR = NAM_BLOCK;
: 1953      2792 3      WORK_FAB [FAB$L_NAM] = .NAM_BLOCK_PTR;
: 1954      2793 2      END;
: 1955      2794 2
: 1956      2795 2      [2] :                               ! JOURNAL
: 1957      2796 3      BEGIN
: 1958      2797 3      !+
: 1959      2798 3      !- Use journal attributes
: 1960      2799 3
: 1961      2800 3      $XABPRO_INIT (XAB = XABPRO, PRO = <RD, RWD>);
: 1962      2801 3      WORK_FAB [FAB$L_XAB] = XABPRO;
: 1963      2802 3      WORK_FAB [FAB$B_RAT] = FAB$M_CR;
: 1964      2803 3      WORK_FAB [FAB$B_RFM] = FAB$C_VAR;
: 1965      2804 3      WORK_FAB [FAB$W_MRS] = 255;
: 1966      2805 3      !+
: 1967      2806 3      !- Initialize the special name block for the journal file.
: 1968      2807 3
: 1969      P 2808 3      $NAM_INIT (NAM = JOURNAL_NAM, ESA = MYBUF2, ESS = MIN (255, NAM$C_MAXRSS), RSA = MYBUF,
: 1970      2809 3      RSS = MIN (255, NAM$C_MAXRSS));
: 1971      2810 3      NAM_BLOCK_PTR = JOURNAL_NAM;
: 1972      2811 3      WORK_FAB [FAB$L_NAM] = .NAM_BLOCK_PTR;
: 1973      2812 2      END;
: 1974      2813 2      TES;
: 1975      2814 2
: 1976      2815 2      !+
: 1977      2816 2      !- Specify the file name.
: 1978      2817 2
: 1979      2818 2      WORK_FAB [FAB$L_FNA] = .FNA;
: 1980      2819 2      WORK_FAB [FAB$B_FNS] = .FNS;
: 1981      2820 2      !+
: 1982      2821 2      !- Initialize the RAB.
: 1983      2822 2
: 1984      P 2823 2      $RAB_INIT (RAB = .RAB, RAC = SEQ, ROP = <LOC, WBH>, UBF = MYBUF, USZ = 255, FAB = WORK_FAB, RHB = VFCHD,
: 1985      2824 2      [TX = 0]);
: 1986      2825 2      !+
: 1987      2826 2      !- Move in the default name string.
: 1988      2827 2
: 1989      2828 2      $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_DNAM, .DNS, .DNA);
: 1990      2829 2      WORK_FAB [FAB$L_DNA] = .DNA;
: 1991      2830 2      WORK_FAB [FAB$B_DNS] = .DNS;
```

```
1992 2831 2 | +
1993 2832 2 | | Set up the related name from the input file, if requested.
1994 2833 2 | | -
1995 2834 2 | |
1996 2835 2 | | IF .RLF
1997 2836 2 | | THEN
1998 2837 3 | | BEGIN
1999 2838 3 | | $NAM INIT (NAM = RELAT_NAM, ESA = MYBUF2, ESS = NAM$C_MAXRSS, RSA = INPUT_NAME, RSS = NAM$C_MAXRSS);
2000 2839 3 | | RELAT_NAM [NAM$B_RSL] = .INPUT_NAME_LEN;
2001 2840 3 | | $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_RLNAM, .RELAT_NAM [NAM$B_RSL], .RELAT_NAM [NAM$L_RSA]);
2002 2841 3 | | NAM_BLOCK_PTR [NAM$L_RLF] = RELAT_NAM;
2003 2842 2 | | END;
2004 2843 2 | |
2005 2844 2 | | +
2006 2845 2 | | Attempt the open and connect. Return the IFI if they succeed, otherwise
2007 2846 2 | | set up IO_STS and IO_STV and return 0.
2008 2847 2 | | -
2009 2848 2 | | IO_STS = $RMS CREATE (FAB = WORK_FAB);
2010 2849 2 | | $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_CRE_STS, %UPVAL, WORK_FAB [FAB$L_STS]);
2011 2850 2 | | $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_CRE_STV, %UPVAL, WORK_FAB [FAB$L_STV]);
2012 2851 2 | |
2013 2852 2 | | IF .IO_STS
2014 2853 2 | | THEN
2015 2854 3 | | BEGIN
2016 2855 3 | | RSL = .NAM_BLOCK_PTR [NAM$B_RSL];
2017 2856 3 | | STRING_DESC (.FICL_DESC, RSL, MYBUF);
2018 2857 3 | | END
2019 2858 2 | | ELSE
2020 2859 3 | | BEGIN
2021 2860 3 | |
2022 2861 4 | | IF (.NAM_BLOCK_PTR [NAM$B_ESL] NEQ 0)
2023 2862 3 | | THEN
2024 2863 4 | | BEGIN
2025 2864 4 | | RSL = .NAM_BLOCK_PTR [NAM$B_ESL];
2026 2865 4 | | RSA = .NAM_BLOCK_PTR [NAM$L_ESA];
2027 2866 4 | | STRING_DESC (.FICL_DESC, RSL, .RSA); ! get resultant filename
2028 2867 3 | | END;
2029 2868 3 | |
2030 2869 2 | | END;
2031 2870 2 | |
2032 2871 2 | | IF .IO_STS
2033 2872 2 | | THEN
2034 2873 3 | | BEGIN
2035 2874 3 | | $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_CRE_IFI, 2, WORK_FAB [FAB$W_IFI]);
2036 2875 3 | | IO_STS = $RMS CONNECT (RAB = .RAB);
2037 2876 3 | | $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_CON_STS, %UPVAL, RAB [RAB$L_STS]);
2038 2877 3 | | $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_CON_STV, %UPVAL, RAB [RAB$L_STV]);
2039 2878 3 | |
2040 2879 3 | | IF .IO_STS
2041 2880 3 | | THEN
2042 2881 4 | | BEGIN
2043 2882 4 | | $$TRACE (EDT$$STR_IFO, EDT$$STR_SOFO_RNAM, .NAM_BLOCK_PTR [NAM$B_RSL], .NAM_BLOCK_PTR [NAM$L_RSA]);
2044 2883 4 | | $$TRACE (EDT$$STR_OFO, EDT$$STR_SOFO_CON_ISI, 2, RAB [RAB$W_ISI]);
2045 2884 4 | | .STV = .IO_STV;
2046 2885 4 | | .STS = .IO_STS;
2047 2886 4 | | RETURN (.WORK_FAB [FAB$W_IFI]);
2048 2887 4 | | END
```

```

: 2049      2888      3      ELSE
: 2050      2889      3      IO_STV = .RAB [RAB$L_STV]
: 2051      2890      3
: 2052      2891      3      END
: 2053      2892      2      ELSE
: 2054      2893      2      IO_STV = .WORK_FAB [RAB$L_STV];
: 2055      2894      2
: 2056      2895      2      .STV = .IO_STV;
: 2057      2896      2      .STS = .IO_STV;
: 2058      2897      2      RETURN (0);
: 2059      2898      1      END;

```

```
! of routine EDT$OPN_OF IDEF
```

```
$RMS_PTR=      NAM_BLOCK
$RMS_PTR=      NAM_BLOCK
$RMS_PTR=      JOURNAL_NAM
```

Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$OPN_OFIDEF - open a file for output by na

G 16
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 62
(19)

			0083	CE	FE24	CA	90	0009C	MOVW	INPUT_RFM, WORK_FAB+31	2747
						05	11	000A3	BRB	7\$	
			0083	CE		03	90	000A5	6\$: MOVW	#3, WORK_FAB+31	2749
			8E	AD	FE2C	CA	80	000AA	7\$: MOVW	INPUT_MRS, WORK_FAB+54	2751
						24	11	000B0	BRB	14\$	2705
			0082	CE		02	90	000B2	8\$: MOVW	#2, WORK_FAB+30	2760
						57	D5	000B7	TSTL	R7	2762
						09	11	000B9	BRB	10\$	
			0082	CE		02	90	000BB	9\$: MOVW	#2, WORK_FAB+30	2781
					14	AC	D5	000C0	TSTL	SEQ	2783
						07	12	000C3	10\$: BNEQ	12\$	
			0083	CE		02	90	000C5	11\$: MOVW	#2, WORK_FAB+31	
						05	11	000CA	BRB	13\$	
			0083	CE		03	90	000CC	12\$: MOVW	#3, WORK_FAB+31	
			8E	AD	FF	8F	9B	000D1	13\$: MOVZBW	#255, WORK_FAB+54	2785
0060	8F	00		6E		00	2C	000D6	14\$: MOVW	#0, (SP), #0, #96, \$RMS_PTR	2790
					FE98	CA		000DD			
			FE98	CA	6002	8F	80	000E0	MOVW	#24578, \$RMS_PTR	
			FE9A	CA		01	8E	000E7	MNEGB	#1, \$RMS_PTR+2	
			FE9C	CA	FB1C	CA	9E	000EC	MOVAB	MYBUF, \$RMS_PTR+4	
			FEA2	CA		01	8E	000F3	MNEGB	#1, \$RMS_PTR+10	
			FEA4	CA	FC1C	CA	9E	000F8	MOVAB	MYBUF2, \$RMS_PTR+12	
				57	FE98	CA	9E	000FF	MOVAB	NAM_BLOCK, NAM_BLOCK_PTR	2791
						55	11	00104	BRB	16\$	2699
0058	8F	00		6E		00	2C	00106	15\$: MOVW	#0, (SP), #0, #88, \$RMS_PTR	2800
					A8	AD		0010D			
			A8	AD	5B13	8F	80	0010F	MOVW	#22547, \$RMS_PTR	
			B0	AD	FF46	8F	80	00115	MOVW	#-186, \$RMS_PTR+8	
			FF7C	CD	A8	AD	9E	0011B	MOVAB	XABPRO, WORK_FAB+36	2801
			0082	CE	0202	8F	80	00121	MOVW	#514, WORK_FAB+30	2802
			8E	AD	FF	8F	9B	00128	MOVZBW	#255, WORK_FAB+54	2804
0060	8F	00		6E		00	2C	0012D	MOVW	#0, (SP), #0, #96, \$RMS_PTR	2809
					FE38	CA		00134			
			FE38	CA	6002	8F	80	00137	MOVW	#24578, \$RMS_PTR	
			FE3A	CA		01	8E	0013E	MNEGB	#1, \$RMS_PTR+2	
			FE3C	CA	FB1C	CA	9E	00143	MOVAB	MYBUF, \$RMS_PTR+4	
			FE42	CA		01	8E	0014A	MNEGB	#1, \$RMS_PTR+10	
			FE44	CA	FC1C	CA	9E	0014F	MOVAB	MYBUF2, \$RMS_PTR+12	
				57	FE38	CA	9E	00156	MOVAB	JOURNAL_NAM, NAM_BLOCK_PTR	2810
			80	AD		57	D0	0015B	16\$: MOVL	NAM_BLOCK_PTR, WORK_FAB+40	2773
			84	AD		59	D0	0015F	MOVL	FNA, WORK_FAB+44	2818
			8C	AD		56	90	00163	MOVW	FNS, WORK_FAB+52	2819
				56	04	AC	D0	00167	MOVL	RAB, R6	2824
0044	8F	00		6E		00	2C	0016B	MOVW	#0, (SP), #0, #68, (R6)	
						66		00172			
				66	4401	8F	80	00173	MOVW	#17409, (R6)	
			04	A6	C0010400	8F	D0	00178	MOVL	#66560, 4(R6)	
					1E	A6	94	00180	CLRB	30(R6)	
			20	A6	FF	8F	9B	00183	MOVZBW	#255, 32(R6)	
			24	A6	FB1C	CA	9E	00188	MOVAB	MYBUF, 36(R6)	
			2C	A6	08	AA	9E	0018E	MOVAB	VFCHD, 44(R6)	
			3C	A6	64	AE	9E	00193	MOVAB	WORK_FAB, 60(R6)	
			88	AD	0C	AC	D0	00198	MOVL	DNA, WORK_FAB+48	2829
			8D	AD	10	AC	90	0019D	MOVW	DNS, WORK_FAB+53	2830
				2E	18	AC	E9	001A2	BLBC	RLF, 17\$	2835
0060	8F	00		6E		00	2C	001A6	MOVW	#0, (SP), #0, #96, \$RMS_PTR	2838
					04	AE		001AD			

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS

EDT\$SOPN_OFIDEF - open a file for output by na

H 16

16-Sep-1984 00:35:33

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 63

(19)

04	AE	6002	8F	B0	001AF	MOVW	#24578, \$RMS_PTR	:	
06	AE		01	8E	001B5	MNEGB	#1, \$RMS_PTR+2	:	
08	AE	FD1C	CA	9E	001B9	MOVAB	INPUT_NAME, \$RMS_PTR+4	:	
0E	AE		01	8E	001BF	MNEGB	#1, \$RMS_PTR+10	:	
10	AE	FC1C	CA	9E	001C3	MOVAB	MYBUF2, \$RMS_PTR+12	:	
07	AE	FE1C	CA	90	001C9	MOVB	INPUT_NAME_LEN, RELAT_NAM+3	:	2839
10	A7	04	AE	9E	001CF	MOVAB	RELAT_NAM, -16(NAM_BLOCK_PTR)	:	2841
		64	AE	9F	001D4	PUSHAB	WORK_FAB	:	2848
00000000G	00		01	FB	001D7	CALLS	#1, SYS\$CREATE	:	
	6A		50	D0	001DE	MOVL	R0, IO_STS	:	
	0A		6A	E9	001E1	BLBC	IO_STS, 18\$	:	2852
	6E	03	A7	9A	001E4	MOVZBL	3(NAM_BLOCK_PTR), RSL	:	2855
		FB1C	CA	9F	001E8	PUSHAB	MYBUF	:	2856
			0F	11	001EC	BRB	19\$	:	
		0B	A7	95	001EE	TSTB	11(NAM_BLOCK_PTR)	:	2861
			16	13	001F1	BEQL	20\$	:	
	6E	0B	A7	9A	001F3	MOVZBL	11(NAM_BLOCK_PTR), RSL	:	2864
	50	0C	A7	D0	001F7	MOVL	12(NAM_BLOCK_PTR), RSA	:	2865
			50	DD	001FB	PUSHL	RSA	:	2866
		04	AE	9F	001FD	PUSHAB	RSL	:	
			58	DD	00200	PUSHL	R8	:	
00000000G	00		03	FB	00202	CALLS	#3, STR\$COPY_R	:	
	24		6A	E9	00209	BLBC	IO_STS, 22\$	:	2871
			56	DD	0020C	PUSHL	R6	:	2875
00000000G	00		01	FB	0020E	CALLS	#1, SYS\$CONNECT	:	
	6A		50	D0	00215	MOVL	R0, IO_STS	:	
	0E		6A	E9	00218	BLBC	IO_STS, 21\$	:	2879
28	BC	04	AA	DC	0021B	MOVL	IO_STV, @STV	:	2884
24	BC		6A	D0	00220	MOVL	IO_STS, @STS	:	2885
	50	66	AE	3C	00224	MOVZWL	WORK_FAB+2, R0	:	2886
				04	00228	RET		:	
04	AA	0C	A6	D0	00229	MOVL	12(R6), IO_STV	:	2889
			05	11	0022E	BRB	23\$	:	2879
04	AA	70	AE	D0	00230	MOVL	WORK_FAB+12, IO_STV	:	2893
28	BC	04	AA	D0	00235	MOVL	IO_STV, @STV	:	2895
24	BC		6A	D0	0023A	MOVL	IO_STS, @STS	:	2896
			50	D4	0023E	CLRL	R0	:	2897
			04	00240	RET			:	2898

; Routine Size: 577 bytes, Routine Base: _EDT\$CODE + 08D7

; 2060 2899 1

```
2062 2900 1 %SBTTL 'EDT$$CLS_FI - close a file'
2063 2901 1
2064 2902 1 GLOBAL ROUTINE EDT$$CLS_FI (
2065 2903 1     IFI,
2066 2904 1     RAB,
2067 2905 1     DEL,
2068 2906 1     NAME_DESC,
2069 2907 1     STS,
2070 2908 1     STV
2071 2909 1 ) =
2072 2910 1
2073 2911 1 ++
2074 2912 1 FUNCTIONAL DESCRIPTION:
2075 2913 1
2076 2914 1     Close a file and optionally delete it.
2077 2915 1
2078 2916 1 FORMAL PARAMETERS:
2079 2917 1
2080 2918 1     IFI             The internal file identifier.
2081 2919 1
2082 2920 1     RAB             The RAB which was connected.
2083 2921 1
2084 2922 1     DEL             1 = delete the last file opened, 2 = delete the journal file
2085 2923 1
2086 2924 1     NAME_DESC      Name of the file
2087 2925 1
2088 2926 1     STS             Resulting RMS STS value
2089 2927 1
2090 2928 1     STV             Resulting RMS STV value
2091 2929 1
2092 2930 1 IMPLICIT INPUTS:
2093 2931 1
2094 2932 1     NONE
2095 2933 1
2096 2934 1 IMPLICIT OUTPUTS:
2097 2935 1
2098 2936 1     NONE
2099 2937 1
2100 2938 1 ROUTINE VALUE:
2101 2939 1
2102 2940 1     NONE
2103 2941 1
2104 2942 1 SIDE EFFECTS:
2105 2943 1
2106 2944 1     None.
2107 2945 1
2108 2946 1 --
2109 2947 1
2110 2948 2 BEGIN
2111 2949 2
2112 2950 2 MAP
2113 2951 2     RAB : $RAB_DECL,
2114 2952 2     NAME_DESC : BLOCK [8, BYTE];
2115 2953 2
2116 2954 2 LOCAL
2117 2955 2     WORK_FAB : $FAB_DECL;
2118 2956 2
```

```
! of routine EDTSSCLS_FI
```

Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$CLS_FI - close a file

K 16
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 66
(20)

02	AE	04	AC	B0	00033	MOVW	IF1, WORK_FAB+2	:	2971
		0C	AC	D5	00038	TSTL	DEL	:	2978
			20	13	0003B	BEQL	3\$	:	
	01	0C	AC	D1	0003D	CMPL	DEL, #1	:	2982
			0A	12	00041	BNEQ	1\$	:	
28	AE	00000000'	EF	9E	00043	MOVAB	NAM_BLOCK, WORK_FAB+40	:	
			08	11	0004B	BRB	2\$	:	
28	AE	00000000'	EF	9E	0004D	MOVAB	JOURNAL_NAM, WORK_FAB+40	:	
05	AE	00010080	8F	C8	00055	BISL2	#65664, WORK_FAB+Z	:	2984
			5E	DD	0005D	PUSHL	SP	:	2990
00000000G	00		01	FB	0005F	CALLS	#1, SYSSCLOSE	:	
	66		50	D0	00066	MOVL	R0, (R6)	:	
18	BC	0C	AE	D0	00069	MOVL	WORK_FAB+12, @STV	:	2991
	50		66	D0	0006E	MOVL	(R6), R0	:	2994
			04	00071	RET			:	2995

; Routine Size: 114 bytes, Routine Base: _EDT\$CODE + 0B18

; 2158 2996 1

EDT\$10MOD
V04-000

EDT\$10MOD - I/O for VAX/VMS

EDT\$\$RD_IF1 - get a record from a file open fo

L 16

16-Sep-1984 00:35:33

14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 67

(21)

```
: 2160 2997 1 %$BTTL 'EDT$$RD_IF1 - get a record from a file open for input'
: 2161 2998 1
: 2162 2999 1 GLOBAL ROUTINE EDT$$RD_IF1 (
: 2163 3000 1     RAB,
: 2164 3001 1     FILE_DESC,
: 2165 3002 1     RHB_DESC,
: 2166 3003 1     STS,
: 2167 3004 1     STV,
: 2168 3005 1     VFC
: 2169 3006 1 ) =
: 2170 3007 1
: 2171 3008 1 ++
: 2172 3009 1 FUNCTIONAL DESCRIPTION:
: 2173 3010 1
: 2174 3011 1     Get a record from a file which is open for input.
: 2175 3012 1
: 2176 3013 1 FORMAL PARAMETERS:
: 2177 3014 1
: 2178 3015 1     RAB           The address of the RAB to connect.
: 2179 3016 1
: 2180 3017 1     FILE_DESC    The address of a descriptor containing filename
: 2181 3018 1
: 2182 3019 1     RHB_DESC    The address of header info descriptor
: 2183 3020 1
: 2184 3021 1     STS         RMS status returned
: 2185 3022 1
: 2186 3023 1     STV         RMS STV word returned
: 2187 3024 1
: 2188 3025 1     VFC         1 = file has sequence numbers, 0 = it doesn't
: 2189 3026 1
: 2190 3027 1 IMPLICIT INPUTS:
: 2191 3028 1
: 2192 3029 1     NONE
: 2193 3030 1
: 2194 3031 1 IMPLICIT OUTPUTS:
: 2195 3032 1
: 2196 3033 1     NONE
: 2197 3034 1
: 2198 3035 1 ROUTINE VALUE:
: 2199 3036 1
: 2200 3037 1     If error or EOF, 0; if success, 1.
: 2201 3038 1
: 2202 3039 1 SIDE EFFECTS:
: 2203 3040 1
: 2204 3041 1     NONE
: 2205 3042 1
: 2206 3043 1 --
: 2207 3044 1
: 2208 3045 2 BEGIN
: 2209 3046 2
: 2210 3047 2 LOCAL
: 2211 3048 2     REC_ADDR,
: 2212 3049 2     REC_LEN;
: 2213 3050 2
: 2214 3051 2 MAP
: 2215 3052 2     RHB_DESC : REF BLOCK [, BYTE],
: 2216 3053 2     RAB : REF $RAB_DECL;
```

