


```

BBBBBBBB  AAAAAA  SSSSSSSS  KK      KK      IIIIII  LL      LL
BBBBBBBB  AAAAAA  SSSSSSSS  KK      KK      IIIIII  LL      LL
BB      BB  AA      AA  SS      SS      KK      KK      II      LL      LL
BB      BB  AA      AA  SS      SS      KK      KK      II      LL      LL
BB      BB  AA      AA  SS      SS      KK      KK      II      LL      LL
BBBBBBBB  AA      AA  SSSSSS  KKKKKK  II      LL      LL
BBBBBBBB  AA      AA  SSSSSS  KKKKKK  II      LL      LL
BB      BB  AAAAAAAAAA  SS      KK      KK      II      LL      LL
BB      BB  AAAAAAAAAA  SS      KK      KK      II      LL      LL
BB      BB  AA      AA  SS      KK      KK      II      LL      LL
BB      BB  AA      AA  SS      KK      KK      II      LL      LL
BBBBBBBB  AA      AA  SSSSSSSS  KK      KK      IIIIII  LLLLLLLLLL  LLLLLLLLLL
BBBBBBBB  AA      AA  SSSSSSSS  KK      KK      IIIIII  LLLLLLLLLL  LLLLLLLLLL

```

```

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
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS

```

```

1 0001 0 MODULE BASSKILL (
2 0002 0 IDENT = '1-007'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
10 0010 1 * ALL RIGHTS RESERVED. *
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
17 0017 1 * TRANSFERRED. *
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
21 0021 1 * CORPORATION. *
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: BASIC-PLUS-2 Miscellaneous
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module implements the BASIC KILL statement,
36 0036 1 which deletes a file.
37 0037 1
38 0038 1 ENVIRONMENT: VAX-11 User Mode
39 0039 1
40 0040 1 AUTHOR: John Sauter, CREATION DATE: 01-MAR-1979
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 - Original. JBS 01-MAR-1979
45 0045 1 1-002 - Correct a typo in a comment. JBS 14-MAY-1979
46 0046 1 1-003 - Go through the full error analyzer if an error occurs.
47 0047 1 JBS 09-AUG-1979
48 0048 1 1-004 - Add code to allow an open file to be KILLED. PLL 15-Jun-1982
49 0049 1 1-005 - Edit 004 should have included a guard against ASTs. PLL 17-Jun-1982
50 0050 1 1-006 - Edit 004 should have released the CCB after doing what it needed to
51 0051 1 once it found a match. MDL 30-Sep-1982
52 0052 1 1-007 - Edit 004 should also release the CCB when it doesn't find a match;
53 0053 1 otherwise the IOINPROG bit for that unit gets set and never cleared,
54 0054 1 causing subsequent OPENS to fail with IOCHAALR. MDL 6-Feb-1984
55 0055 1 --
56 0056 1
57 0057 1 !<BLF/PAGE>
    
```

```

59      0058 1  |
60      0059 1  | SWITCHES:
61      0060 1  |
62      0061 1  |
63      0062 1  | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
64      0063 1  |
65      0064 1  |
66      0065 1  | LINKAGES:
67      0066 1  |
68      0067 1  |
69      0068 1  | REQUIRE 'RTLIN:OTSLNK';
70      0497 1  |
71      0498 1  |
72      0499 1  | TABLE OF CONTENTS:
73      0500 1  |
74      0501 1  |
75      0502 1  | FORWARD ROUTINE
76      0503 1  |     BASSKILL : NOVALUE,
77      0504 1  |     PUSH_CCB : CALL_CCB;
78      0505 1  |
79      0506 1  | INCLUDE FILES:
80      0507 1  |
81      0508 1  |
82      0509 1  | REQUIRE 'RTLIN:RTLPSECT';
83      0604 1  |
84      0605 1  | REQUIRE 'RTLML:OTSLUB';
85      0745 1  |
86      0746 1  | LIBRARY 'RTLSTARLE';
87      0747 1  |
88      0748 1  |
89      0749 1  | MACROS:
90      0750 1  |
91      0751 1  |     NONE
92      0752 1  |
93      0753 1  | EQUATED SYMBOLS:
94      0754 1  |
95      0755 1  |     NONE
96      0756 1  |
97      0757 1  | PSECTS:
98      0758 1  |
99      0759 1  | DECLARE_PSECTS (BAS);
100     0760 1  |
101     0761 1  | OWN STORAGE:
102     0762 1  |
103     0763 1  |     NONE
104     0764 1  |
105     0765 1  | EXTERNAL REFERENCES:
106     0766 1  |
107     0767 1  |
108     0768 1  | EXTERNAL ROUTINE
109     0769 1  |     BASS$STOP : NOVALUE,
110     0770 1  |     BASS$STOP_RMS : NOVALUE,
111     0771 1  |     BASS$NEXT_LUN : NOVALUE,
112     0772 1  |     BASS$CB_PUSH : JSB_CB_PUSH NOVALUE,
113     0773 1  |     BASS$CB_POP : JSB_CB_POP NOVALUE,
114     0774 1  |     LIB$SIG_TO_RET;

! Change a file's name
! Calls BASS$CB_PUSH

! Macros for defining psects

! System definitions

! Declare psects for BASS$ facility

! signal fatal error
! signals fatal RMS error
! find next allocated LUN
! get a CCB
! release a CCB
! condition handler