```
2217 3054 2
2218 3055 2    $$TRACE (EDT$$TR_IFR, EDT$$TR_SIFR_ISI, 2, RAB [RAB$W_ISI]);
2219 3056 2    VFCHD = 0;
2220 3057 2    +
2221 3058 2    Do the get.
2222 3059 2    -
2223 3060 2    .STS = $RMS_GET (RAB = .RAB);
2224 3061 2    .STV = .RAB[RAB$L_STV];
2225 3062 2    $$TRACE (EDT$$TR_IFR, EDT$$TR_SIFR_STS, %UPVAL, RAB [RAB$L_STS]);
2226 3063 2    $$TRACE (EDT$$TR_IFR, EDT$$TR_SIFR_STV, %UPVAL, RAB [RAB$L_STV]);
2227 3064 2    +
2228 3065 2    Return the record address and length.
2229 3066 2    -
2230 3067 2    $$TRACE (EDT$$TR_IFR, EDT$$TR_SIFR_REC, .RAB [RAB$W_RSZ], .RAB [RAB$L_RBF]);
2231 3068 2    REC_ADDR = .RAB [RAB$L_RBF];
2232 3069 2    REC_LEN = .RAB [RAB$W_RSZ];
2233 3070 2    STRING_DESC (.FILE_DESC, REC_LEN, .REC_ADDR);
2234 3071 2    +
2235 3072 2    If the file has sequence numbers, the sequence number of this record is in
2236 3073 2    the first two bytes of the record prefix, in binary.
2237 3074 2    -
2238 3075 2
2239 3076 2    IF (.VFC EQL 1) THEN REC_LEN = 2 ELSE REC_LEN = 0;
2240 3077 2
2241 3078 2    STRING_DESC (.RHB_DESC, REC_LEN, VFCHD);
2242 3079 2    +
2243 3080 2    Check for bad status.
2244 3081 2    -
2245 3082 2
2246 3083 2    IF ( NOT ..STS) THEN RETURN (0);
2247 3084 2
2248 3085 2    RETURN (1);
2249 3086 1    END;
```

! of routine EDT\$\$RD_IFI

```
001C 00000
54 00000000' EF 9E 00002
53 00000000G 00 9E 00009
5E          04 C2 00010
          64 D4 00013
52          04 AC D0 00015
          52 DD 00019
00000000G 00 01 FB 00018
          10 BC 50 D0 00022
          14 BC 0C A2 D0 00026
          50 28 A2 D0 0002B
          6E 22 A2 3C 0002F
          50 DD 00033
          04 AE 9F 00035
          08 AC DD 00038
          63 03 FB 0003B
          01 18 AC D1 0003E
          05 12 00042
          6E 02 D0 00044
```

.EXTRN SYSSGET

```
.ENTRY EDT$$RD_IFI, Save R2,R3,R4
MOVAB VFCHD, R4
MOVAB STR$COPY_R, R3
SUBL2 #4, SP
CLRL VFCHD
MOVL RAB, R2
PUSHL R2
CALLS #1, SYSSGET
MOVL R0, @STS
MOVL 12(R2), @STV
MOVL 40(R2), REC_ADDR
MOVZWL 34(R2), REC_LEN
PUSHL REC_ADDR
PUSHAB REC_LEN
PUSHL FILE_DESC
CALLS #3, STR$COPY_R
CMPL VFC, #1
BNEQ 1$
MOVL #2, REC_LEN
```

```
: 2999
:
:
: 3056
: 3060
:
: 3061
: 3068
: 3069
: 3070
:
: 3076
:
```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$RD_IFI - get a record from a file open fo

B 1
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 69
(21)

	02	11	00047	BRB	2\$	
	6E	D4	00049	CLRL	REC_LEN	
	54	DD	0004B	PUSHL	R4	3078
04	AE	9F	0004D	PUSHAB	REC_LEN	
0C	AC	DD	00050	PUSHL	RHB_DESC	
63	03	FB	00053	CALLS	#3-STR\$COPY_R	
04	BC	E9	00056	BLBC	@SIS, 3\$	3083
50	01	D0	0005A	MOVL	#1, R0	3085
		04	0005D	RET		
	50	D4	0005E	CLRL	R0	3086
		04	00060	RET		

; Routine Size: 97 bytes, Routine Base: _EDT\$CODE + 0B8A

; 2250 3087 1

```
2252 3088 1 %SBTTL 'EDT$SWR_OFI - put a record to an output file'
2253 3089 1
2254 3090 1 GLOBAL ROUTINE EDT$SWR_OFI (                               | Put a record to an output file
2255 3091 1     RAB,                                           | Address of the RAB
2256 3092 1     FILE_DESC,
2257 3093 1     RHB_DESC,
2258 3094 1     STS,
2259 3095 1     STV
2260 3096 1     ) =
2261 3097 1
2262 3098 1 ++
2263 3099 1 FUNCTIONAL DESCRIPTION:
2264 3100 1
2265 3101 1     Put a record to an output file.
2266 3102 1
2267 3103 1 FORMAL PARAMETERS:
2268 3104 1
2269 3105 1     RAB                The address of the RAB to connect.
2270 3106 1
2271 3107 1     FILE_DESC          The address of a descriptor containing filename
2272 3108 1
2273 3109 1     RHB_DESC            The address of header info descriptor
2274 3110 1
2275 3111 1     STS                RMS status returned
2276 3112 1
2277 3113 1     STV                RMS STV word returned
2278 3114 1
2279 3115 1 IMPLICIT INPUTS:
2280 3116 1
2281 3117 1     NONE
2282 3118 1
2283 3119 1 IMPLICIT OUTPUTS:
2284 3120 1
2285 3121 1     NONE
2286 3122 1
2287 3123 1 ROUTINE VALUE:
2288 3124 1
2289 3125 1     1 if the operation succeeds, status if it fails.
2290 3126 1
2291 3127 1 SIDE EFFECTS:
2292 3128 1
2293 3129 1     Signals on error.
2294 3130 1
2295 3131 1 --
2296 3132 1
2297 3133 2 BEGIN
2298 3134 2
2299 3135 2 MAP
2300 3136 2     RHB_DESC : REF BLOCK [8, BYTE],
2301 3137 2     FILE_DESC : REF BLOCK [8, BYTE],
2302 3138 2     RAB : REF $RAB_DECL;
2303 3139 2
2304 3140 2     $$TRACE (EDT$$STR_OFI, EDT$$STR_SOFW_ISI, 2, RAB [RAB$W_ISI]);
2305 3141 2
2306 3142 2 Set up the record address and size.
2307 3143 2
2308 3144 2     RAB [RAB$L_RBF] = .FILE_DESC [DSC$A_POINTER];
```

```
2309 3145 2 RAB [RAB$W_RSZ] = .FILE_DESC [DSC$W_LENGTH];
2310 3146 2 $$TRACE (EDT$$STR_OFW, EDT$$STR_SOFW_REC, .RAB [RAB$W_RSZ], .RAB [RAB$L_RBF]);
2311 3147 2
2312 3148 2 + Set up the two bytes of record header, in case this file has sequence numbers.
2313 3149 2 - If for some reason this file has record headers longer than two bytes, the
2314 3150 2 remaining bytes will be zero.
2315 3151 2
2316 3152 2 CH$COPY (.RHB_DESC [DSC$W_LENGTH], .RHB_DESC [DSC$A_POINTER], 0, 255, VFCHD);
2317 3153 2
2318 3154 2 + Do the operation, if it succeeds, then return good status.
2319 3155 2 -
2320 3156 2
2321 3157 2 WHILE 1 DO
2322 3158 2 BEGIN
2323 3159 2 .STS = $RMS_PUT (RAB = .RAB);
2324 3160 2 .STV = .RAB [RAB$L_STV];
2325 3161 2 $$TRACE (EDT$$STR_OFW, EDT$$STR_SOFW_STS, %UPVAL, RAB [RAB$L_STS]);
2326 3162 2 $$TRACE (EDT$$STR_OFW, EDT$$STR_SOFW_STV, %UPVAL, RAB [RAB$L_STV]);
2327 3163 2
2328 3164 2 IF (..STS) THEN RETURN (1);
2329 3165 2
2330 3166 2 +
2331 3167 2 Check for the exceeded quota status. Use the CTX field of the rab
2332 3168 2 to indicate whether or not this file has already exceeded.
2333 3169 2 -
2334 3170 2
2335 3171 2 IF ((..STS EQL RMS$_EXT) AND (..STV EQL SS$_EXDISKQUOTA))
2336 3172 2 THEN
2337 3173 2 BEGIN
2338 3174 2
2339 3175 2 IF .PAB [RAB$L_CTX] THEN RETURN (0) ELSE RAB [RAB$L_CTX] = .RAB [RAB$L_CTX] + 1;
2340 3176 2
2341 3177 2 END
2342 3178 2 ELSE
2343 3179 2 EXITLOOP;
2344 3180 2
2345 3181 2 END;
2346 3182 2
2347 3183 2 +
2348 3184 2 Bad status other than exceeded disk quota, put out a message and return 0.
2349 3185 2 -
2350 3186 2 RETURN (0);
2351 3187 1 END;
```

! of routine EDT\$\$WR_OF1

.EXTRN SYSSPUT

```
.ENTRY EDT$$WR_OF1, Save R2,R3,R4,R5,R6
MOVL RAB, R6
MOVL FILE_DESC, R0
MOVL 4(R0), 40(R6)
MOVW (R0), 34(R6)
MOVL RHB_DESC, R0
MOVCS (R0), 24(R0), #0, #255, VFCHD
PUSHL R6
```

00FF 8F

00

04

B0

00000000

```
007C 00000
AC D0 00002
AC D0 00006
AO D0 0000A
60 B0 0000F
AC D0 00013
60 2C 00017
EF 0001F
56 DD 00024 1$:
```

```
3090
3144
3145
3152
3159
```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$SWR_OFI - put a record to an output file

E 1
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 72
(22)

00000000G	00		01	FB	00026	CALLS	#1, SYSSPUT		
	10		50	D0	0002D	MOVL	R0, @STS		
	14		A6	D0	00031	MOVL	12(R6), @STV	3160	
		0C	BC	E9	00036	BLBC	@STS, 2\$	3164	
		10	01	D0	0003A	MOVL	#1, R0		
				04	0003D	RET			
0001C022	8F	10	BC	D1	0003E	CMPL	@STS, #114722	3171	
			13	12	00046	BNEQ	3\$		
000003EC	8F	14	BC	D1	00048	CMPL	@STV, #1004		
			09	12	00050	BNEQ	3\$		
	05	18	A6	E8	00052	BLBS	24(R6), 3\$	3175	
		18	A6	D6	00056	INCL	24(R6)		
			C9	11	00059	BRB	1\$	3171	
			50	D4	0005B	CLRL	R0	3187	
				04	0005D	RET			

; Routine Size: 94 bytes, Routine Base: _EDT\$CODE + 0BEB

; 2352 3188 1

```
2354 3189 1 %SBTTL 'EDT$$REN_FI - rename a temporary file to a permanent file'
2355 3190 1
2356 3191 1 GLOBAL ROUTINE EDT$$REN_FI (
2357 3192 1     OLD_FILE_DESC      | Rename a file
2358 3193 1     ,NEW_FILE_DESC     | descriptor of old filename
2359 3194 1     ,FORCE_MAXV      | descriptor of new filename
2360 3195 1     ,STS              | Force maximum version number
2361 3196 1     ,STV              | rms status word
2362 3197 1     ) =          | rms status value word
2363 3198 1
2364 3199 1 ++
2365 3200 1 FUNCTIONAL DESCRIPTION:
2366 3201 1     This routine renames a temporary file to a permanent output file
2367 3202 1     name string.
2368 3203 1
2369 3204 1 FORMAL PARAMETERS:
2370 3205 1
2371 3206 1     OLD_FILE_DESC      The address of a descriptor containing an old filename
2372 3207 1
2373 3208 1     NEW_FILE_DESC      The address of a descriptor containing a new filename
2374 3209 1
2375 3210 1     FORCE_MAXV          Force maximum version number
2376 3211 1
2377 3212 1     STS                RMS status returned
2378 3213 1
2379 3214 1     STV                RMS status value word returned
2380 3215 1
2381 3216 1 IMPLICIT INPUTS:
2382 3217 1
2383 3218 1     NONE
2384 3219 1
2385 3220 1 IMPLICIT OUTPUTS:
2386 3221 1
2387 3222 1     IO_STS             The last I/O error message from IOMOD
2388 3223 1     IO_STV
2389 3224 1
2390 3225 1 ROUTINE VALUE:
2391 3226 1
2392 3227 1     If the file is renamed successfully, 1; otherwise, 0.
2393 3228 1
2394 3229 1 SIDE EFFECTS:
2395 3230 1
2396 3231 1     NONE
2397 3232 1
2398 3233 1 --
2399 3234 1
2400 3235 2 BEGIN
2401 3236 2
2402 3237 2 MAP
2403 3238 2     OLD_FILE_DESC : REF BLOCK [, BYTE],
2404 3239 2     NEW_FILE_DESC : REF BLOCK [, BYTE];
2405 3240 2
2406 3241 2 LOCAL
2407 3242 2     O_RSA,
2408 3243 2     O_RSL,
2409 3244 2     O_FNA,
2410 3245 2     O_FNS,
```

```
2411 3246 2      N_RSA,
2412 3247 2      N_RSL,
2413 3248 2      N_DNA,
2414 3249 2      N_DNS,
2415 3250 2      OLD_FAB : $FAB_DECL,
2416 3251 2      NEW_FAB : $FAB_DECL,
2417 3252 2  %IF %DECLARED(%QUOTE XAB$B PROT_OPT) %THEN
2418 3253 2      XABPRO : $XABPRO_DECL,
2419 3254 2  %FI
2420 3255 2      OLD_NAM_BLOCK : $NAM_DECL,
2421 3256 2      OLD_NAM_BLOCK_PTR : REF $NAM_DECL,
2422 3257 2      NEW_NAM_BLOCK : $NAM_DECL,
2423 3258 2      NEW_NAM_BLOCK_PTR : REF $NAM_DECL;
2424 3259 2
2425 3260 2      O_FNA = .OLD_FILE_DESC [DSC$A_POINTER];
2426 3261 2      O_FNS = .OLD_FILE_DESC [DSC$W_LENGTH];
2427 3262 2      N_DNA = .NEW_FILE_DESC [DSC$A_POINTER];
2428 3263 2      N_DNS = .NEW_FILE_DESC [DSC$W_LENGTH];
2429 3264 2  !+
2430 3265 2  !- Setup the FAB for the open.
2431 3266 2  !-
2432 3267 2      $FAB_INIT (FAB = OLD_FAB);
2433 3268 2  %IF %DECLARED(%QUOTE XAB$B PROT_OPT) %THEN
2434 3269 2      $FAB_INIT (FAB = NEW_FAB, XAB= XABPRO);
2435 3270 2      $XABPRO_INIT (XAB = XABPRO, PROT_OPT = <PROPAGATE>);
2436 3271 2  %ELSE
2437 3272 2      $FAB_INIT (FAB = NEW_FAB);
2438 3273 2  %FI
2439 3274 2  !+
2440 3275 2  !- Initialize the name blocks.
2441 3276 2  !-
2442 3277 2      $NAM_INIT (  NAM = OLD_NAM_BLOCK
2443 3278 2      ,RSA = MYBUF
2444 3279 2      ,RSS = MIN (255, NAM$C_MAXRSS));
2445 3280 2      OLD_NAM_BLOCK_PTR = OLD_NAM_BLOCK;
2446 3281 2      OLD_FAB [FAB$C_NAM] = .OLD_NAM_BLOCK_PTR;
2447 3282 2
2448 3283 2      $NAM_INIT (  NAM = NEW_NAM_BLOCK
2449 3284 2      ,RSA = MYBUF2
2450 3285 2      ,RSS = MIN (255, NAM$C_MAXRSS));
2451 3286 2      NEW_NAM_BLOCK_PTR = NEW_NAM_BLOCK;
2452 3287 2      NEW_FAB [FAB$C_NAM] = .NEW_NAM_BLOCK_PTR;
2453 3288 2  !+
2454 3289 2  !- Specify the file name.
2455 3290 2  !-
2456 3291 2      OLD_FAB [FAB$L_FNA] = .O_FNA;
2457 3292 2      OLD_FAB [FAB$B_FNS] = .O_FNS;
2458 3293 2  !+
2459 3294 2  !- In order to create the next highest version number when an explicit file
2460 3295 2  !- version number is given in the file spec, we must pass ';' as the file name.
2461 3296 2  !-
2462 3297 2
2463 3298 2      IF .FORCE_MAXV
2464 3299 2      THEN
2465 3300 3          BEGIN
2466 3301 3              NEW_FAB [FAB$L_FNA] = UPLIT (BYTE (';'));
2467 3302 3              NEW_FAB [FAB$B_FNS] = 1;          ! Length of FNA string
```

```

2468 3303 2      END;
2469 3304 2
2470 3305 2      NEW_FAB [FAB$L_DNA] = .N_DNA;
2471 3306 2      NEW_FAB [FAB$B_DNS] = .N_DNS;
2472 3307 2
2473 3308 2      Attempt the the rename.  Return FAB$L_STS
2474 3309 2
2475 3310 2      IO_STS = $RMS_RENAME (OLDFAB = OLD_FAB, NEWFAB = NEW_FAB);
2476 3311 2
2477 3312 2      IF .IO_STS
2478 3313 2      THEN
2479 3314 3          BEGIN
2480 3315 3              O_RSL = .OLD_NAM_BLOCK_PTR [NAM$B_RSL];
2481 3316 3              STRING_DESC T.OLD_FILE_DESC, O_RSL, MYBUF);
2482 3317 3              N_RSL = .NEW_NAM_BLOCK_PTR [NAM$B_RSL];
2483 3318 3              STRING_DESC T.NEW_FILE_DESC, N_RSL, MYBUF2);
2484 3319 3          END
2485 3320 2      ELSE
2486 3321 3          BEGIN
2487 3322 3
2488 3323 4              IF (.OLD_NAM_BLOCK_PTR [NAM$B_ESL] NEQ 0)
2489 3324 3              THEN
2490 3325 4                  BEGIN
2491 3326 4                      O_RSL = .OLD_NAM_BLOCK_PTR [NAM$B_ESL];
2492 3327 4                      O_RSA = .OLD_NAM_BLOCK_PTR [NAM$L_ESA];
2493 3328 5                      STRING_DESC T.OLD_FILE_DESC, O_RSL, .O_RSA) ! get resultant filename
2494 3329 3                      END;
2495 3330 3
2496 3331 4              IF (.NEW_NAM_BLOCK_PTR [NAM$B_ESL] NEQ 0)
2497 3332 3              THEN
2498 3333 4                  BEGIN
2499 3334 4                      N_RSL = .NEW_NAM_BLOCK_PTR [NAM$B_ESL];
2500 3335 4                      N_RSA = .NEW_NAM_BLOCK_PTR [NAM$L_ESA];
2501 3336 5                      STRING_DESC T.NEW_FILE_DESC, N_RSL, .N_RSA) ! get resultant filename
2502 3337 3                      END;
2503 3338 3
2504 3339 2          END;
2505 3340 2
2506 3341 2      .STS = .IO_STS;
2507 3342 2      IO_STV = .OLD_FAB [FAB$L_STV];
2508 3343 2      .STV = .IO_STV;
2509 3344 2      RETURN (.IO_STS);
2510 3345 1      END;

```

! of routine EDT\$\$REN_FI

			00C49		.BLKB	3	
	3B	00C4C	P.AAD:		.ASCII	\;	
					.EXTRN	SYSS\$RENAME	
			OFFC 00000		.ENTRY	EDT\$\$REN_FI, Save R2,R3,R4,R5,R6,R7,R8,R9,-	3191
						R10,R11	
5E	FE40	CE	9E 00002		MOVAB	-448(SP), SP	
58	04	AC	DD 00007		MOVL	OLD_FILE_DESC, R8	3260
	04	A8	DD 0000B		PUSHL	4(R8)	
5B		68	3C 0000E		MOVZWL	(R8), O_FNS	3261

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$REN_FI - rename a temporary file to a per

I 1
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 76
(23)

0050	8F	00	57 08 AC D0 00011	MOVL	NEW_FILE_DESC, R7	3262
			5A 04 A7 D0 00015	MOVL	4(R7), N_DNA	
			59 67 3C 00019	MOVZWL	(R7), N_DNS	3263
			6E 00 2C 0001C	MOVCS	#0, (SPT), #0, #80, \$RMS_PTR	3267
			B0 AD 5003 8F B0 00023			
			C6 AD 02 90 00025	MOVW	#20483, \$RMS_PTR	
			CF AD 02 90 0002B	MOVB	#2, \$RMS_PTR+22	
0050	8F	00	6E 00 2C 0002F	MOVB	#2, \$RMS_PTR+31	
			FF60 CD 00 2C 00033	MOVCS	#0, (SP), #0, #80, \$RMS_PTR	3269
			5003 8F B0 0003A			
			FF60 CD 02 90 0003D	MOVW	#20483, \$RMS_PTR	
			FF76 CD 02 90 00044	MOVB	#2, \$RMS_PTR+22	
			FF7F CD 02 90 00049	MOVB	#2, \$RMS_PTR+31	
0058	8F	00	84 AD 00CC CE 9E 0004E	MOVAB	XABPRO, \$RMS_PTR+36	
			6E 00 2C 00054	MOVCS	#0, (SP), #0, #88, \$RMS_PTR	3270
			00CC CE 0005B			
			5813 8F B0 0005E	MOVW	#22547, \$RMS_PTR	
0060	8F	00	00D7 CE 01 90 00065	MOVB	#1, \$RMS_PTR+11	
			6E 00 2C 0006A	MOVCS	#0, (SP), #0, #96, \$RMS_PTR	3279
			6C AE 00071			
			6E AE 6002 8F B0 00073	MOVW	#24578, \$RMS_PTR	
			70 AE 00000000' 01 8E 00079	MNEGB	#1, \$RMS_PTR+2	
			56 AE 6C AE 9E 0007D	MOVAB	MYBUF, \$RMS_PTR+4	
			D8 AD 56 D0 00085	MOVAB	OLD_NAM_BLOCK, OLD_NAM_BLOCK_PTR	3280
0060	8F	00	6E 00 2C 00089	MOVL	OLD_NAM_BLOCK_PTR, OLD_FAB+40	3281
			0C AE 00094	MOVCS	#0, (SPT), #0, #96, \$RMS_PTR	3285
			0E AE 6002 8F B0 00096	MOVW	#24578, \$RMS_PTR	
			10 AE 00000000' 01 8E 0009C	MNEGB	#1, \$RMS_PTR+2	
			52 AE 0C AE 9E 000A0	MOVAB	MYBUF2, \$RMS_PTR+4	
			88 AD 52 D0 000A8	MOVAB	NEW_NAM_BLOCK, NEW_NAM_BLOCK_PTR	3286
			DC AD 6E D0 000AC	MOVL	NEW_NAM_BLOCK_PTR, NEW_FAB+40	3287
			E4 AD 5B 90 000B0	MOVL	O_FNA, OLD_FAB+44	3291
			0A 0C AC E9 000B8	MOVB	O_FNS, OLD_FAB+52	3292
			8C AD FF3F CF 9E 000BC	BLBC	FORCE_MAXV, 1\$	3298
			94 AD 01 90 000C2	MOVAB	P.AAD, NEW_FAB+44	3301
			90 AD 5A D0 000C6	MOVB	#1, NEW_FAB+52	3302
			95 AD 59 90 000CA	MOVL	N_DNA, NEW_FAB+48	3305
			FF60 CD 9F 000CE	MOVB	N_DNS, NEW_FAB+53	3306
			7E 7C 000D2	PUSHAB	NEW_FAB	3310
			B0 AD 9F 000D4	CLRQ	-(SP)	
			00 04 FB 000D7	PUSHAB	OLD_FAB	
00000000G			00 EF 50 D0 000DE	CALLS	#4, SYSS\$RENAME	
00000000'			24 00000000' EF E9 000E5	MOVL	R0, IO_STS	
			04 AE 03 A6 9A 000EC	BLBC	IO_STS, 2\$	3312
			00000000' EF 9F 000F1	MOVZBL	3(OLD_NAM_BLOCK_PTR), O_RSL	3315
			08 AE 9F 000F7	PUSHAB	MYBUF	3316
			58 DD 000FA	PUSHAB	O_RSL	
00000000G			00 03 FB 000FC	PUSHL	R8	
08 AE 00000000'			A2 9A 00103	CALLS	#3, STR\$COPY R	
			EF 9F 00108	MOVZBL	3(NEW_NAM_BLOCK_PTR), N_RSL	3317
			2C 11 0010E	PUSHAB	MYBUF2	3318
			08 A6 95 00110	BRB	4\$	
			17 13 00113	TSTB	11(OLD_NAM_BLOCK_PTR)	3323
			04 AE 08 A6 9A 00115	BEQL	3\$	
			50 0C A6 D0 0011A	MOVZBL	11(OLD_NAM_BLOCK_PTR), O_RSL	3326
				MOVL	12(OLD_NAM_BLOCK_PTR), O_RSA	3327

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS

EDT\$\$REN_FI - rename a temporary file to a per

J 1

16-Sep-1984 00:35:33

14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 77

(23)

EDT
V04

			50	DD	0011E	PUSHL	O_RSA	:	3328
		08	AE	9F	00120	PUSHAB	O_RSL	:	
			58	DD	00123	PUSHL	R8	:	
00000000G	00		03	FB	00125	CALLS	#3, STR\$COPY_R	:	
		08	A2	95	0012C	TSTB	11(NEW_NAM_BLOCK_PTR)	:	3331
			17	13	0012F	BEQL	5\$	:	
	08	AE	08	A2	9A	MOVZBL	11(NEW_NAM_BLOCK_PTR), N_RSL	:	3334
		50	0C	A2	D0	MOVL	12(NEW_NAM_BLOCK_PTR), N_RSA	:	3335
				50	DD	PUSHL	N_RSA	:	3336
			0C	AE	9F	PUSHAB	N_RSL	:	
				57	DD	PUSHL	R7	:	
00000000G	00		03	FB	00141	CALLS	#3, STR\$COPY_R	:	
	10	BC	00000000'	EF	D0	MOVL	IO_STS, @STS-	:	3341
00000000'		EF		AD	D0	MOVL	OLD_FAB+12, IO_STV	:	3342
	14	BC	00000000'	EF	D0	MOVL	IO_STV, @STV	:	3343
		50	00000000'	EF	D0	MOVL	IO_STS, R0	:	3344
				04	00167	RET		:	3345

; Routine Size: 360 bytes, Routine Base: _EDT\$CODE + 0C4D

; 2511 3346 1

```

: 2513 3347 1 %SBTTL 'EDT$$FLUSH_OBUF - flush output buffers'
: 2514 3348 1
: 2515 3349 1 GLOBAL ROUTINE EDT$$FLUSH_OBUF (           ! Flush output buffers
: 2516 3350 1     RAB,                                   ! Address of the RAB
: 2517 3351 1     STV                                     !
: 2518 3352 1     ) =
: 2519 3353 1
: 2520 3354 1 ++
: 2521 3355 1 FUNCTIONAL DESCRIPTION:
: 2522 3356 1
: 2523 3357 1     This routine flushes the buffers for a file which is open for output.
: 2524 3358 1
: 2525 3359 1 FORMAL PARAMETERS:
: 2526 3360 1
: 2527 3361 1     RAB                                Address of the RAB, representing the file whose buffers
: 2528 3362 1                                           are to be flushed.
: 2529 3363 1
: 2530 3364 1 IMPLICIT INPUTS:
: 2531 3365 1
: 2532 3366 1     NONE
: 2533 3367 1
: 2534 3368 1 IMPLICIT OUTPUTS:
: 2535 3369 1
: 2536 3370 1     NONE
: 2537 3371 1
: 2538 3372 1 ROUTINE VALUE:
: 2539 3373 1
: 2540 3374 1     The status code from the RMS $FLUSH call
: 2541 3375 1
: 2542 3376 1 SIDE EFFECTS:
: 2543 3377 1
: 2544 3378 1     NONE
: 2545 3379 1
: 2546 3380 1 --
: 2547 3381 1
: 2548 3382 2 BEGIN
: 2549 3383 2
: 2550 3384 2 MAP
: 2551 3385 2     RAB : REF $RAB_DECL;
: 2552 3386 2
: 2553 3387 2 LOCAL
: 2554 3388 2     STATUS;
: 2555 3389 2
: 2556 3390 2 $STRACE (EDT$$STR_FLU, EDT$$STR_SFLU_ISI, 2, RAB [RAB$W_ISI]);
: 2557 3391 2 STATUS = ($RMS FLUSH (RAB = .RAB));
: 2558 3392 2 .STV = .RAB [RAB$L_STV];
: 2559 3393 2 RETURN (.STATUS)
: 2560 3394 1 END;

```

! of routine EDT\$\$FLUSH_OBUF

.EXTRN SYSSFLUSH

```

          52      04      0004 00000
          AC      DO 00002
          52      DD 00006
00000000G 00      01      FB 00008

```

```

.ENTRY EDT$$FLUSH_OBUF, Save R2
MOVL   RAB, R2
PUSHL  R2
CALLS  #1, SYSSFLUSH

```

```

: 3349
: 3391
:
:

```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$FLUSH_OBUF - flush output buffers

L 1
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 79
(24)

EDT
V04

08 BC 0C A2 D0 0000F
04 00014

MOVL 12(R2), @STV
RET

: 3392
: 3394

; Routine Size: 21 bytes, Routine Base: _EDT\$CODE + 0DB5

; 2561 3395 1

```

: 2563 3396 1 %SBTTL 'CC_ENABLE - enable for control C traps'
: 2564 3397 1 ROUTINE CC_ENABLE ! Enable for control C traps
: 2565 3398 1 =
: 2566 3399 1
: 2567 3400 1 ++
: 2568 3401 1 FUNCTIONAL DESCRIPTION:
: 2569 3402 1
: 2570 3403 1 Enable for control C traps. When a control C trap is intercepted, the routine
: 2571 3404 1 CC_AST will be called.
: 2572 3405 1
: 2573 3406 1 FORMAL PARAMETERS:
: 2574 3407 1
: 2575 3408 1 NONE
: 2576 3409 1
: 2577 3410 1 IMPLICIT INPUTS:
: 2578 3411 1
: 2579 3412 1 TT_CHAN
: 2580 3413 1 CC_AST
: 2581 3414 1 T_DEVCLASS
: 2582 3415 1
: 2583 3416 1 IMPLICIT OUTPUTS:
: 2584 3417 1
: 2585 3418 1 NONE
: 2586 3419 1
: 2587 3420 1 ROUTINE VALUE:
: 2588 3421 1
: 2589 3422 1 The status code from $QIOW, or 1.
: 2590 3423 1
: 2591 3424 1 SIDE EFFECTS:
: 2592 3425 1
: 2593 3426 1 Allocates and deallocates an event flag.
: 2594 3427 1
: 2595 3428 1 --
: 2596 3429 1
: 2597 3430 2 BEGIN
: 2598 3431 2
: 2599 3432 2 LOCAL
: 2600 3433 2 GETEF_STATUS,
: 2601 3434 2 FREEEF_STATUS,
: 2602 3435 2 EF,
: 2603 3436 2 QIO_STATUS;
: 2604 3437 2
: 2605 3438 2 +
: 2606 3439 2 This routine is not called if we are reading a trace file.
: 2607 3440 2 -
: 2608 3441 2 $$TRACE_READING (<ASSERT (0)>);
: 2609 3442 2 +
: 2610 3443 2 Do nothing if SYSS$INPUT is not a terminal.
: 2611 3444 2 -
: 2612 3445 2
: 2613 3446 2 IF (.t_devclass NEQ DC$_TERM) THEN RETURN (1);
: 2614 3447 2
: 2615 3448 2 +
: 2616 3449 2 Get an event flag for the UI^
: 2617 3450 2 -
: 2618 3451 2 GETEF_STATUS = LIB$GET_EF (EF);
: 2619 3452 2
```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
CC_ENABLE - enable for control C traps

N 1
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1
Page 81
(25)

EDT
V04

```
: 2620      3453  2      IF ( NOT .GETEF_STATUS) THEN SIGNAL_STOP (.GETEF_STATUS);
: 2621      3454  2
: 2622      3455  2      QIO_STATUS = $QIOW (EFN = .EF, CHAN = .TT_CHAN, FUNC = IOS_SETMODE + IOSM_CTRLCAST, P1 = CC_AST);
: 2623      3456  2      FREEEF_STATUS = LIB$FREE_EF (.EF);
: 2624      3457  2
: 2625      3458  2      IF ( NOT .FREEEF_STATUS) THEN SIGNAL_STOP (.FREEEF_STATUS);
: 2626      3459  2
: 2627      3460  2      RETURN (.QIO_STATUS);
: 2628      3461  1      END;
```

! of routine CC_ENABLE

.EXTRN SYSSQIOW

```
000C 00000 CC_ENABLE:
      53 00000000G 00 9E 00002 .WORD Save R2,R3 : 3397
      5E          04 C2 00009 MOVAB LIB$STOP, R3
00000042 8F 00000000' EF D1 0000C SUBL2 #4, SP : 3446
      50          04 13 00017 CMPL T DEVCLASS, #66
      01          01 D0 00019 BEQL 1$
      04          04 0001C MOVL #1, R0
      5E          5E DD 0001D 1$: PUSHL SP : 3451
00000000G 00          01 FB 0001F CALLS #1, LIB$GET_EF
      05          50 E8 00026 BLBS GETEF_STATUS, 2$ : 3453
      63          50 DD 00029 PUSHL GETEF_STATUS
      01          01 FB 0002B CALLS #1, LIB$STOP
      7E          7E 7C 0002E 2$: CLRQ -(SP) : 3455
      7E          7E 7C 00030 CLRQ -(SP)
      7E          7E D4 00032 CLRL -(SP)
      0000V      CF 9F 00034 PUSHAB CC_AST
      7E          7E 7C 00038 CLRQ -(SP)
      7E          7E D4 0003A CLRL -(SP)
      7E          8F 3C 0003C MOVZWL #291, -(SP)
      0123      EF DD 00041 PUSHL TT_CHAN
      00000000' 2C AE DD 00047 PUSHL EF
00000000G 00          0C FB 0004A CALLS #12, SYSSQIOW
      52          50 D0 00051 MOVL R0, QIO_STATUS
      5E          5E DD 00054 PUSHL SP : 3456
00000000G 0'          01 FB 00056 CALLS #1, LIB$FREE_EF
      05          50 E8 0005D BLBS FREEEF_STATUS, 3$ : 3458
      63          50 DD 00060 PUSHL FREEEF_STATUS
      50          01 FB 00062 CALLS #1, LIB$STOP
      52          52 D0 00065 3$: MOVL QIO_STATUS, R0 : 3460
      04          04 00068 RET : 3461
```

; Routine Size: 105 bytes, Routine Base: _EDT\$CODE + 0DCA

```
2630 3462 1 XSBTTL 'CC_AST - take a control C trap'
2631 3463 1 ROUTINE CC_AST                                ! Take a control C trap
2632 3464 1 : NOVALUE =
2633 3465 1
2634 3466 1 ++
2635 3467 1 FUNCTIONAL DESCRIPTION:
2636 3468 1
2637 3469 1 This is the AST routine for CTRL/C. Simply set the EDT$$G_CC flag and
2638 3470 1 re-enable the ASI.
2639 3471 1
2640 3472 1 FORMAL PARAMETERS:
2641 3473 1
2642 3474 1 NONE
2643 3475 1
2644 3476 1 IMPLICIT INPUTS:
2645 3477 1
2646 3478 1 NONE
2647 3479 1
2648 3480 1 IMPLICIT OUTPUTS:
2649 3481 1
2650 3482 1 EDT$$G_CC
2651 3483 1
2652 3484 1 ROUTINE VALUE:
2653 3485 1
2654 3486 1 NONE
2655 3487 1
2656 3488 1 SIDE EFFECTS:
2657 3489 1
2658 3490 1 Calls CC_ENABLE to re-enable control C traps.
2659 3491 1
2660 3492 1 --
2661 3493 1
2662 3494 2 BEGIN
2663 3495 2 EDT$$G_CC = 1;
2664 3496 2 CC_ENABLE ();
2665 3497 2 END;                                ! of routine CC_AST
```

```
00000000G 00          0000 00000 CC_AST: .WORD Save nothing
01 D0 00002          MOVL #1, EDT$$G_CC
00 FB 00009          CALLS #0, CC_ENABLE
04 0000D          RET
```

```
: 3463
: 3495
: 3496
: 3497
```

; Routine Size: 14 bytes, Routine Base: _EDT\$CODE + 0E33

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$TERM_RCC - restore from control C

C 2
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1
Page 83
(27)

EDT
V04

```
: 2667 3498 1 %SBTTL 'EDT$$TERM_RCC - restore from control C'
: 2668 3499 1
: 2669 3500 1 GLOBAL ROUTINE EDT$$TERM_RCC ! Restore from control C
: 2670 3501 1 : NOVALUE =
: 2671 3502 1
: 2672 3503 1 ++
: 2673 3504 1 FUNCTIONAL DESCRIPTION:
: 2674 3505 1
: 2675 3506 1 Do any system-dependent restoration of the terminal that is needed
: 2676 3507 1 after a control C is processed. None is needed on VAX/VMS.
: 2677 3508 1
: 2678 3509 1 FORMAL PARAMETERS:
: 2679 3510 1
: 2680 3511 1 NONE
: 2681 3512 1
: 2682 3513 1 IMPLICIT INPUTS:
: 2683 3514 1
: 2684 3515 1 NONE
: 2685 3516 1
: 2686 3517 1 IMPLICIT OUTPUTS:
: 2687 3518 1
: 2688 3519 1 NONE
: 2689 3520 1
: 2690 3521 1 ROUTINE VALUE:
: 2691 3522 1
: 2692 3523 1 NONE
: 2693 3524 1
: 2694 3525 1 SIDE EFFECTS:
: 2695 3526 1
: 2696 3527 1 NONE
: 2697 3528 1
: 2698 3529 1 --
: 2699 3530 1
: 2700 3531 2 BEGIN
: 2701 3532 2 RETURN;
: 2702 3533 1 END; ! of routine EDT$$TERM_RCC
```

0000 00000
04 00002

.ENTRY EDT\$\$TERM_RCC, Save nothing
RET

: 3500
: 3533

: Routine Size: 3 bytes, Routine Base: _EDT\$CODE + 0E41

: 2703 3534 1

```
: 2705 3535 1 %SBTTL 'EDT$TI_OPN - open the terminal'
: 2706 3536 1
: 2707 3537 1 GLOBAL ROUTINE EDT$TI_OPN ! Open the terminal
: 2708 3538 1 =
: 2709 3539 1
: 2710 3540 1 ++
: 2711 3541 1 FUNCTIONAL DESCRIPTION:
: 2712 3542 1
: 2713 3543 1 This routine opens the files SYSS$INPUT and SYSS$OUTPUT for terminal
: 2714 3544 1 I/O.
: 2715 3545 1
: 2716 3546 1 FORMAL PARAMETERS:
: 2717 3547 1
: 2718 3548 1 NONE
: 2719 3549 1
: 2720 3550 1 IMPLICIT INPUTS:
: 2721 3551 1
: 2722 3552 1 TT_OPEN
: 2723 3553 1
: 2724 3554 1 IMPLICIT OUTPUTS:
: 2725 3555 1
: 2726 3556 1 TERM_IN_RAB
: 2727 3557 1 IO_STS
: 2728 3558 1 IO-STV
: 2729 3559 1 TERM_OUT_RAB
: 2730 3560 1 TT_OPEN
: 2731 3561 1 EDT$SG-TI-TYP
: 2732 3562 1 EDT$SG-DEC CRT
: 2733 3563 1 EDT$SG-EIGHT BIT
: 2734 3564 1 EDT$SG-TI-SCROLL
: 2735 3565 1 EDT$SG-TI-EDIT
: 2736 3566 1 EDT$SG-ENB AUTRPT
: 2737 3567 1 EDT$SG-TI DUMB
: 2738 3568 1 T_DEVCLASS, etc
: 2739 3569 1
: 2740 3570 1 ROUTINE VALUE:
: 2741 3571 1
: 2742 3572 1 1
: 2743 3573 1
: 2744 3574 1 SIDE EFFECTS:
: 2745 3575 1
: 2746 3576 1 If SYSS$INPUT or SYSS$OUTPUT won't open, calls EDT$IO_ERRMSG ,
: 2747 3577 1 then exits without returning to the caller.
: 2748 3578 1
: 2749 3579 1 --
: 2750 3580 1
: 2751 3581 2 BEGIN
: 2752 3582 2
: 2753 3583 2 LOCAL
: 2754 3584 2 EF_STATUS,
: 2755 3585 2 EF,
: 2756 3586 2 QIO STATUS,
: 2757 3587 2 ITEM_LIST : BLOCKVECTOR [7, 3] FIELD (ITEM_FIELDS), ! Storage for terminal char.
: 2758 3588 2 TT_INFO : BLOCK [2] FIELD (SETM_FIELDS), ! for setting terminal char
: 2759 3589 2 NAME_BUF : VECTOR [64, BYTE],
: 2760 3590 2 NAME_DESC : VECTOR [2],
: 2761 3591 2 RES_DESC : VECTOR [2].
```