```

```

: 116 0775 1 GLOBAL ROUTINE BASSKILL (           ! Delete a file
: 117 0776 1     FILE_NAME                       ! Name of file to delete
: 118 0777 1     ) : NOVA CUE =
: 119 0778 1
: 120 0779 1     ++
: 121 0780 1     FUNCTIONAL DESCRIPTION:
: 122 0781 1
: 123 0782 1         Delete a file. This is done by using the $ERASE
: 124 0783 1         RMS macro.
: 125 0784 1
: 126 0785 1     FORMAL PARAMETERS:
: 127 0786 1
: 128 0787 1         FILE_NAME.rt.dx The name of the file to delete.
: 129 0788 1
: 130 0789 1     IMPLICIT INPUTS:
: 131 0790 1
: 132 0791 1         NONE
: 133 0792 1
: 134 0793 1     IMPLICIT OUTPUTS:
: 135 0794 1
: 136 0795 1         NONE
: 137 0796 1
: 138 0797 1     ROUTINE VALUE:
: 139 0798 1     COMPLETION CODES:
: 140 0799 1
: 141 0800 1         NONE
: 142 0801 1
: 143 0802 1     SIDE EFFECTS:
: 144 0803 1
: 145 0804 1         Deletes the specified file and removes its directory entry.
: 146 0805 1
: 147 0806 1     --
: 148 0807 1
: 149 0808 2     BEGIN
: 150 0809 2
: 151 0810 2     GLOBAL REGISTER
: 152 0811 2         CCB = 11 : REF BLOCK [,BYTE];
: 153 0812 2
: 154 0813 2     MAP
: 155 0814 2         FILE_NAME : REF BLOCK [8, BYTE];
: 156 0815 2
: 157 0816 2     LOCAL
: 158 0817 2         FAB_BLOCK : $FAB_DECL,
: 159 0818 2         NAM_BLOCK : $NAM_DECL,
: 160 0819 2         NAMBUF : VECTOR [NAM$C_MAXRSS, BYTE],
: 161 0820 2         ERASE_RESULT,
: 162 0821 2         LUN_FLAG,
: 163 0822 2         STATUS : INITIAL (1),
: 164 0823 2         UNIT;
: 165 0824 2
: 166 0825 2     CH$FILL (%C' ', NAM$C_MAXRSS, NAMBUF);           ! init resultant name to blanks
: 167 0826 2
: 168 P 0827 2     $FAB_INIT (FAB = FAB_BLOCK,                       !
: 169 P 0828 2         FNA = .FILE_NAME [DSC$A_POINTER],           !
: 170 P 0829 2         FNS = .FILE_NAME [DSC$W_LENGTH],           !
: 171 0830 2         NAM = NAM_BLOCK);
: 172 0831 2

```

```

: 173 P 0832 2 $NAM_INIT (NAM = NAM_BLOCK,
: 174 PP 0833 ESS = NAM$C_MAXRSS,
: 175 PP 0834 ESA = NAM$BUF,
: 176 P 0835 RSS = NAM$C_MAXRSS,
: 177 0836 RSA = NAM$BUF);
: 178 0837 !
: 179 0838