```
2762      3592 2      WORK_FAB : $FAB_DECL;
2763      3593 2
2764      3594 2      !+
2765      3595 2      ! If the open has already taken place, return immediately.
2766      3596 2      !-
2767      3597 2
2768      3598 2      IF .TT_OPEN THEN RETURN (1);
2769      3599 2
2770      3600 2      !+
2771      3601 2      ! Don't pass this way again.
2772      3602 2      !-
2773      3603 2      TT_OPEN = 1;
2774      3604 2      !+
2775      3605 2      ! Open the trace file, and write the first record on it.
2776      3606 2      !-
2777      3607 2
2778      3608 2      %IF edt$tr_act
2779      3609 2      %THEN
2780      3610 2      BEGIN
2781      3611 2
2782      3612 2      EXTERNAL ROUTINE
2783      3613 2      EDT$STR_OPN : ADDRESSING_MODE (GENERAL);
2784      3614 2
2785      3615 2      EXTERNAL
2786      3616 2      EDT$SL_TR_INFLG;
2787      3617 2      ! 1 iff input from trace file
2788      3618 2
2789      3619 2      LOCAL
2790      3620 2      OPEN_STATUS,
2791      3621 2      TRACE_READ_STATUS;
2792      3622 2
2793      3623 2      OPEN_STATUS = EDT$STR_OPN (EDT$SL_TR_INFLG);
2794      3624 2
2795      3625 2      IF ( NOT .OPEN_STATUS) THEN SIGNAL_STOP (.OPEN_STATUS);
2796      3626 2
2797      3627 2      $$TRACE (EDT$STR_START, EDT$STR_SSTART_INIT, 0, 0);
2798      3628 2      !+
2799      3629 2      ! If we are reading a trace file, read the first two records and don't open
2800      3630 2      ! the terminal.
2801      3631 2      !-
2802      3632 2      TRACE_READ_STATUS = $$TRACE_READ (EDT$STR_START, EDT$STR_SSTART_INIT, 0, 0, 0);
2803      3633 2
2804      3634 2      IF (.TRACE_READ_STATUS)
2805      3635 2      THEN
2806      3636 2      BEGIN
2807      3637 2      $$TRACE_READ (EDT$STR_START, EDT$STR_SSTART_TERM, %UPVAL, EDT$SG_TI_TYP, 0);
2808      3638 2      EDT$SG_DEC CRT = 1;
2809      3639 2      EDT$SG_EIGHT BIT = 0;
2810      3640 2      EDT$SG_TI_SCROLL = 1;
2811      3641 2      EDT$SG_ENB AUTRPT = 1;
2812      3642 2      EDT$SG_TI_EDIT = 0;
2813      3643 2      EDT$SG_TI_DUMB = 0;
2814      3644 2      t devclass = DCS_TERM;
2815      3645 2      $$TRACE (EDT$STR_START, EDT$STR_SSTART_TERM, %UPVAL, EDT$SG_TI_TYP);
2816      3646 2      END;
2817      3647 2
2818      3648 2      %FI
```

```
2819 3649 2
2820 3650 2
2821 3651 2
2822 3652 2
2823 3653 2
2824 3654 2
2825 3655 2
2826 3656 2
2827 3657 2
2828 3658 2
2829 3659 2
2830 3660 2
2831 3661 2
2832 3662 2
2833 3663 2
2834 3664 2
2835 3665 2
2836 3666 2
2837 3667 2
2838 3668 2
2839 3669 2
2840 3670 2
2841 3671 2
2842 3672 2
2843 3673 2
2844 3674 2
2845 3675 2
2846 3676 2
2847 3677 2
2848 3678 2
2849 3679 2
2850 3680 2
2851 3681 2
2852 3682 2
2853 3683 2
2854 3684 2
2855 3685 2
2856 3686 2
2857 3687 2
2858 3688 2
2859 3689 2
2860 3690 2
2861 3691 2
2862 3692 2
2863 3693 2
2864 3694 2
2865 3695 2
2866 3696 2
2867 3697 2
2868 3698 2
2869 3699 2
2870 3700 2
2871 3701 2
2872 3702 2
2873 3703 2
2874 3704 2
2875 3705 2

+ If we are reading a trace file, bail out now.
-
  $$TRACE_READING (<RETURN (1)>);
+
- Open the SYSS$INPUT file for terminal input.
  $FAB_INIT (FAB = WORK_FAB, FAC = <GET>, RAT = CR);
  WORK_FAB [FAB$L_FNA] = UPLIT BYTE('SYSS$INPUT');
  WORK_FAB [FAB$B_FNS] = 9;
  $RAB_INIT (RAB = TERM_IN_RAB, RAC = SEQ, UBF = MYBUF, USZ = 255, FAB = WORK_FAB, RHB = VFCHD, CTX = 0);

  IF (EDT$$G_IO_STS = $RMS_OPEN (FAB = WORK_FAB))
  THEN
    BEGIN
      EDT$$G_IO_STS = $RMS_CONNECT (RAB = TERM_IN_RAB);
      IF ( NOT .EDT$$G_IO_STS) THEN EDT$$G_IO_STV = .TERM_IN_RAB [RAB$L_STV];
    END
  ELSE
    EDT$$G_IO_STV = .WORK_FAB [FAB$L_STV];

  IF ( NOT .EDT$$G_IO_STS)
  THEN
    BEGIN
      SIGNAL_STOP (EDT$_OPNINTRM, 0, .EDT$$G_IO_STS, .EDT$$G_IO_STV);
    END;

+ Open the SYSS$OUTPUT file for terminal output.
-
  $FAB_INIT (FAB = WORK_FAB, FAC = <PUT>, RAT = CR);
  WORK_FAB [FAB$L_FNA] = UPLIT BYTE('SYSS$OUTPUT');
  WORK_FAB [FAB$B_FNS] = 10;
  $RAB_INIT (RAB = TERM_OUT_RAB, RAC = SEQ, UBF = MYBUF, USZ = 255, FAB = WORK_FAB, RHB = VFCHD, CTX = 0);

  IF (EDT$$G_IO_STS = $RMS_CREATE (FAB = WORK_FAB))
  THEN
    BEGIN
      EDT$$G_IO_STS = $RMS_CONNECT (RAB = TERM_OUT_RAB);
      IF ( NOT .EDT$$G_IO_STS) THEN EDT$$G_IO_STV = .TERM_OUT_RAB [RAB$L_STV];
    END
  ELSE
    EDT$$G_IO_STV = .WORK_FAB [FAB$L_STV];

  IF ( NOT .EDT$$G_IO_STS) THEN SIGNAL_STOP (EDT$_OPNOUTTRM, 0, .EDT$$G_IO_STS, .EDT$$G_IO_STV);

+ Now also open the terminal for $QIOs.
-
  EDT$$G_DEC CRT = 0;          ! Make sure this variable is initialized
  NAME_DESC [0] = 9;
  NAME_DESC [1] = NAME_BUF;
```

```
2876 3706 2 EDT$COPY_MEM (9, UPLIT ('SYSS$INPUT'), NAME_BUF);
2877 3707 2 RES_DESC [1] = NAME_BUF;
2878 3708 2 RES_DESC [0] = 64;
2879 3709 2
2880 3710 2 WHILE ($TRNLOG (LOGNAM = NAME_DESC, RSLLEN = NAME_DESC [0], RSLBUF = RES_DESC) EQL SSS_NORMAL) DO
2881 3711 3 BEGIN
2882 3712 3
2883 3713 4 IF (CH$RCHAR (NAME_BUF) EQL 27)
2884 3714 3 THEN
2885 3715 4 BEGIN
2886 3716 4 NAME_DESC [1] = CH$PTR (NAME_BUF, 4);
2887 3717 4 NAME_DESC [0] = .NAME_DESC [0] - 4;
2888 3718 4 END
2889 3719 3 ELSE
2890 3720 3 NAME_DESC [1] = CH$PTR (NAME_BUF);
2891 3721 3
2892 3722 4 IF (CH$RCHAR (CH$PTR (.NAME_DESC [1], .NAME_DESC [0] - 1)) EQL %C':')
2893 3723 3 THEN
2894 3724 3 NAME_DESC [0] = .NAME_DESC [0] - 1;
2895 3725 3
2896 3726 2 END;
2897 3727 2
2898 3728 2 NAME_DESC [1] = NAME_BUF;
2899 3729 2
2900 3730 2 IF $ASSIGN (DEVNAM = NAME_DESC, CHAN = TT_CHAN)
2901 3731 2 THEN
2902 3732 2 BEGIN
2903 3733 2
2904 3734 2 !+ Set up the item list for the $GETDVI routine. We need to access device
2905 3735 2 type, characteristics, width, dependent characteristics, extended
2906 3736 2 characteristics, and class.
2907 3737 2 -
2908 3738 2
2909 3739 2 INCR I FROM 0 TO 5 DO
2910 3740 4 BEGIN
2911 3741 4 ITEM_LIST [I, BF_LEN] = 4;
2912 3742 4 ITEM_LIST [I, RLEN_ADR] = 0;
2913 3743 3 END;
2914 3744 3
2915 3745 3 !
2916 3746 3 ITEM_LIST [0, I_CODE] = DVI$DEVCHAR;
2917 3747 3 ITEM_LIST [0, BOF_ADR] = T_DEVCHAR;
2918 3748 3 ITEM_LIST [1, I_CODE] = DVI$DEVCLASS;
2919 3749 3 ITEM_LIST [1, BOF_ADR] = t_devclass;
2920 3750 3 ITEM_LIST [2, I_CODE] = DVI$DEVDEPEND;
2921 3751 3 ITEM_LIST [2, BOF_ADR] = T_DEVDEP;
2922 3752 3 ITEM_LIST [3, I_CODE] = DVI$DEVDEPEND2;
2923 3753 3 ITEM_LIST [3, BOF_ADR] = t_devextdep;
2924 3754 3 ITEM_LIST [4, I_CODE] = DVI$DEVTYPE;
2925 3755 3 ITEM_LIST [4, BOF_ADR] = t_devtype;
2926 3756 3 ITEM_LIST [5, I_CODE] = DVI$DEVBUFSIZ;
2927 3757 3 ITEM_LIST [5, BOF_ADR] = T_DEVWID;
2928 3758 3 ITEM_LIST [6, I_CODE] = 0; ! End of item list
2929 3759 3
2930 3760 3 !+ Get an event flag
2931 3761 3 -
2932 3762 3 EF_STATUS = LIB$GET_EF (EF);
```

: R

: 3

```
2933 3763 3
2934 3764 3      IF ( NOT .EF_STATUS) THEN SIGNAL_STOP (.EF_STATUS);
2935 3765 3
2936 3766 3      $GETDVI (EFN = .EF, CHAN = .TT_CHAN, DEVNAM = NAME_DESC, ITMLST = ITEM_LIST);
2937 3767 3      $WAITFR (EFN = .EF);
2938 3768 3
2939 3769 4      IF (.t_devclass EQL DC$_TERM)
2940 3770 3      THEN
2941 3771 4          BEGIN
2942 3772 4      +
2943 3773 4      | Get terminal width. If this is not a terminal, then the width has
2944 3774 4      | previously been initialized to '80'
2945 3775 4      |
2946 3776 4          EDT$$G_TI_WID = .T_DEVWID;
2947 3777 4      +
2948 3778 4      | Get terminal type.
2949 3779 4      |
2950 3780 4
2951 3781 4      SELECTONE .t_devtype OF
2952 3782 4      SET
2953 3783 4
2954 3784 4      [DTS_VT52, DTS_VT55] :
2955 3785 5          BEGIN
2956 3786 5          EDT$$G_TI_TYP = TERM_VT52;
2957 3787 5          EDT$$G_TI_SCROLL = 0;
2958 3788 5          EDT$$G_ENB_AUTRPT = 0;
2959 3789 5          EDT$$G_TI_EDIT = 0;
2960 3790 5          EDT$$G_TI_DUMB = 0;
2961 3791 4          END;
2962 3792 4
2963 3793 4      [DTS_VT100, DTS_VT125] :
2964 3794 5          BEGIN
2965 3795 5          EDT$$G_TI_TYP = TERM_VT100;
2966 3796 5          EDT$$G_TI_SCROLL = 1;
2967 3797 5          EDT$$G_ENB_AUTRPT = 1;
2968 3798 5          EDT$$G_TI_EDIT = .t_devextdep [TT2$V_EDIT];
2969 3799 5          EDT$$G_TI_DUMB = 0;
2970 3800 5          edt$$g_dec_crt = 1
2971 3801 4          END;
2972 3802 4
2973 3803 4      [OTHERWISE] :
2974 3804 5          BEGIN
2975 3805 5          EDT$$G_TI_TYP = TERM_HCPY; ! In case this is not a CRT
2976 3806 5          EDT$$G_TI_SCROLL = 0;
2977 3807 5          EDT$$G_ENB_AUTRPT = 0;
2978 3808 5          EDT$$G_TI_EDIT = 0;
2979 3809 5      +
2980 3810 5      | The terminal driver has extended characteristics if any bit is set in
2981 3811 5      | T_DEVEXTDEP. If this is a CRT with at least VT100 functionality, treat
2982 3812 5      | it as a VT100.
2983 3813 5      |
2984 3814 5
2985 3815 6          IF (.T_DEVDEP [TT$V_SCOPE] ND .t_devextdep [TT2$V_ANSICRT])
2986 3816 5          THEN
2987 3817 6              BEGIN
2988 3818 6              EDT$$G_TI_TYP = TERM_VT_10;
2989 3819 6              EDT$$G_TI_SCROLL = .t_devextdep [TT2$V_DECCRT];
```

```

2990 3820 6          EDT$G_TI_EDIT = .t_devextd p [TT2$V_EDIT];
2991 3821 6 !+
2992 3822 6 !- Set the terminal 'service class' specifier.
2993 3823 6 !-
2994 3824 6
2995 3825 6 %IF %DECLARED(%QUOTE TT2$V_DECCRT2) %THEN
2996 3826 6         IF .t_devextdep [TT2$V_DECCR 2] THEN
2997 3827 6             edt$g_dec_crt = 2
2998 3828 6         ELSE
2999 3829 6 %FI
3000 3830 6             edt$g_dec_crt = 1
3001 3831 5         END;
3002 3832 5
3003 3833 5         EDT$G_TI_DUMB = 0;
3004 3834 4         END;
3005 3835 4         TES;
3006 3836 4
3007 3837 4 !+
3008 3838 4 !- Set the eight-bit flag and store the page length.
3009 3839 4 !-
3010 3840 4         EDT$G_EIGHT_BIT = .T DEVDEP [TT$V_EIGHTBIT];
3011 3841 4         EDT$G_TI_PLEN = .T_DEVDEP [TT$V_PAGE];
3012 3842 4 !+
3013 3843 4 !- Set LOWER attribute and disable escape sequences if not already done
3014 3844 4 !-
3015 3845 4
3016 3846 5         IF (.T_DEVDEP [TT$V_ESCAPE] OR ( NOT .T_DEVDEP [TT$V_LOWER]))
3017 3847 4         THEN
3018 3848 5             BEGIN
3019 3849 5                 TT_INFO [S_CLASS] = .t_devclass;
3020 3850 5                 TT_INFO [S_TYPE] = .t_devtype;
3021 3851 5                 TT_INFO [S_WID] = .T DEVWID;
3022 3852 5                 TT_INFO [S_CHAR] = .T DEVDEP;
3023 3853 5                 BLOCK [TT_INFO [S_CHAR], TT$V_ESCAPE] = 0;
3024 3854 5                 BLOCK [TT_INFO [S_CHAR], TT$V_LOWER] = 1;
3025 3855 5 !
3026 3856 5                 QIO_STATUS = $QIOW (EFN = .EF, CHAN = .TT_CHAN, FUNC = IO$_SETMODE, P1 = TT_INFO);
3027 3857 5
3028 3858 5                 IF ( NOT .QIO_STATUS) THEN SIGNAL_STOP (.QIO_STATUS);
3029 3859 5
3030 3860 5                 EDT$G_TI_RESET = 1;
3031 3861 4                 END;
3032 3862 4
3033 3863 4 !+
3034 3864 4 !- Free up the event flag
3035 3865 4 !-
3036 3866 4                 EF_STATUS = LIB$FREE_EF (EF);
3037 3867 4
3038 3868 4                 IF ( NOT .EF_STATUS) THEN SIGNAL_STOP (.EF_STATUS);
3039 3869 4
3040 3870 4 !+
3041 3871 4 !- Enable control-c handling.
3042 3872 4 !-
3043 3873 4                 CC_ENABLE ();
3044 3874 4                 END
3045 3875 3             ELSE
3046 3876 3                 EDT$G_TI_TYP = TERM_UNKNOWN;
```

```
3047 3877 3
3048 3878 2      END;
3049 3879 2
3050 3880 2
3051 3881 2      +
3052 3882 2      | Write second startup record.
3053 3883 2      |
3054 3884 2      | $STRACE (EDT$STR_START, EDT$STR_SSTART_TERM, %UPVAL, EDT$$G_TI_TYP);
3055 3885 1      | RETURN (1);
      | END;
      | ! of routine EDT$$TI_OPN
```

```
54 54 55 50 4E 49 24 53 59 53 00E44 P.AAE: .ASCII \SYSS$INPUT\
54 55 50 54 55 4F 24 53 59 53 00E4D P.AAF: .ASCII \SYSS$OUTPUT\
00 00 00 54 55 50 4E 49 24 53 59 53 00E57 .BLKB 1
00 00 00 54 55 50 4E 49 24 53 59 53 00E58 P.AAG: .ASCII \SYSS$INPUT\<0><0><0>

$RMS_PTR=      TERM IN RAB
$RMS_PTR=      TERM OUT RAB
      .EXTRN SYS$STRNLOG, SYSS$ASSIGN
      .EXTRN SYSS$GETDVI, SYSS$WAITFR

      OFFC 00000      .ENTRY EDT$$TI_OPN, Save R2,R3,R4,R5,R6,R7,R8,R9,-
      5B 00000000G 00 9E 00002      MOVAB EDT$$G_TI_EDIT, R11
      5A 00000000G 00 9E 00009      MOVAB EDT$$G_TI_SCROLL, R10
      59 00000000G 00 9E 00010      MOVAB EDT$$G_DEC CRT, R9
      58 00000000G 00 9E 00017      MOVAB EDT$$G_TI_TYP, R8
      57 00000000G 00 9E 0001E      MOVAB LIB$STOP, R7
      56 00000000' EF 9E 00025      MOVAB EDT$$G_IO_STS, R6
      5E FF00 CE 9E 0002C      MOVAB -256(SP), -SP
      03 FE80 C6 E9 00031      BLBC TT_OPEN, 1$
      0356 31 00036      BRW 27$
      0050 8F      00 FE80 C6 01 D0 00039 1$:      MOVL #1, TT_OPEN
      6E 00 2C 0003E      MOVCS #0, (SP), #0, #80, $RMS_PTR
      04 AE 5003 8F B0 00047      MOVW #20483, $RMS_PTR
      1A AE 02 90 0004D      MOVW #2, $RMS_PTR+22
      22 AE 0202 8F B0 00051      MOVW #514, $RMS_PTR+30
      30 AE 86 AF 9E 00057      MOVAB P.AAE, WORK_FAB+44
      38 AE 09 90 0005C      MOVW #9, WORK_FAB+52
      0044 8F      00 6E 00 2C 00060      MOVCS #0, (SP), #0, #68, $RMS_PTR
      FDEC C6 00067      MOVW #17409, $RMS_PTR
      4401 8F B0 0006A      CLRB $RMS_PTR+30
      FE0A C6 94 00071      MOVZBW #255, $RMS_PTR+32
      FE0C C6 FF 8F 9B 00075      MOVAB MYBUF, $RMS_PTR+36
      FE10 C6 FA10 C6 9E 0007B      MOVAB VFCHD, $RMS_PTR+44
      FE18 C6 FEFC C6 9E 00082      MOVAB WORK_FAB, $RMS_PTR+60
      FE28 C6 04 AE 9E 00089      MOVAB WORK_FAB
      04 AE 9F 0008F      PUSHAB WORK_FAB
      00000000G 00 01 FB 00092      CALLS #1, SYSS$OPEN
      66 50 D0 00099      MOVL R0, EDT$$G_IO_STS
      19 50 E9 0009C      BLBC R0, 2$
      FDEC C6 9F 0009F      PUSHAB TERM IN RAB
      00000000G 00 01 FB 000A3      CALLS #1, SYSS$CONNECT
      66 50 D0 000AA      MOVL R0, EDT$$G_IO_STS
      1E 66 E8 000AD      BLBS EDT$$G_IO_STS, 4$
```

; R
; 33658
3659
3660
3662
3665
3667

0050	8F	00	04	A6	FDF8	C6	D0	000B0	MOVL	TERM_IN_RAB+12, EDT\$G_IO_STV	3662
			05	11	000B6	05	11	000B6	BRB	3\$	3671
			04	A6	10	AE	D0	000B8	2\$: MOVL	WORK_FAB+12, EDT\$G_IO_STV	3673
			0E			66	E8	000BD	3\$: BLBS	EDT\$G_IO_STS, 4\$	3676
			7E			66	7D	000C0	MOVQ	EDT\$G_IO_STS, -(SP)	
						7E	D4	000C3	CLRL	-(SP)	
					00000000G	8F	DD	000C5	PUSHL	#EDT\$ OPNINTRM	
			67			04	FB	000CB	CALLS	#4, LIB\$STOP	
			6E			00	2C	000CE	4\$: MOVCS	#0, (SP), #0, #80, \$RMS_PTR	3682
					04	AE		000D5			
			04	AE	5003	8F	B0	000D7	MOVW	#20483, \$RMS_PTR	
			1A	AE		01	90	000DD	MOVW	#1, \$RMS_PTR+22	
			22	AE	0202	8F	B0	000E1	MOVW	#514, \$RMS_PTR+30	
			30	AE	FEFE	CF	9E	000E7	MOVAB	P.AAF, WORK_FAB+44	3683
			38	AE		0A	90	000ED	MOVW	#10, WORK_FAB+52	3684
0044	8F	00	6E			00	2C	000F1	MOVCS	#0, (SP), #0, #68, \$RMS_PTR	3685
					FE30	C6		000F8			
			FE30	C6	4401	8F	B0	000FB	MOVW	#17409, \$RMS_PTR	
					FE4E	C6	94	00102	CLRB	\$RMS_PTR+30	
			FE50	C6	FF	8F	9B	00106	MOVZBW	#255, \$RMS_PTR+32	
			FE54	C6	FA10	C6	9E	0010C	MOVAB	MYBUF, \$RMS_PTR+36	
			FE5C	C6	FEFC	C6	9E	00113	MOVAB	VFCHD, \$RMS_PTR+44	
			FE6C	C6	04	AE	9E	0011A	MOVAB	WORK_FAB, \$RMS_PTR+60	
					04	AE	9F	00120	PUSHAB	WORK_FAB	3687
			00000000G	00		01	FB	00123	CALLS	#1, SYSS\$CREATE	
				66		50	D0	0012A	MOVL	R0, EDT\$G_IO_STS	
				19		50	E9	0012D	BLBC	R0, 5\$	
					FE30	C6	9F	00130	PUSHAB	TERM_OUT_RAB	3690
			00000000G	00		01	FB	00134	CALLS	#1, SYSS\$CONNECT	
				66		50	D0	0013B	MOVL	R0, EDT\$G_IO_STS	
				1E		66	E8	0013E	BLBS	EDT\$G_IO_STS, 7\$	3692
			04	A6	FE3C	C6	D0	00141	MOVL	TERM_OUT_RAB+12, EDT\$G_IO_STV	
						05	11	00147	BRB	6\$	3687
			04	A6	10	AE	D0	00149	5\$: MOVL	WORK_FAB+12, EDT\$G_IO_STV	3696
			0E			66	E8	0014E	6\$: BLBS	EDT\$G_IO_STS, 7\$	3698
			7E			66	7D	00151	MOVQ	EDT\$G_IO_STS, -(SP)	
						7E	D4	00154	CLRL	-(SP)	
					00000000G	8F	DD	00156	PUSHL	#EDT\$ OPNOUTTRM	
			67			04	FB	0015C	CALLS	#4, LIB\$STOP	
						69	D4	0015F	7\$: CLRL	EDT\$G_DEC CRT	3703
			5C	AE		09	D0	00161	MOVL	#9, NAME_DESC	3704
			60	AE	64	AE	9E	00165	MOVAB	NAME_BUF, NAME_DESC+4	3705
64	AE		FE85	CF		09	28	0016A	MOVCS	#9, P.AAG, NAME_BUF	3706
			58	AE	64	AE	9E	00171	MOVAB	NAME_BUF, RES_DESC+4	3707
			54	AE	40	8F	9A	00176	MOVZBL	#64, RES_DESC	3708
						7E	7C	0017B	8\$: CLRQ	-(SP)	3710
						7E	D4	0017D	CLRL	-(SP)	
					60	AE	9F	0017F	PUSHAB	RES_DESC	
					6C	AE	9F	00182	PUSHAB	NAME_DESC	
					70	AE	9F	00185	PUSHAB	NAME_DESC	
			00000000G	00		06	FB	00188	CALLS	#6, SYSS\$TRNLOG	
				01		50	D1	0018F	CMPL	R0, #1	
						27	12	00192	BNEQ	11\$	
				1B	64	AE	91	00194	CMPB	NAME_BUF, #27	3713
						0B	12	00198	BNEQ	9\$	
			60	AE	68	AE	9E	0019A	MOVAB	NAME_BUF+4, NAME_DESC+4	3716
			5C	AE		04	C2	0019F	SUBL2	#4, NAME_DESC	3717

50	60	AE	64	AE	9E	001A3	BRB	10\$	3713
	60	AE	5C	AE	9E	001A5	MOVAB	NAME_BUF, NAME_DESC+4	3720
		3A	FF	AE	C1	001AA	ADDL3	NAME_DESC, NAME_DESC+4, R0	3722
				AO	91	001B0	CMPB	-1(R0), #58	
			5C	AE	12	001B4	BNEQ	8\$	
				AE	D7	001B6	DECL	NAME_DESC	3724
	60	AE	64	AE	11	001B9	BRB	8\$	3710
				AE	9E	001BB	MOVAB	NAME_BUF, NAME_DESC+4	3728
				7E	7C	001C0	CLRQ	-(SP)	3730
			FE7C	C6	9F	001C2	PUSHAB	TT_CHAN	
			68	AE	9F	001C6	PUSHAB	NAME_DESC	
00000000G	00			04	FB	001C9	CALLS	#4, SYSS\$ASSIGN	
	03			50	E8	001D0	BLBS	R0, 12\$	
				01B9	31	001D3	BRW	27\$	
				50	D4	001D6	CLRL	I	3739
51	50			OC	C5	001D8	MULL3	#12, I, R1	3741
			AC	AD41	9F	001DC	PUSHAB	ITEM_LIST[R1]	
	9E			04	B0	001E0	MOVW	#4, 3(SP)+	
			B4	AD41	9F	001E3	PUSHAB	ITEM_LIST+8[R1]	3742
				9E	D4	001E7	CLRL	@(SP)+	
EB	50			05	F3	001E9	AOBLEQ	#5, I, 13\$	3739
	AE	AD		02	B0	001ED	MOVW	#2, ITEM_LIST+2	3746
	B0	AD	FE84	C6	9E	001F1	MOVAB	T_DEVCHAR, ITEM_LIST+4	3747
	BA	AD		04	B0	001F7	MOVW	#4, ITEM_LIST+14	3748
	BC	AD	FE88	C6	9E	001FB	MOVAB	T_DEVCLASS, ITEM_LIST+16	3749
	C6	AD		0A	B0	00201	MOVW	#10, ITEM_LIST+28	3750
	C8	AD	FE8C	C6	9E	00205	MOVAB	T_DEVDEP, ITEM_LIST+28	3751
	D2	AD		1C	B0	0020B	MOVW	#28, ITEM_LIST+38	3752
	D4	AD	FE90	C6	9E	0020F	MOVAB	T_DEVEXTDEP, ITEM_LIST+40	3753
	DE	AD		06	B0	00215	MOVW	#6, ITEM_LIST+50	3754
	E0	AD	FE94	C6	9E	00219	MOVAB	T_DEVTYPE, ITEM_LIST+52	3755
	EA	AD		08	B0	0021F	MOVW	#8, ITEM_LIST+62	3756
	EC	AD	FE98	C6	9E	00223	MOVAB	T_DEVWID, ITEM_LIST+64	3757
			F6	AD	B4	00229	CLRQ	ITEM_LIST+74	3758
				5E	DD	0022C	PUSHL	SP	3762
00000000G	00			01	FB	0022E	CALLS	#1, LIB\$GET_EF	
	52			50	D0	00235	MOVL	R0, EF_STATUS	
	05			52	E8	00238	BLBS	EF_STATUS, 14\$	3764
				52	DD	0023B	PUSHL	EF_STATUS	
	67			01	FB	0023D	CALLS	#1, LIB\$STOP	
				7E	7C	00240	CLRQ	-(SP)	3766
				7E	7C	00242	CLRQ	-(SP)	
			AC	AD	9F	00244	PUSHAB	ITEM_LIST	
			70	AE	9F	00247	PUSHAB	NAME_DESC	
			FE7C	C6	DD	0024A	PUSHL	TT_CHAN	
			1C	AE	DD	0024E	PUSHL	EF	
00000000G	00			08	FB	00251	CALLS	#8, SYSS\$GETDVI	
				6E	DD	00258	PUSHL	EF	3767
00000000G	00			01	FB	0025A	CALLS	#1, SYSS\$WAITFR	
00000042	8F		FE88	C6	D1	00261	CMPL	T_DEVCLASS, #66	3769
				03	13	0026A	BEQL	15\$	
				011E	31	0026C	BRW	26\$	
00000000G	00		FE98	C6	D0	0026F	MOVL	T_DEVWID, EDT\$G_TI_WID	3776
	50		FE94	C6	D0	00278	MOVL	T_DEVTYPE, R0	3781
	3F			50	D1	0027D	CMPL	R0, #63	3784
				18	15	00280	BLEQ	16\$	
00000041	8F			50	D1	00282	CMPL	R0, #65	

; R
; 3

				68	00000000G	0F 14 00289	BGTR	16\$		
				01		D0 0028B	MOVL	#1, EDT\$SG_TI_TYP	3786	
				6A		D4 0028E	CLRL	EDT\$SG_TI_SCROLL	3787	
				00		D4 00290	CLRL	EDT\$SG_ENB_AURPT	3788	
				6B		D4 00296	CLRL	EDT\$SG_TI_EDIT	3789	
				68		11 00298	BRB	20\$	3790	
				50		D1 0029A	16\$:	CMPL R0, #96	3793	
				09		13 002A1	BEQL	17\$		
				50		D1 002A3	CMPL	R0, #100		
				1F		12 002AA	BNEQ	18\$		
				02		D0 002AC	17\$:	MOVL #2, EDT\$SG_TI_TYP	3795	
				01		D0 002AF	MOVL	#1, EDT\$SG_TI_SCROLL	3796	
				01		D0 002B2	MOVL	#1, EDT\$SG_ENB_AURPT	3797	
				04		EF 002B9	EXTZV	#4, #1, T_DEVEXTDEP+3, EDT\$SG_TI_EDIT	3798	
				00		D4 002C0	CLRL	EDT\$SG_TI_DUMB	3799	
				01		D0 002C6	MOVL	#1, EDT\$SG_DEC_CRT	3800	
				3D		11 002C9	BRB	21\$		
				03		D0 002CB	18\$:	MOVL #3, EDT\$SG_TI_TYP	3805	
				6A		D4 002CE	CLRL	EDT\$SG_TI_SCROLL	3806	
				00		D4 002D0	CLRL	EDT\$SG_ENB_AURPT	3807	
				6B		D4 002D6	CLRL	EDT\$SG_TI_EDIT	3808	
				04		E1 002D8	BBC	#4, T_DEVDEP+1, 20\$	3815	
				02		D0 002DE	BLBC	T_DEVEXTDEP+3, 20\$		
				05		EF 002E6	EXTZV	#5, #1, T_DEVEXTDEP+3, EDT\$SG_TI_SCROLL	3818	
				04		EF 002ED	EXTZV	#4, #1, T_DEVEXTDEP+3, EDT\$SG_TI_EDIT	3819	
				06		E1 002F4	BBC	#6, T_DEVEXTDEP+3, 19\$	3820	
				02		D0 002FA	MOVL	#2, EDT\$SG_DEC_CRT	3826	
				03		11 002FD	BRB	20\$	3827	
				01		D0 002FF	19\$:	MOVL #1, EDT\$SG_DEC_CRT	3830	
				00		D4 00302	20\$:	CLRL EDT\$SG_TI_DUMB	3833	
				07		EF 00308	21\$:	EXTZV #7, #1, T_DEVDEP+1, EDT\$SG_EIGHT_BIT	3840	
				06		9A 00313	MOVZBL	T_DEVDEP+3, EDT\$SG_TI_PLEN	3841	
				03		E0 0031C	BBS	#3, T_DEVDEP, 22\$	3846	
				06		95 00322	TSTB	T_DEVDEP		
				4A		19 00326	BLSS	24\$		
				06		90 00328	22\$:	MOVB T_DEVCLASS, TT_INFO	3849	
				50		90 0032E	MOVB	R0, TT_INFO+1	3850	
				06		B0 00332	MOVW	T_DEVWID, TT_INFO+2	3851	
				06		D0 00338	MOVL	T_DEVDEP, TT_INFO+4	3852	
				08		8A 0033E	BICB2	#8, TT_INFO+4	3853	
				8F		88 00342	BISB2	#128, TT_INFO+4	3854	
				7E		7C 00347	CLRQ	-(SP)	3856	
				7E		7C 00349	CLRQ	-(SP)		
				7E		D4 0034B	CLRL	-(SP)		
				AD		9F 0034D	PUSHAB	TT_INFO		
				7E		7C 00350	CLRQ	-(SP)		
				23		7D 00352	MOVQ	#35, -(SP)		
				06		DD 00355	PUSHL	TT_CHAN		
				AE		DD 00359	PUSHL	EF		
				0C		FB 0035C	CALLS	#12, SYS\$QIOW		
				50		E8 00363	BLBS	QIO_STATUS, 23\$	3858	
				50		DD 00366	PUSHL	QIO_STATUS		
				01		FB 00368	CALLS	#1, LIB\$STOP		
				01		D0 0036B	23\$:	MOVL #1, EDT\$SG_TI_RESET	3860	
				5E		DD 00372	24\$:	PUSHL SP	3866	
				01		FB 00374	CALLS	#1, LIB\$FREE_EF		

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$TI_OPN - open the terminal

N 2
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 94
(28)

	52	50	D0 00378	MOVL	R0, EF_STATUS		
	05	52	E8 0037E	BLBS	EF_STATUS, 25\$	:	3868
		52	DD 00381	PUSHL	EF_STATUS	:	
	67	01	FB 00383	CALLS	#1, LIB\$STOP	:	
FBDB	CF	00	FB 00386	CALLS	#0, CC_ENABLE	:	3873
		02	11 00388	BRB	27\$	:	3769
		68	D4 0038D	CLRL	EDT\$\$G-TI_TYP	:	3876
	50	01	D0 0038F	MOVL	#1, R0	:	3884
		04	00392	RET		:	3885

; Routine Size: 915 bytes, Routine Base: _EDT\$CODE + 0E64

; 3056 3886 1

EDT\$
V04-

FFF1

```
3058 3887 1 %SBTTL 'EDT$STI_RES - restore the terminal'
3059 3888 1
3060 3889 1 GLOBAL ROUTINE EDT$STI_RES ! Restore the terminal
3061 3890 1 =
3062 3891 1
3063 3892 1 ++
3064 3893 1 FUNCTIONAL DESCRIPTION:
3065 3894 1
3066 3895 1 This routine restores the terminal characteristics to what t y
3067 3896 1 were when ED* was entered.
3068 3897 1
3069 3898 1 FORMAL PARAMETERS:
3070 3899 1
3071 3900 1 NONE
3072 3901 1
3073 3902 1 IMPLICIT INPUTS:
3074 3903 1
3075 3904 1 T_DEVCHAR
3076 3905 1 T_DEVCLASS
3077 3906 1 T_DEVDEP
3078 3907 1 T_DEVTYPE
3079 3908 1 T_DEVWID
3080 3909 1
3081 3910 1 IMPLICIT OUTPUTS:
3082 3911 1
3083 3912 1 NONE
3084 3913 1
3085 3914 1 ROUTINE VALUE:
3086 3915 1
3087 3916 1 0 = unsuccessful
3088 3917 1 1 = successful
3089 3918 1
3090 3919 1 SIDE EFFECTS:
3091 3920 1
3092 3921 1 Changes the logical terminal characteristics
3093 3922 1
3094 3923 1 --
3095 3924 1
3096 3925 2 BEGIN
3097 3926 2
3098 3927 2 LOCAL
3099 3928 2 QIO_STATUS,
3100 3929 2 EF_STATUS,
3101 3930 2 EF;
3102 3931 2 TT_INFO : BLOCK [2] FIELD (SETM_FIELDS); ! Terminal characteristics
3103 3932 2
3104 3933 2 ++
3105 3934 2 If SYSS$INPUT is not a terminal, do nothing.
3106 3935 2 --
3107 3936 2
3108 3937 2 IF (.t_devclass NEQ DC$_TERM) THEN RETURN (1);
3109 3938 2
3110 3939 2 TT_INFO [S_CLASS] = .t_devclass;
3111 3940 2 TT_INFO [S_TYPE] = .t_devtype;
3112 3941 2 TT_INFO [S_WID] = .T_DEVWID;
3113 3942 2 TT_INFO [S_CHAR] = .T_DEVDEP;
3114 3943 2 ++
```

: Rc

: 34

```
3115 3944 2 ! Get an event flag
3116 3945 2 !-
3117 3946 2 EF_STATUS = LIB$GET_EF (EF);
3118 3947 2
3119 3948 2 IF ( NOT .EF_STATUS) THEN SIGNAL_STOP (.EF_STATUS);
3120 3949 2
3121 3950 2 QIO_STATUS = $QIOW (EFN = .EF, CHAN = .TT_CHAN, FUNC = IOS_SETMODE, P1 = TT_INFO);
3122 3951 2
3123 3952 2 !+ Free up the event flag
3124 3953 2 !-
3125 3954 2 EF_STATUS = LIB$FREE_EF (EF);
3126 3955 2
3127 3956 2 IF ( NOT .EF_STATUS) THEN SIGNAL_STOP (.EF_STATUS);
3128 3957 2
3129 3958 2 IF (.QIO_STATUS) THEN RETURN (1) ELSE RETURN (0);
3130 3959 2
3131 3960 1 END;
```

' of routine EDT\$STI_RES

			003C 00000	.ENTRY EDT\$STI_RES, Save R2,R3,R4,R5	3889
	55	00000000G	00 9E 00002	MOVAB LIB\$STOP, R5	
	54	00000000'	EF 9E 00009	MOVAB T_DEVCLASS, R4	
	5E		0C C2 00010	SUBL2 #T2, SP	
00000042	8F		64 D1 00013	CMPL T_DEVCLASS, #66	3937
			5C 12 0001A	BNEQ 35	
04	AE		64 90 0001C	MOVAB T_DEVCLASS, TT_INFO	3939
05	AE	0C	A4 90 00020	MOVAB T_DEVTYPE, TT_INFO+1	3940
06	AE	10	A4 B0 00025	MOVW T_DEVWID, TT_INFO+2	3941
08	AE	04	A4 D0 0002A	MOVL T_DEVDEP, TT_INFO+4	3942
			5E DD 0002F	PUSHL SP	3946
00000000G	00		01 FB 00031	CALLS #1, LIB\$GET_EF	
	52		50 D0 00038	MOVL R0, EF_STATUS	
	05		52 E8 0003B	BLBS EF_STATUS, 15	3948
			52 DD 0003E	PUSHL EF_STATUS	
	65		01 FB 00040	CALLS #1, LIB\$STOP	
			7E 7C 00043	CLRQ -(SP)	3950
			7E 7C 00045	CLRQ -(SP)	
			7E D4 00047	CLRL -(SP)	
		18	AE 9F 00049	PUSHAB TT_INFO	
			7E 7C 0004C	CLRQ -(SP)	
	7E		23 7D 0004E	MOVQ #35, -(SP)	
		F4	A4 DD 00051	PUSHL TT_CHAN	
		2C	AE DD 00054	PUSHL EF	
00000000G	00		0C FB 00057	CALLS #12, SYS\$QIOW	
	53		50 D0 0005E	MOVL R0, QIO_STATUS	
			5E DD 00061	PUSHL SP	3954
00000000G	00		01 FB 00063	CALLS #1, LIB\$FREE_EF	
	52		50 D0 0006A	MOVL R0, EF_STATUS	
	05		52 E8 0006D	BLBS EF_STATUS, 25	3956
			52 DD 00070	PUSHL EF_STATUS	
	65		01 FB 00072	CALLS #1, LIB\$STOP	
	04		53 E9 00075	BLBC QIO_STATUS, 45	3958
	50		01 D0 00078	MOVL #1, R0	
			04 0007B	RET	

```
EDT$!OMOD - I/O for VAX/VMS
EDT$!I_RES - restore the terminal
```

D 3
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742 Page 97
DISK\$VMSMASTER:[EDIT.SRC]IOMOD.B32;1 (29)

EDTS
V04-

```
50 D4 0007C 48: CLRL R0
    04 0007E RET
```

3960

```
; Routine Size: 127 bytes,    Routine Base: _EDT$CODE + 11F7
```

: 3132 3961 1

: Rc

35

```
3134 3962 1 %SBTTL 'EDT$$TI_DEAS - deassign the terminal'
3135 3963 1
3136 3964 1 GLOBAL ROUTINE EDT$$TI_DEAS ! Deassign the terminal channel
3137 3965 1 : NOVALUE =
3138 3966 1
3139 3967 1 ++
3140 3968 1 FUNCTIONAL DESCRIPTION:
3141 3969 1
3142 3970 1 This routine deassigns the terminal channel and restores OWN storage.
3143 3971 1
3144 3972 1 FORMAL PARAMETERS:
3145 3973 1
3146 3974 1 NONE
3147 3975 1
3148 3976 1 IMPLICIT INPUTS:
3149 3977 1
3150 3978 1 TT_CHAN
3151 3979 1 EDT$$L_TR_INFLG
3152 3980 1 T_DEVCLASS
3153 3981 1 T_DEVDEP
3154 3982 1 T_DEVEXTDEP
3155 3983 1 T_DEVTYPE
3156 3984 1
3157 3985 1 IMPLICIT OUTPUTS:
3158 3986 1
3159 3987 1 ATTRIBUTES_SET
3160 3988 1 ATTRIBUTES_DEFAULTED
3161 3989 1 TT_OPEN
3162 3990 1
3163 3991 1 ROUTINE VALUE:
3164 3992 1
3165 3993 1 0 = unsuccessful
3166 3994 1 1 = successful
3167 3995 1
3168 3996 1 SIDE EFFECTS:
3169 3997 1
3170 3998 1 Sets application keypad mode or numeric keypad mode
3171 3999 1 Deassigns the terminal channel
3172 4000 1
3173 4001 1 --
3174 4002 1
3175 4003 2 BEGIN
3176 4004 2
3177 4005 2 %IF edt$$tr_act
3178 4006 2 %THEN
3179 4007 2
3180 4008 2 EXTERNAL
3181 4009 2 EDT$$L_TR_INFLG; ! 1 iff input from trace file
3182 4010 2
3183 4011 2 %FI
3184 4012 2
3185 4013 2 LOCAL
3186 4014 2 STATUS,
3187 4015 2 term_set;
3188 4016 2
3189 4017 2 !+
3190 4018 2 ! Set either application keypad mode or numeric keypad mode depending on the user's stated
```

```
3191 4019 2 | preference. This only applies to VT100 and VT52 terminals.
3192 4020 2 |
3193 4021 2 |   term_set = 1;
3194 4022 2 |
3195 4023 2 | %IF edt$$tr_act
3196 4024 2 | %THEN
3197 4025 2 |
3198 4026 2 |   IF .edt$$l_tr_inflg THEN term_set = 0;
3199 4027 2 |
3200 4028 2 | %FI
3201 4029 2 |
3202 4030 2 |   IF (.t_devclass NEQ DC$_TERM) THEN term_set = 0;
3203 4031 2 |
3204 4032 2 |   IF ( NOT .t_devdep [tt$_scope]) THEN term_set = 0;
3205 4033 2 |
3206 4034 4 |   IF ( NOT ( .t_devextdep [TT2$_DECRT] OR
3207 4035 4 |               (.t_devtype EQL DT$_VT52) OR
3208 4036 2 |               (.t_devtype EQL DT$_VT55) ) ) THEN term_set = 0;
3209 4037 2 |
3210 4038 2 |   IF .term_set
3211 4039 2 |   THEN
3212 4040 2 | %IF %DECLARED(%QUOTE TT2$_APP_KEYPAD) %THEN
3213 4041 2 |   IF .t_devextdep [TT2$_APP_KEYPAD]
3214 4042 2 |   THEN
3215 4043 2 |     EDT$$TI_WRSTR (UPLIT (BYTE (ASC_K_ESC, '=')), 2)
3216 4044 2 |   ELSE
3217 4045 2 | %FI
3218 4046 2 |     EDT$$TI_WRSTR (UPLIT (BYTE (ASC_K_ESC, '>')), 2);
3219 4047 2 |
3220 4048 2 | + Make sure all of this module's OWN storage is back to its initial state
3221 4049 2 | in case we get called again
3222 4050 2 | -
3223 4051 2 |   ATTRIBUTES_SET = 0;
3224 4052 2 |   ATTRIBUTES_DEFAULTED = 0;
3225 4053 2 | +
3226 4054 2 | - If we are taking input from the terminal, close the terminal's channel.
3227 4055 2 | -
3228 4056 2 |
3229 4057 2 | %IF edt$$tr_act
3230 4058 2 | %THEN
3231 4059 2 |
3232 4060 2 |   IF ( NOT .EDT$$L_TR_INFLG) THEN STATUS = $DASSGN (CHAN = .TT_CHAN) ELSE STATUS = 1;
3233 4061 2 |
3234 4062 2 | %ELSE
3235 4063 2 |   STATUS = $DASSGN (CHAN = .TT_CHAN);
3236 4064 2 | %FI
3237 4065 2 |
3238 4066 2 |   TT_OPEN = 0;
3239 4067 2 |
3240 4068 2 |   IF ( NOT .STATUS) THEN SIGNAL_STOP (.STATUS);
3241 4069 2 |
3242 4070 1 |   END;
                                     ! of routine EDT$$TI_DEAS
```

			3D	01279	.ASCII	\=\	:
				0127A	.BLKB	2	:
			1B	0127C	.BYTE	27	:
			3E	0127D	.ASCII	\>\	:
					.EXTRN	SYS\$DASSGN	
			0004	00000	.ENTRY	EDT\$STI DEAS, Save R2	: 3964
	52	00000000'	EF	9E 00002	MOVAB	T DEVEXTDEP, R2	:
	50		01	D0 00009	MOVL	#T, TERM SET	: 4021
00000042	8F	F8	A2	D1 0000C	CMPL	T DEVCLASS, #66	: 4030
			02	13 00014	BEQL	1\$	:
			50	D4 00016	CLRL	TERM SET	:
02	FD	A2	04	E0 00018	BBS	#4, T DEVDEP+1, 2\$	: 4032
			50	D4 0001D	CLRL	TERM SET	:
16	03	A2	05	E0 0001F	BBS	#5, T DEVEXTDEP+3, 3\$	: 4034
00000040	8F	04	A2	D1 00024	CMPL	T DEVTYPE, #64	: 4035
			0C	13 0002C	BEQL	3\$	:
00000041	8F	04	A2	D1 0002E	CMPL	T DEVTYPE, #65	: 4036
			02	13 00036	BEQL	3\$	:
			50	D4 00038	CLRL	TERM SET	:
	16		50	E9 0003A	BLBC	TERM SET, 6\$	: 4038
		02	A2	95 0003D	TSTB	T DEVEXTDEP+2	: 4041
			07	18 00040	BGEQ	4\$	:
			02	DD 00042	PUSHL	#2	: 4043
		B3	AF	9F 00044	PUSHAB	P.AAH	:
			05	11 00047	BRB	5\$	:
			02	DD 00049	PUSHL	#2	: 4046
		B0	AF	9F 0004B	PUSHAB	P.AAI	:
0000V	CF		02	FB 0004E	CALLS	#2, EDT\$STI WRSTR	:
		FE94	C2	7C 00053	CLRQ	ATTRIBUTES_SET	: 4051
		EC	A2	DD 00057	PUSHL	TT_CHAN	: 4063
00000000G	00		01	FB 0005A	CALLS	#1, SYS\$DASSGN	:
		F0	A2	D4 00061	CLRL	TT_OPEN	: 4066
	09		50	E8 00064	BLBS	STATUS, 7\$	: 4068
			50	DD 00067	PUSHL	STATUS	:
00000000G	00		01	FB 00069	CALLS	#1, LIB\$STOP	:
			04	00070	RET		: 4070

; Routine Size: 113 bytes, Routine Base: _EDT\$CODE + 127E

: 3243 4071 1

```

3245 4072 1 %SBTTL 'EDT$SRD_CMDLN - read a command line'
3246 4073 1
3247 4074 1 GLOBAL ROUTINE EDT$SRD_CMDLN (
3248 4075 1     PROMPT,
3249 4076 1     PR_LEN,
3250 4077 1     BUF,
3251 4078 1     LEN,
3252 4079 1     BUF_LN
3253 4080 1 ) =
3254 4081 1
3255 4082 1
3256 4083 1 **
3257 4084 1 FUNCTIONAL DESCRIPTION:
3258 4085 1     Read a command line. A line is read from the terminal.
3259 4086 1
3260 4087 1 FORMAL PARAMETERS:
3261 4088 1
3262 4089 1     PROMPT           The address of the prompt string
3263 4090 1     PR_LEN          The length of the prompt string
3264 4091 1     BUF            The address of the receiving buffer
3265 4092 1     LEN            The address of a longword to receive the length read
3266 4093 1     BUF_LN         (optional) the length of the buffer, defaults to 255
3267 4094 1
3268 4095 1 IMPLICIT INPUTS:
3269 4096 1
3270 4097 1     TERM_IN_RAB
3271 4098 1
3272 4099 1 IMPLICIT OUTPUTS:
3273 4100 1
3274 4101 1     TERM_IN_RAB
3275 4102 1     IO_STS
3276 4103 1
3277 4104 1 ROUTINE VALUE:
3278 4105 1
3279 4106 1     Return value is 1 if EOF, 0 otherwise.
3280 4107 1
3281 4108 1 SIDE EFFECTS:
3282 4109 1
3283 4110 1     Calls $IO_ERRMSG for I/O errors.
3284 4111 1
3285 4112 1 --
3286 4113 1
3287 4114 2 BEGIN
3288 4115 2
3289 4116 2 BUILTIN
3290 4117 2     ACTUALCOUNT;
3291 4118 2
3292 4119 2 ASSERT (.EDT$G_RDAHED EQL 0);
3293 4120 2 IO_STS = 0;
3294 4121 2
3295 4122 2 WHILE ( NOT .IO_STS) DO
3296 4123 3 BEGIN
3297 4124 3     $TRACE (EDT$STR_KEY, EDT$STR_SK_PROMPT, .PR_LEN, .PROMPT);
3298 4125 3     TERM_IN_RAB [RAB$L_PBF] = .PROMPT;
3299 4126 3     TERM_IN_RAB [RAB$B_PSZ] = .PR_LEN;
3300 4127 3     TERM_IN_RAB [RAB$L_UBF] = .BUF;
3301 4128 3     TERM_IN_RAB [RAB$W_USZ] = 255;
```

```
3302 4129 3      TERM_IN_RAB [RAB$$_ROP] = RAB$$_PMT;
3303 4130 3
3304 4131 3      IF (ACTUALCOUNT () EQL 5) THEN TERM_IN_RAB [RAB$$_USZ] = .BUF_LN;
3305 4132 3
3306 4133 3      $$TRACE (EDT$$_STR_IDLE, EDT$$_STR_SIDE_KIN, 0, 0);
3307 4134 3
3308 4135 4      IF ( NOT $$TRACE_READ (EDT$$_STR_KEY, EDT$$_STR_SK_STS, %UPVAL, IO_STS, 0))
3309 4136 3      THEN
3310 4137 3          IO_STS = $RMS_GET (RAB = TERM_IN_RAB);
3311 4138 3
3312 4139 3      $$TRACE (EDT$$_STR_KEY, EDT$$_STR_SK_STS, %UPVAL, IO_STS);
3313 4140 3
3314 4141 4      IF ( NOT $$TRACE_READ (EDT$$_STR_KEY, EDT$$_STR_SK_LN, .TERM_IN_RAB [RAB$$_USZ], .BUF, .LEN))
3315 4142 3      THEN
3316 4143 3          .LEN = .TERM_IN_RAB [RAB$$_RSZ];
3317 4144 3
3318 4145 3      $$TRACE (EDT$$_STR_KEY, EDT$$_STR_SK_LN, ..LEN, .BUF);
3319 4146 3
3320 4147 3      IF (.IO_STS EQL RMSS_EOF) THEN RETURN (1);
3321 4148 3
3322 4149 4      IF ( NOT .IO_STS)
3323 4150 3      THEN
3324 4151 4          BEGIN
3325 4152 4              EDT$$_G_IO_STS = .IO_STS;
3326 4153 4              EDT$$_G_IO_STV = .TERM_IN_RAB [RAB$$_STV];
3327 4154 4              EDT$$_FMT MSG (EDT$$_ERRINPTRM);
3328 4155 4              EDT$$_IO_ERRMSG ();
3329 4156 3          END;
3330 4157 3
3331 4158 2      END;
3332 4159 2
3333 4160 2      RETURN (0);
3334 4161 1      END;
```

! of routine EDT\$\$_RD_CMDLN

			0004	00000	.ENTRY	EDT\$\$_RD_CMDLN, Save R2	4074
	52	00000000'	EF	9E 00002	MOVAB	IO_STS, R2	
		00000000G	00	D5 00009	TSTL	EDT\$\$_G_RDAHED	4119
			07	13 0000F	BEQL	1\$	
00000000G	00		00	FB 00011	CALLS	#0, EDT\$\$_INTER_ERR	
			62	D4 00018	CLRL	IO_STS	4120
	73		62	E8 0001A	BLBS	IO_STS, 5\$	4122
FF28	C2	04	AC	D0 0001D	MOVL	PROMPT, TERM_IN_RAB+48	4125
FF2C	C2	08	AC	90 00023	MOVB	PR_LEN, TERM_IN_RAB+52	4126
FF1C	C2	0C	AC	D0 00029	MOVL	BUF, TERM_IN_RAB+36	4127
FF18	C2	FF	8F	9B 0002F	MOVZBW	#255, TERM_IN_RAB+32	4128
FEFC	C2	40000000	8F	D0 00035	MOVL	#1073741824, TERM_IN_RAB+4	4129
	05		6C	91 0003E	CMPB	(AP), #5	4131
			06	12 00041	BNEQ	3\$	
FF18	C2	14	AC	B0 00043	MOVW	BUF_LN, TERM_IN_RAB+32	
		FEF8	C2	9F 00049	PUSHAB	TERM_IN_RAB	4137
00000000G	00		01	FB 0004D	CALLS	#1, SYS\$GET	
	62		50	D0 00054	MOVL	R0, IO_STS	
10	BC	FF1A	C2	3C 00057	MOVZWL	TERM_IN_RAB+34, @LEN	4143

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$RD_CMDLN - read a command line

J 3
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 103
(31)

EDT\$
V04-

0001827A	50	62	D0	0005D	MOVL	IO_STS, R0	:	4147
	8F	50	D1	00060	CMPL	R0, #98938	:	
		04	12	00067	BNEQ	4\$	:	
	50	01	D0	00069	MOVL	#1, R0	:	
		04	04	0006C	RET		:	
	AA	50	E8	0006D	4\$: BLBS	R0, 2\$	:	4149
010C	C2	50	D0	00070	MOVL	R0, EDT\$\$G_IO_STS	:	4152
0110	C2	L2	D0	00075	MOVL	TERM IN RAB+12, EDT\$\$G_IO_STV	:	4153
00000000G	00	8F	DD	0007C	PUSHL	#EDT\$ ERRINPTRM	:	4154
EC83	CF	01	FB	00082	CALLS	#1, EDT\$\$FMT MSG	:	
		00	FB	00089	CALLS	#0, EDT\$\$IO_ERRMSG	:	4155
		8A	11	0008E	BRB	2\$	:	4122
		50	D4	00090	5\$: CLRL	R0	:	4160
		04	00	00092	RET		:	4161

; Routine Size: 147 bytes, Routine Base: _EDT\$CODE + 12EF

; 3335 4162 1

0000

```

3337 4163 1 %SBTTL 'EDT$STI_WRLN - write a line to the terminal'
3338 4164 1
3339 4165 1 GLOBAL ROUTINE EDT$STI_WRLN (           ! Write a line to the terminal
3340 4166 1     BUF,                               ! Starting address of the line
3341 4167 1     LEN,                               ! Length of the line
3342 4168 1     ) : NOVALUE =
3343 4169 1
3344 4170 1
3345 4171 1 **
3346 4172 1 FUNCTIONAL DESCRIPTION:
3347 4173 1     This routine writes a line to the terminal.
3348 4174 1     If an error is detected, it is considered fatal, since
3349 4175 1     we are not communicating with the user.
3350 4176 1
3351 4177 1 FORMAL PARAMETERS:
3352 4178 1
3353 4179 1     BUF           the address of the buffer containing the line.
3354 4180 1     LEN           the length of the line to be written.
3355 4181 1
3356 4182 1 IMPLICIT INPUTS:
3357 4183 1
3358 4184 1     TERM_OUT_RAB
3359 4185 1
3360 4186 1 IMPLICIT OUTPUTS:
3361 4187 1
3362 4188 1     TERM_OUT_RAB
3363 4189 1     IO_STV, if there is an error
3364 4190 1
3365 4191 1 ROUTINE VALUE:
3366 4192 1
3367 4193 1     NONE
3368 4194 1
3369 4195 1 SIDE EFFECTS:
3370 4196 1
3371 4197 1     Calls EDT$$FATAL_IOERR on an I/O error.
3372 4198 1
3373 4199 1 --
3374 4200 1
3375 4201 2 BEGIN
3376 4202 2
3377 4203 2 LOCAL
3378 4204 2     STAT;
3379 4205 2
3380 4206 2     TERM_OUT_RAB [RAB$L_RBF] = .BUF;
3381 4207 2     TERM_OUT_RAB [RAB$W_RSZ] = .LEN;
3382 4208 2     $$TRACE (EDT$$TR_SCR, EDT$$TR_SSCR_LN, .LEN, .BUF);
3383 4209 2     $$TRACE_READING ?(RETURN (0))?;
3384 4210 2
3385 4211 2     IF (STAT = $RMS_PUT (RAB = TERM_OUT_RAB)) THEN RETURN;
3386 4212 2
3387 4213 2     EDT$$G_IO_STV = .TERM_OUT_RAB [RAB$L_STV];
3388 4214 2     EDT$$FATAL_IOERR (.STAT);
3389 4215 1 END;                                     ! of routine EDT$STI_WRLN
```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$TI_WRLN - write a line to the terminal

L 3
16-Sep-1984 00:35:33
14-Sep-1984 12:27:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 105
(32)

EDT\$
V04-

: 37

			0004 00000	.ENTRY	EDT\$\$TI_WRLN, Save R2	:	4165
	52	00000000'	EF 9E 00002	MOVAB	TERM_OUT_RAB+40, R2	:	
	62	04	AC D0 00009	MOVL	BUF, TERM_OUT_RAB+40	:	4206
FA	A2	08	AC B0 0000D	MOVW	LEN, TERM_OUT_RAB+34	:	4207
		D8	A2 9F 00012	PUSHAB	TERM_OUT_RAB	:	4211
00000000G	00		01 FB 00015	CALLS	#1, SYSSPUT	:	
	0D		50 E8 0001C	BLBS	STAT, 1\$	:	
01AC	C2	E4	A2 D0 0001F	MOVL	TERM_OUT_RAB+12, EDT\$\$G_IO_STV	:	4213
			50 DD 00025	PUSHL	STAT	:	4214
EC85	CF		01 FB 00027	CALLS	#1, EDT\$\$FATAL_IOERR	:	
			04 0002C 1\$:	RET		:	4215

: Routine Size: 45 bytes, Routine Base: _EDT\$CODE + 1382

: 3390 4216 1

```

3392 4217 1 %SBTTL 'EDT$STI_GETCH - Get one character'
3393 4218 1
3394 4219 1 GLOBAL ROUTINE EDT$STI_GETCH (           ; Get one character
3395 4220 1      C                                   ; Where to store character
3396 4221 1      ) : NOVALUE =
3397 4222 1
3398 4223 1 ++
3399 4224 1 FUNCTIONAL DESCRIPTION:
3400 4225 1      Read a single character from the terminal.
3401 4226 1
3402 4227 1 FORMAL PARAMETERS:
3403 4228 1
3404 4229 1      C                                   Where to store character read
3405 4230 1
3406 4231 1 IMPLICIT INPUTS:
3407 4232 1
3408 4233 1      TT_CHAN
3409 4234 1      INPUT_EFN
3410 4235 1      HAVE_EFNS
3411 4236 1      T_DEVCLASS
3412 4237 1
3413 4238 1 IMPLICIT OUTPUTS:
3414 4239 1
3415 4240 1      EDT$$G_RDAHED
3416 4241 1      EDT$ST_RDAHED
3417 4242 1
3418 4243 1 ROUTINE VALUE:
3419 4244 1
3420 4245 1      NONE
3421 4246 1
3422 4247 1 SIDE EFFECTS:
3423 4248 1
3424 4249 1      NONE
3425 4250 1
3426 4251 1 --
3427 4252 1
3428 4253 1 BEGIN
3429 4254 2
3430 4255 2 LOCAL
3431 4256 2      STS,
3432 4257 2      TMASK : VECTOR [2],
3433 4258 2      FUNC_VAL;
3434 4259 2
3435 4260 2 ASSERT (.EDT$$G_RDAHED EQL 0);
3436 4261 2
3437 4262 2 ++
3438 4263 2 If SYSS$INPUT is not a terminal, just return a control Z.
3439 4264 2 --
3440 4265 2
3441 4266 3 IF (.t_devclass NEQ DC$_TERM)
3442 4267 2 THEN
3443 4268 3 BEGIN
3444 4269 3     .C = ASC_K_CTRL_Z;
3445 4270 3     RETURN;
3446 4271 2     END;
3447 4272 2
3448 4273 2 ++
```

```

3449 4274 2 ! Make sure we have the event flags to drive the terminal.
3450 4275 2 !-
3451 4276 2
3452 4277 2 IF ( NOT .HAVE_EFNS) THEN EDT$STI_ENTERCHM ();
3453 4278 2
3454 4279 2 TMASK [0] = 32;
3455 4280 2 TMASK [1] = UPLIT (-1, -1, -1, -1, -1, -1, -1, -1);
3456 4281 2 $$TRACE (EDT$STR_IDLE, EDT$STR_SIDLE_KIN, 0, 0);
3457 4282 2 ASSERT (.HAVE_EFNS);
3458 4283 2
3459 4284 3 IF ( NOT $$TRACE_READ (EDT$STR_KEY, EDT$STR_SK_CH_NE, K_RDAHED_LEN, EDT$ST_RDAHED, EDT$SG_RDAHED)) !
3460 4285 2 THEN
3461 4286 2 DO
3462 4287 2 BEGIN
3463 4288 3 FUNC_VAL = IOS_READVBLK + IOSM_NOECHO + IOSM_NOFILTR + IOSM_TRMNOECHO;
3464 4289 3
3465 4290 3 IF (( NOT edt$str_act) AND (.EDT$ST_PMT_KPD [0] EQL 0)) THEN FUNC_VAL = .FUNC_VAL + IOSM_ESCAPE;
3466 4291 3
3467 4292 3 STS = $QIOW (EFN = .INPUT_EFN, CHAN = .TT_CHAN, P1 = EDT$ST_RDAHED, P2 = K_RDAHED_LEN, !
P 4293 3 P4 = TMASK, IOSB = IOSB, FUNC = .FUNC_VAL);
3469 4294 3
3470 4295 3 IF ( NOT .STS) THEN SIGNAL_STOP (EDT$ERRINPTRM, 0, .STS);
3471 4296 3
3472 4297 3 EDT$SG_RDAHED = .IOSB [1] + .IOSB [3];
3473 4298 3
3474 4299 3 !+
3475 4300 3 Keep doing the QIO until we get something. We can get nothing if we were interrupted by a control Y
3476 4301 3 and then continued by a CONTINUE command.
3477 4302 3 !-
3478 4303 3 END
3479 4304 2 UNTIL (.EDT$SG_RDAHED GTR 0);
3480 4305 2
3481 4306 2 $$TRACE (EDT$STR_KEY, EDT$STR_SK_CH_NE, .EDT$SG_RDAHED, EDT$ST_RDAHED);
3482 4307 2 .C = CH$RCHAR (EDT$ST_RDAHED);
3483 4308 2 EDT$SG_RDAHED = .EDT$SG_RDAHED - 1;
3484 4309 2
3485 4310 2 IF (.EDT$SG_RDAHED GTR 0) THEN CH$MOVE (.EDT$SG_RDAHED, CH$PLUS (EDT$ST_RDAHED, 1), EDT$ST_RDAHED);
3486 4311 2
3487 4312 1 END;
! of EDT$STI_GETCH
```

```

FFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF FFFFFFFFF 013AF .BLKB 1
P.AAJ: .LONG -1, -1, -1, -1, -1, -1, -1, -1
FFFFFFFF FFFFFFFFF 013B0
FFFFFFFF FFFFFFFFF 013C8
```

```

59 00000000G 00 03FC 00000 .ENTRY EDT$STI_GETCH, Save R2,R3,R4,R5,R6,R7,R8,R9 : 4219
58 00000000G 00 9E 00002 MOVAB EDT$INTER_ERR, R9
57 00000000G 00 9E 00009 MOVAB EDT$SG_RDAHED, R8
56 00000000G 00 9E 00010 MOVAB EDT$ST_RDAHED, R7
5E 00000000G EF 9E 00017 MOVAB HAVE_EFNS, R6
08 C2 0001E SUBL2 #8, SP
68 D5 00021 TSTL EDT$SG_RDAHED : 4261
03 13 00023 BEQL 1$
69 00 FB 00025 CALLS #0, EDT$INTER_ERR
```

EDTS
V04-

PC	Op	OpC	OpD	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMM	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK	OpSL	OpSM	OpSN
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

; Routine Size: 179 bytes, Routine Base: _EDT\$CODE + 13D0

: 3488 4313 1

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS

EDT\$STI_WRSTR - write to terminal without carr

C 4

16-Sep-1984 00:35:33

14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 109

(34)

EDT\$
V04-

: 39

```
: 3490 4314 1 %SBTTL 'EDT$STI_WRSTR - write to terminal without carriage control'
: 3491 4315 1
: 3492 4316 1 GLOBAL ROUTINE EDT$STI_WRSTR (
: 3493 4317 1     BUF,
: 3494 4318 1     LEN
: 3495 4319 1 ) : NOVALUE =
: 3496 4320 1
: 3497 4321 1 ++
: 3498 4322 1 FUNCTIONAL DESCRIPTION:
: 3499 4323 1
: 3500 4324 1     Write a buffer on the terminal without implied carriage control.
: 3501 4325 1
: 3502 4326 1 FORMAL PARAMETERS:
: 3503 4327 1
: 3504 4328 1     BUF           Address of the buffer to write
: 3505 4329 1
: 3506 4330 1     LEN           Length of that buffer
: 3507 4331 1
: 3508 4332 1 IMPLICIT INPUTS:
: 3509 4333 1
: 3510 4334 1     TT_CHAN
: 3511 4335 1     OUTPUT_EFN
: 3512 4336 1     HAVE_EFNS
: 3513 4337 1     T_DEVCLASS
: 3514 4338 1
: 3515 4339 1 IMPLICIT OUTPUTS:
: 3516 4340 1
: 3517 4341 1     NONE
: 3518 4342 1
: 3519 4343 1 ROUTINE VALUE:
: 3520 4344 1
: 3521 4345 1     NONE
: 3522 4346 1
: 3523 4347 1 SIDE EFFECTS:
: 3524 4348 1
: 3525 4349 1     NONE
: 3526 4350 1
: 3527 4351 1 --
: 3528 4352 1
: 3529 4353 2 BEGIN
: 3530 4354 2
: 3531 4355 2 ++ Do nothing if the length is zero or SYSS$INPUT is not a terminal.
: 3532 4356 2 --
: 3533 4357 2
: 3534 4358 3 IF ((.LEN GTR 0) AND (.t_devclass EQL DCS_TERM))
: 3535 4359 3 THEN
: 3536 4360 3 BEGIN
: 3537 4361 3
: 3538 4362 3 ++ Make sure the event flags are allocated.
: 3539 4363 3 --
: 3540 4364 3
: 3541 4365 3 IF ( NOT .HAVE_EFNS) THEN EDT$STI_ENTERCHM ();
: 3542 4366 3
: 3543 4367 3 ASSERT (.HAVE_EFNS);
: 3544 4368 3 $$TRACE (EDT$STR_SCR, EDT$STR_SSCR_CH, .LEN, .BUF);
: 3545 4369 3 $$TRACE READING T<RETURN (0)>;
: 3546 4370 3 ASSERT T$WAITFR (EFN = .OUTPUT_EFN));
```

: Ro

: 39

: 39

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS

EDT\$\$TI_WRSTR - write to terminal without carr

D 4

16-Sep-1984 00:35:33

14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 110

(34)

```
: 3547 P 4371 3 ASSERT ($QIOW (EFN = .OUTPUT_EFN, CHAN = .TT_CHAN, P1 = .BUF, P2 = .LEN,  
: 3548 4372 3 FUNC = IOS_WRITEVBLK + IOSM_NOFORMAT));  
: 3549 4373 2 END;  
: 3550 4374 2  
: 3551 4375 1 END;
```

' of routine EDT\$\$TI_WRSTR

			000C 00000	.ENTRY	EDT\$\$TI_WRSTR, Save R2,R3	4316
	53	00000000G	00 9E 00002	MOVAB	EDT\$\$INTER_ERR, R3	
	52	00000000'	EF 9E 00009	MOVAB	HAVE_EFNS, R2	
		08	AC D5 00010	TSTL	LEN	4358
			4C 15 00013	BLEQ	3\$	
00000042	8F	E4	A2 D1 00015	CMPL	T_DEVCLASS, #66	
			42 12 0001D	BNEQ	3\$	
	0B		62 E8 0001F	BLBS	HAVE_EFNS, 1\$	4365
0000V	CF		00 FB 00022	CALLS	#0, EDT\$\$TI_ENTERCHM	
	03		62 E8 00027	BLBS	HAVE_EFNS, T\$	4367
	63		00 FB 0002A	CALLS	#0, EDT\$\$INTER_ERR	
		FC	A2 DD 0002D 1\$:	PUSHL	OUTPUT_EFN	4370
00000000G	00		01 FB 00030	CALLS	#1, SYSSWAITFR	
	03		50 E8 00037	BLBS	R0, 2\$	
	63		00 FB 0003A	CALLS	#0, EDT\$\$INTER_ERR	
			7E 7C 0003D 2\$:	CLRQ	-(SP)	4372
			7E 7C 0003F	CLRQ	-(SP)	
	7E	04	AC 7D 00041	MOVQ	BUF, -(SP)	
			7E 7C 00045	CLRQ	-(SP)	
			7E D4 00047	CLRL	-(SP)	
	7E	0130	8F 3C 00049	MOVZWL	#304, -(SP)	
		D8	A2 DD 0004E	PUSHL	TT_CHAN	
		FC	A2 DD 00051	PUSHL	OUTPUT_EFN	
00000000G	00		0C FB 00054	CALLS	#12, SYSSQIOW	
	03		50 E8 0005B	BLBS	R0, 3\$	
	63		00 FB 0005E	CALLS	#0, EDT\$\$INTER_ERR	
			04 00061 3\$:	RET		4375

; Routine Size: 98 bytes, Routine Base: _EDT\$CODE + 1483

; 3552 4376 1

EDT\$
V04-

: 39
: 39
: 39

: Si
: Ru
: El
: Li
: Le
: Me
: Co

```
3554 4377 1 %SBTTL 'EDT$$TI_RDITYAHED - read a character from typeahead'
3555 4378 1
3556 4379 1 GLOBAL ROUTINE EDT$$TI_RDITYAHED (          ! Read a character from typeahead
3557 4380 1     TEXT          ! Address of the string read
3558 4381 1     ) =
3559 4382 1
3560 4383 1 ++
3561 4384 1 FUNCTIONAL DESCRIPTION:
3562 4385 1
3563 4386 1     This routine reads the type-ahead buffer. It returns the string in
3564 4387 1     TEXT, the number of characters read as its value.
3565 4388 1
3566 4389 1 FORMAL PARAMETERS:
3567 4390 1
3568 4391 1     TEXT          Address of the string read
3569 4392 1
3570 4393 1 IMPLICIT INPUTS:
3571 4394 1
3572 4395 1     INPUT_EFN
3573 4396 1     OUTPUT_EFN
3574 4397 1     HAVE_EFNS
3575 4398 1     T_DEVCLASS
3576 4399 1
3577 4400 1 IMPLICIT OUTPUTS:
3578 4401 1
3579 4402 1     IOSB
3580 4403 1
3581 4404 1 ROUTINE VALUE:
3582 4405 1
3583 4406 1     The number of characters read
3584 4407 1
3585 4408 1 SIDE EFFECTS
3586 4409 1
3587 4410 1     NONE
3588 4411 1
3589 4412 1 --
3590 4413 1
3591 4414 2 BEGIN
3592 4415 2
3593 4416 2 LOCAL
3594 4417 2     RETURN_VALUE,
3595 4418 2     T_MASK : VECTOR [2];
3596 4419 2
3597 4420 2 !+
3598 4421 2 ! If SYSS$INPUT is not a terminal there can be no typeahead.
3599 4422 2 !-
3600 4423 2
3601 4424 2     IF (.t_devclass NEQ DC$_TERM) THEN RETURN (0);
3602 4425 2
3603 4426 2 !+
3604 4427 2 ! Make sure the event flag numbers are allocated.
3605 4428 2 !-
3606 4429 2
3607 4430 2     IF ( NOT .HAVE_EFNS) THEN EDT$$TI_ENTERCHM ();
3608 4431 2
3609 4432 2     ASSERT (.HAVE_EFNS);
3610 4433 2     T_MASK [0] = 32;
```

```

: 3611      4434 2      TMASK [1] = UPLIT (-1, 0, 0, %x'80000000', -1, 0, 0, 0);
: 3612      4435 2      !+
: 3613      4436 2      !- In trace mode we cannot process typeahead.
: 3614      4437 2      !-
: 3615      4438 2
: 3616      L 4439 3      %IF ( NOT edt$$tr_act)
: 3617      4440 2      %THEN
: 3618      4441 2      ASSERT ($WAITFR (EFN = .OUTPUT EFN));
: 3619      P 4442 2      ASSERT ($QIOW (EFN = .INPUT EFN, CHAN = .TT CHAN, P1 = .TEXT, P2 = K RDAHED LEN, P3 = 0, P4 = TMASK,
: 3620      4443 2      IOSB = IOSB, FUNC = IOS READVBLK + IOSM_NOECHO + IOSM_NOFILTR + IOSM_TIMED));
: 3621      4444 2      RETURN_VALUE = .IOSB [1] + .IOSB [3];
: 3622      U 4445 2      %ELSE
: 3623      U 4446 2      RETURN_VALUE = 0;
: 3624      4447 2      %FI
: 3625      4448 2
: 3626      4449 2      RETURN (.RETURN_VALUE);
: 3627      4450 1      END;
```

! of routine EDT\$\$TI_RDTYAHED

```

000000C0 FFFFFFFF 80000000 00000000 00000000 FFFFFFFF 014E5 P.AAK: .BLKB 3
00000000 00000000 00000000 01500 .LONG -1, 0, 0, -2147483648, -1, 0, 0, 0
```

```

                                000C 00000
                                53 00000000G 00 9E 00002
                                52 00000000' EF 9E 00009
                                SE          08 C2 00010
00000042 8F          E4 A2 D1 00013
                                5A 12 0001B
                                0B          62 E8 0001D
0000V CF          00 FB 00020
                                03          62 E8 00025
                                63          00 FB 00028
                                6E          20 D0 0002B 1$:
04 AE          AF AF 9E 0002E
                                FC A2 DD 00033
00000000G 00          01 FB 00036
                                03          50 E8 0003D
                                63          00 FB 00040
                                7E 7C 00043 2$:
                                08 AE 9F 00045
                                7E          20 7D 00048
                                04 AC DD 0004B
                                7E 7C 0004E
                                D0 A2 9F 00050
                                7E 02F1 8F 3C 00053
                                D8 A2 DD 00058
                                F8 A2 DD 0005B
00000000G 00          0C FB 0005E
                                03          50 E8 00065
                                63          00 FB 00068
                                50 D2 A2 3C 0006B 3$:
                                51 D6 A2 3C 0006F
                                50          51 C0 00073

.ENTRY EDT$$TI_RDTYAHED, Save R2,R3
MOVAB EDT$$INTER_ERR, R3
MOVAB HAVE_EFNS, R2
SUBL2 #8, SP
CMPL T DEVCLASS, #66
BNEQ 4$
BLBS HAVE_EFNS, 1$
CALLS #0, EDT$$TI_ENTERCHM
BLBS HAVE_EFNS, TS
CALLS #0, EDT$$INTER_ERR
MOVL #32, TMASK
MOVAB P.AAK, TMASK+4
PUSHL OUTPUT_EFN
CALLS #1, SYSSWAITFR
BLBS R0, 2$
CALLS #0, EDT$$INTER_ERR
CLRQ -(SP)
PUSHAB TMASK
MOVQ #32, -(SP)
PUSHL TEXT
CLRQ -(SP)
PUSHAB IOSB
MOVZWL #753, -(SP)
PUSHL TT CHAN
PUSHL INPUT_EFN
CALLS #12, SYSSQIOW
BLBS R0, 3$
CALLS #0, EDT$$INTER_ERR
MOVZWL IOSB+2, RETURN_VALUE
MOVZWL IOSB+6, R1
ADDL2 R1, RETURN_VALUE
```

.....

```

3630 4452 1 %SBTTL 'EDT$STI_RDSTR - read characters until control or DEL'
3631 4453 1
3632 4454 1 GLOBAL ROUTINE EDT$STI_RDSTR (
3633 4455 1     BUF,
3634 4456 1     LEN,
3635 4457 1     LEN_READ
3636 4458 1 ) : NOVALUE =
3637 4459 1
3638 4460 1 ++
3639 4461 1 FUNCTIONAL DESCRIPTION:
3640 4462 1
3641 4463 1     This routine reads characters with echo on until any control character or
3642 4464 1     a DEL character are seen.
3643 4465 1
3644 4466 1 FORMAL PARAMETERS:
3645 4467 1
3646 4468 1     BUF           Address of the buffer to read into.
3647 4469 1
3648 4470 1     LEN           Length of the buffer.
3649 4471 1
3650 4472 1     LEN_READ      Address of a longword to receive the number of characters read.
3651 4473 1
3652 4474 1 IMPLICIT INPUTS:
3653 4475 1
3654 4476 1     TT_CHAN
3655 4477 1     OUTPUT_EFN
3656 4478 1     T_MASK
3657 4479 1     EDT$ST_FMT_BUF
3658 4480 1     EDT$SA_FMT_CUR
3659 4481 1     IOSB
3660 4482 1     HAVE_EFNS
3661 4483 1     T_DEVCLASS
3662 4484 1
3663 4485 1 IMPLICIT OUTPUTS:
3664 4486 1
3665 4487 1     IOSB
3666 4488 1     EDT$SA_FMT_CUR
3667 4489 1     EDT$ST_RDAHED
3668 4490 1     EDT$SG_RDAHED
3669 4491 1
3670 4492 1 ROUTINE VALUE:
3671 4493 1
3672 4494 1     NONE
3673 4495 1
3674 4496 1 SIDE EFFECTS:
3675 4497 1
3676 4498 1     NONE
3677 4499 1
3678 4500 1 --
3679 4501 1
3680 4502 2 BEGIN
3681 4503 2
3682 4504 2 LOCAL
3683 4505 2     T_MASK : VECTOR [2],
3684 4506 2     TERM : VECTOR [K_RDAHED_LEN, BYTE],
3685 4507 2     LEN_TERM;
3686 4508 2
```

```

! The string that terminated the line, if any
! number of characters in the terminator
```

```
3687 4509 2 1+
3688 4510 2 1- If SYSSINPUT is not a terminal just return control Z.
3689 4511 2 1-
3690 4512 2 1-
3691 4513 3 IF (.t_devclass NEQ DCS_TERM)
3692 4514 3 THEN
3693 4515 3 BEGIN
3694 4516 3 .LEN_READ = 0;
3695 4517 3 EDT$ST_RDAHED = ASC_K_CTRL_Z;
3696 4518 3 EDT$SG_RDAHED = 1;
3697 4519 3 RETURN;
3698 4520 3 END;
3699 4521 2 1+
3700 4522 2 1- Make sure the event flags are allocated.
3701 4523 2 1-
3702 4524 2 1-
3703 4525 2 1-
3704 4526 2 IF ( NOT .HAVE_EFNS) THEN EDT$STI_ENTERCHM ();
3705 4527 2
3706 4528 2 ASSERT (.EDT$SG_RDAHED EQL 0);
3707 4529 2 ASSERT (.HAVE_EFNS);
3708 4530 2 1+
3709 4531 2 1- Set up the terminator mask.
3710 4532 2 1-
3711 4533 2 TMASK [0] = 32;
3712 4534 2 TMASK [1] = UPLIT (-1, 0, 0, XX'80000000', -1, 0, 0, 0);
3713 4535 2 $$TRACE (EDT$STR_KEY, EDT$STR_SK_PROMPT, .EDT$SA_FMT_CUR - EDT$ST_FMT_BUF, EDT$ST_FMT_BUF);
3714 4536 2 $$TRACE (EDT$STR_IDLE, EDT$STR_STIDLE_KIN, 0, 0);
3715 4537 2
3716 4538 2 IF $$TRACE_READ (EDT$STR_KEY, EDT$STR_SK_CH, MIN (.LEN, 72), .BUF, .LEN_READ)
3717 4539 2 THEN
3718 4540 3 BEGIN
3719 4541 3 1+
3720 4542 3 1- Since we got the record from the trace file, get the terminator from the trace file.
3721 4543 3 1-
3722 4544 3
3723 4545 3 LOCAL
3724 4546 3 READ_STATUS;
3725 4547 3
3726 4548 3 TERM [0] = 0;
3727 4549 3 READ_STATUS = $$TRACE_READ (EDT$STR_KEY, EDT$STR_SK_TERM_NE, K_RDAHED_LEN, TERM, LEN_TERM);
3728 4550 3 ASSERT (.READ_STATUS);
3729 4551 3 END
3730 4552 2 ELSE
3731 4553 2
3732 4554 2 DO
3733 4555 2 BEGIN
3734 4556 2 1+
3735 4557 2 1- Make sure output is complete.
3736 4558 2 1-
3737 4559 2
3738 4560 2 LOCAL
3739 4561 2 FUNC_VAL,
3740 4562 2 QIO_STATUS;
3741 4563 2
3742 4564 2 ASSERT ($WAITFR (EFN = .OUTPUT_EFN));
3743 4565 2 FUNC_VAL = IOS_READPROMPT + IOSM_TRMNOCCHO + IOSM_NOFILTR;
```

```
3744 4566 3
3745 4567 3      IF (( NOT edt$Str_act) AND (.EDT$ST_PMT_KPD [0] EQL 0)) THEN FUNC_VAL = .FUNC_VAL + IOSM_ESCAPE;
3746 4568 3
3747 P 4569 3      QIO_STATUS = $QIOW (EFN = .INPUT_EFN, CHAN = .TT_CHAN, P1 = .BUF, P2 = MIN (.LEN, 72),
3748 P 4570 3      P4 = TMASK, P5 = EDT$ST_FMT_BUF, P6 = .EDT$SA_FMT_CUR - EDT$ST_FMT_BUF, IOSB = IOSB,
3749 4571 3      FUNC = .FUNC_VAL);
3750 4572 3
3751 4573 3      IF ( NOT .QIO_STATUS) THEN SIGNAL_STOP (.QIO_STATUS);
3752 4574 3
3753 4575 3      .LEN_READ = .IOSB [1];          ! Number of bytes read
3754 4576 3      LEN_TERM = .IOSB [3];          ! Number of bytes in the terminator string
3755 4577 3      CH$MOVE (.LEN_TERM, CH$PLUS (.BUF, ..LEN_READ), TERM);
3756 4578 3
3757 4579 3      + Keep doing the QIO until we get something. We can get nothing if a control Y interrupts us
3758 4580 3      and we are later continued.
3759 4581 3      -
3760 4582 3      END
3761 4583 2      UNTIL ((.IOSB [1] + .IOSB [3]) GTR 0);
3762 4584 2
3763 4585 2      EDT$SA_FMT_CUR = EDT$ST_FMT_BUF;
3764 4586 2
3765 4587 2      + Make sure the line read and its terminator are recorded in the trace file.
3766 4588 2      -
3767 4589 2      $$TRACE (EDT$STR_KEY, EDT$STR_SK_CH, ..LEN_READ, .BUF);
3768 4590 2      $$TRACE (EDT$STR_KEY, EDT$STR_SK_TERM_NE, .LEN_TERM, TERM);
3769 4591 2
3770 4592 2      + If we had a terminator return it by placing it in the type-ahead buffer.
3771 4593 2      -
3772 4594 2
3773 4595 3      IF (.LEN_TERM GTR 0)
3774 4596 2      THEN
3775 4597 3      BEGIN
3776 4598 3      CH$MOVE (.LEN_TERM, TERM, EDT$ST_RDAHED);
3777 4599 3      EDT$G_RDAHED = .LEN_TERM;
3778 4600 2      END;
3779 4601 2
3780 4602 1      END;                                     ! of EDT$STI_RDSTR
```

```
00000000 FFFFFFFF 80000000 00000000 00000000 FFFFFFFF 01582 P.AAL: .BLKB 2
00000000 00000000 00000000 01584 .LONG -1, 0, 0, -2147483648, -1, 0, 0, 0
00000000 00000000 0159C
```

```
OFFC 00000 .ENTRY EDT$STI_RDSTR, Save R2,R3,R4,R5,R6,R7,R8,- 4454
5B 00000000G 00 9E 00002 R9,R10,R11
5A 00000000G 00 9E 00009 EDT$ST_FMT_BUF, R11
59 00000000G 00 9E 00010 EDT$INTER_ERR, R10
58 00000000' EF 9E 00017 EDT$G_RDAHED, R9
5E 28 C2 0001E HAVE_EFNS, R8
00000042 8F E4 A8 D1 00021 #40, -SP 4513
OE 13 00029 T_DEVCLASS, #66
BC D4 0002B BEQL 1$ 4516
1A D0 0002E CLRL @LEN_READ
MOVL #26, EDT$ST_RDAHED 4517
```

	69		01	D0	00035	MOVL	#1, EDT\$G_RDAHED	4518
				04	00038	RET		4515
	05		68	E8	00039	1\$: BLBS	HAVE EFNS, 2\$	4526
0000V	CF		00	FB	0003C	CALLS	#0, EDT\$STI_ENTERCHM	
			69	D5	00041	2\$: TSTL	EDT\$G_RDAHED	4528
			03	13	00043	BEQL	3\$	
	6A		00	FB	00045	CALLS	#0, EDT\$INTER_ERR	
	03		68	E8	00048	3\$: BLBS	HAVE EFNS, 4\$	4529
	6A		00	FB	0004B	CALLS	#0, EDT\$INTER_ERR	
20	AE		20	D0	0004E	4\$: MOVL	#32, TMASK	4533
24	AE	8B	AF	9E	00052	MOVAB	P.AAL, TMASK+4	4534
	56	08	AC	D0	00057	MOVL	LEN, R6	4571
00000048	8F		56	D1	0005B	CMPL	R6, #72	
			04	15	00062	BLEQ	5\$	
	56	48	8F	9A	00064	MOVZBL	#72, R6	
		FC	A8	DD	00068	5\$: PUSHL	OUTPUT_EFN	4564
00000000G	00		01	FB	0006B	CALLS	#1, SY\$SWAITFR	
	03		50	E8	00072	BLBS	R0, 6\$	
	6A		00	FB	00075	CALLS	#0, EDT\$INTER_ERR	
	51	1237	8F	3C	00078	6\$: MOVZWL	#4663, FUNC_VAL	4565
		00000000G	00	95	0007D	TSTB	EDT\$ST_PMT_RPD	4567
			05	12	00083	BNEQ	7\$	
	51	4000	C1	9E	00085	MOVAB	16384(R1), FUNC_VAL	
	50		6B	9E	0008A	7\$: MOVAB	EDT\$ST_FMT_BUF, -R0	4571
7E 00000000G	00		50	C3	0008D	SUBL3	R0, EDT\$A_FMT_CUR, -(SP)	
			5B	DD	00095	PUSHL	R11	
		28	AE	9F	00097	PUSHAB	TMASK	
			7E	D4	0009A	CLRL	-(SP)	
			56	DD	0009C	PUSHL	R6	
		04	AC	DD	0009E	PUSHL	BUF	
			7E	7C	000A1	CLRQ	-(SP)	
		D0	A8	9F	000A3	PUSHAB	IOSB	
			51	DD	000A6	PUSHL	FUNC_VAL	
		D8	A8	DD	000A8	PUSHL	TT_CHAN	
		F8	A8	DD	000AB	PUSHL	INPUT_EFN	
00000000G	00		0C	FB	000AE	CALLS	#12, SY\$QIOW	
	09		50	E8	000B5	BLBS	QIO_STATUS, 8\$	4573
			50	DD	000B8	PUSHL	QIO_STATUS	
00000000G	00		01	FB	000BA	CALLS	#1, LIB\$STOP	
0C	BC	D2	A8	3C	000C1	8\$: MOVZWL	IOSB+2, @LEN_READ	4575
	57	D6	A8	3C	000C6	MOVZWL	IOSB+6, LEN_TERM	4576
50	04	0C	BC	C1	000CA	ADDL3	@LEN_READ, BUF, R0	4577
6E	60		57	28	000D0	MOV3	LEN_TERM, (R0), TERM	
	50	D2	A8	3C	000D4	MOVZWL	IOSB+2, R0	4583
	51	D6	A8	3C	000D8	MOVZWL	IOSB+6, R1	
	50		51	C0	000DC	ADDL2	R1, R0	
			87	15	000DF	BLEQ	5\$	
00000000G	00		6B	9E	000E1	MOVAB	EDT\$ST_FMT_BUF, EDT\$A_FMT_CUR	4585
			57	D5	000E8	TSTL	LEN_TERM	4595
			0B	15	000EA	BLEQ	9\$	
00000000G	00		57	28	000EC	MOV3	LEN_TERM, TERM, EDT\$ST_RDAHED	4598
	6E		57	D0	000F4	MOVL	LEN_TERM, EDT\$G_RDAHED	4599
	69			04	000F7	9\$: RET		4602

; Routine Size: 248 bytes, Routine Base: _EDT\$CODE + 15A4

: 3781

4603 1

12-Sep-1984 00:35:33

VAX-11 Bliss-32 V4.0-742 Page 118
DISK\$VMSMASTER:[EDIT.SRC]IOMOD.B32;1 (36)

EDTS
V04-

```

3783 4604 1 XSBTTL 'EDT$STI_ENTERCHM - enter change mode'
3784 4605 1
3785 4606 1 GLOBAL ROUTINE EDT$STI_ENTERCHM . Enter change mode
3786 4607 1 : NOVALUE =
3787 4608 1
3788 4609 1 ++
3789 4610 1 FUNCTIONAL DESCRIPTION:
3790 4611 1
3791 4612 1 Enter change mode. Allocate event flags.
3792 4613 1
3793 4614 1 FORMAL PARAMETERS:
3794 4615 1
3795 4616 1 NONE
3796 4617 1
3797 4618 1 IMPLICIT INPUTS:
3798 4619 1
3799 4620 1 HAVE_EFNS
3800 4621 1 T_DEVCLASS
3801 4622 1
3802 4623 1 IMPLICIT OUTPUTS:
3803 4624 1
3804 4625 1 INPUT_EFN
3805 4626 1 OUTPUT_EFN
3806 4627 1 HAVE_EFNS
3807 4628 1
3808 4629 1 ROUTINE VALUE:
3809 4630 1
3810 4631 1 NONE
3811 4632 1
3812 4633 1 SIDE EFFECTS:
3813 4634 1
3814 4635 1 Allocates two event flags for full-duplex terminal I/O, if necessary.
3815 4636 1
3816 4637 1 --
3817 4638 1
3818 4639 2 BEGIN
3819 4640 2
3820 4641 2 LOCAL
3821 4642 2 INPUT_EF_STATUS,
3822 4643 2 OUTPUT_EF_STATUS;
3823 4644 2
3824 4645 2 ++
3825 4646 2 If the event flags are already allocated, or if SYS$INPUT is not a terminal, do nothing.
3826 4647 2 -
3827 4648 2
3828 4649 2 IF ((.HAVE_EFNS) OR (.t_devclass NEQ DC$_TERM)) THEN RETURN;
3829 4650 2
3830 4651 2 ++
3831 4652 2 Allocate event flags for input and output.
3832 4653 2 -
3833 4654 2 INPUT_EF_STATUS = LIB$GET_EF (INPUT_EFN);
3834 4655 2
3835 4656 2 IF ( NOT .INPUT_EF_STATUS) THEN SIGNAL_STOP (.INPUT_EF_STATUS);
3836 4657 2
3837 4658 2 OUTPUT_EF_STATUS = LIB$GET_EF (OUTPUT_EFN);
3838 4659 2
3839 4660 2 IF ( NOT .OUTPUT_EF_STATUS) THEN SIGNAL_STOP (.OUTPUT_EF_STATUS);
```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$TI_ENTERCHM - enter change mode

N 4
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 120
(37)

EDT\$
V04-

```
: 3840      4661  2
: 3841      4662  2      $SETEF (EFN = .OUTPUT_EFN);
: 3842      4663  2      HAVE_EFNS = 1;
: 3843      4664  2      ASSERT (.HAVE_EFNS);
: 3844      4665  1      END;
```

of routine EDT\$\$TI_ENTERCHM

```
00000042 54 00000000G 00 001C 00000
          53 00000000G 00 9E 00002
          52 00000000' EF 9E 00010
          3D          62 E8 00017
          8F          A2 D1 0001A
          E4          33 12 00022
          F8          A2 9F 00024
          63          01 FB 00027
          05          50 E8 0002A
          64          50 DD 0002D
          FC          01 FB 0002F 1$:
          63          01 FB 00035
          05          50 E8 00038
          64          50 DD 0003B
          FC          01 FB 0003D 2$:
00000000G 00          01 FB 00043
          62          01 D0 0004A
          07          62 E8 0004D
00000000G 00          00 FB 00050
          04 00057 3$:
```

.EXTRN SYS\$SETEF

```
.ENTRY EDT$$TI_ENTERCHM, Save R2,R3,R4      : 4606
MOVAB LIB$STOP, R4
MOVAB LIB$GET_EF, R3
MOVAB HAVE_EFNS, R2
BLBS HAVE_EFNS, 3$      : 4649
CMPL T_DEVCLASS, #66
BNEQ 3$
PUSHAB INPUT_EFN      : 4654
CALLS #1, LIB$GET_EF
BLBS INPUT_EF_STATUS, 1$      : 4656
PUSHL INPUT_EF_STATUS
CALLS #1, LIB$STOP
PUSHAB OUTPUT_EFN      : 4658
CALLS #1, LIB$GET_EF
BLBS OUTPUT_EF_STATUS, 2$      : 4660
PUSHL OUTPUT_EF_STATUS
CALLS #1, LIB$STOP
PUSHL OUTPUT_EFN      : 4662
CALLS #1, SYS$SETEF
MOVL #1, HAVE_EFNS      : 4663
BLBS HAVE_EFNS, 3$      : 4664
CALLS #0, EDT$$INTER_ERR
RET      : 4665
```

: Routine Size: 88 bytes. Routine Base: _EDT\$CODE + 169C

: 3845 4666 1

```
3847 4667 1 %SBTTL 'EDT$STI_LEAVECHM - leave change mode'
3848 4668 1
3849 4669 1 GLOBAL ROUTINE EDT$STI_LEAVECHM ! Leave change mode
3850 4670 1 : NOVALUE =
3851 4671 1
3852 4672 1 ++
3853 4673 1 FUNCTIONAL DESCRIPTION:
3854 4674 1
3855 4675 1 Leave change mode.
3856 4676 1
3857 4677 1 FORMAL PARAMETERS:
3858 4678 1
3859 4679 1 NONE
3860 4680 1
3861 4681 1 IMPLICIT INPUTS:
3862 4682 1
3863 4683 1 HAVE_EFNS
3864 4684 1 INPUT_EFN
3865 4685 1 OUTPUT_EFN
3866 4686 1
3867 4687 1 IMPLICIT OUTPUTS:
3868 4688 1
3869 4689 1 HAVE_EFNS
3870 4690 1
3871 4691 1 ROUTINE VALUE:
3872 4692 1
3873 4693 1 NONE
3874 4694 1
3875 4695 1 SIDE EFFECTS:
3876 4696 1
3877 4697 1 NONE
3878 4698 1
3879 4699 1 --
3880 4700 1
3881 4701 2 BEGIN
3882 4702 2
3883 4703 2 LOCAL
3884 4704 2 FREE_INPUT_STATUS,
3885 4705 2 FREE_OUTPUT_STATUS;
3886 4706 2
3887 4707 2 ++
3888 4708 2 ! If we don't have the event flags, do nothing.
3889 4709 2 --
3890 4710 2
3891 4711 2 IF ( NOT .HAVE_EFNS) THEN RETURN;
3892 4712 2
3893 4713 2 ASSERT ($WAITFR (EFN = .OUTPUT_EFN));
3894 4714 2 FREE_INPUT_STATUS = LIB$FREE_EF (INPUT_EFN);
3895 4715 2
3896 4716 2 IF ( NOT .FREE_INPUT_STATUS) THEN SIGNAL_STOP (.FREE_INPUT_STATUS);
3897 4717 2
3898 4718 2 FREE_OUTPUT_STATUS = LIB$FREE_EF (OUTPUT_EFN);
3899 4719 2
3900 4720 2 IF ( NOT .FREE_OUTPUT_STATUS) THEN SIGNAL_STOP (.FREE_OUTPUT_STATUS);
3901 4721 2
3902 4722 2 HAVE_EFNS = 0;
3903 4723 2 ASSERT ( NOT .HAVE_EFNS);
```

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$\$TI_LEAVECHM - leave change mode

C 5
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 122
(38)

EDT!
V04-

; 3904

4724 1 END;

! of routine EDT\$\$TI_LEAVECHM

			003C 00000	.ENTRY	EDT\$\$TI_LEAVECHM, Save R2,R3,R4,R5	: 4669
55	00000000G	00	9E 00002	MOVAB	LIB\$STOP, R5	:
54	00000000G	00	9E 00009	MOVAB	LIB\$FREE_EF, R4	:
53	00000000G	00	9E 00010	MOVAB	EDT\$\$INTER_ERR, R3	:
52	00000000'	EF	9E 00017	MOVAB	HAVE_EFNS, R2	:
34		62	E9 0001E	BLBC	HAVE_EFNS, 4\$	: 4711
		A2	DD 00021	PUSHL	OUTPUT_EFN	: 4713
00000000G		01	FB 00024	CALLS	#1, SY\$WAITFR	:
03		50	E8 0002B	BLBS	R0, 1\$	:
63		00	FB 0002E	CALLS	#0, EDT\$\$INTER_ERR	:
		A2	9F 00031 1\$:	PUSHAB	INPUT_EFN	: 4714
64		01	FB 00034	CALLS	#1, LIB\$FREE_EF	:
05		50	E8 00037	BLBS	FREE_INPUT_STATUS, 2\$	: 4716
		50	DD 0003A	PUSHL	FREE_INPUT_STATUS	:
65		01	FB 0003C	CALLS	#1, LIB\$STOP	:
		A2	9F 0003F 2\$:	PUSHAB	OUTPUT_EFN	: 4718
64		01	FB 00042	CALLS	#1, LIB\$FREE_EF	:
05		50	E8 00045	BLBS	FREE_OUTPUT_STATUS, 3\$	: 4720
		50	DD 00048	PUSHL	FREE_OUTPUT_STATUS	:
65		01	FB 0004A	CALLS	#1, LIB\$STOP	:
		62	D4 0004D 3\$:	CLRL	HAVE_EFNS	: 4722
03		62	E9 0004F	BLBC	HAVE_EFNS, 4\$	: 4723
63		00	FB 00052	CALLS	#0, EDT\$\$INTER_ERR	:
		04	00055 4\$:	RET		: 4724

; Routine Size: 86 bytes, Routine Base: _EDT\$CODE + 16F4

; 3905
; 3906

4725 1
4726 1 !<BLF/PAGE>

EDT\$IOMOD
V04-000

EDT\$IOMOD - I/O for VAX/VMS
EDT\$STI_LEAVECHM - leave change mode

D 5
16-Sep-1984 00:35:33
14-Sep-1984 12:23:19

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[EDT.SRC]IOMOD.B32;1

Page 123
(39)

EDT\$
V04-

: 3908
: 3909
: 3910

4727 1 END
4728 1
4729 0 ELUDOM

: of module EDT\$IOMOD

.EXTRN LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
EDT\$DATA	2840	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON, PIC,ALIGN(2)
EDT\$CODE	5962	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[EDT.SRC]EDT.L32;1	377	11	2	40	00:00.2
\$255\$DUA28:[EDT.SRC]PSECTS.L32;1	2	1	50	7	00:00.1
\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	199	2	581	00:04.0

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACEBACK/LIS=LISS:IOMOD/OBJ=OBJ\$:IOMOD MSRC\$:IOMOD.B32/UPDATE=(ENH\$:IOMOD)

Size: 5784 code + 3018 data bytes
Run Time: 04:20.8
Elapsed Time: 05:35.4
Lines/CPU Min: 1087
Lexemes/CPU-Min: 17411
Memory Used: 388 pages
Compilation Complete

0134 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0134	0135	0136	0137	0138	0139	0140	0141	0142	0143	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191	0192	0193	0194	0195	0196	0197	0198	0199																																		
0200	0201	0202	0203	0204	0205	0206	0207	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299
0300	0301	0302	0303	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495	0496	0497	0498	0499
0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591	0592	0593	0594	0595	0596	0597	0598	0599
0600	0601	0602	0603	0604	0605	0606	0607	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699
0700	0701	0702	0703	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895	0896	0897	0898	0899
0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991	0992	0993	0994	0995	0996	0997	0998	0999

0135

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY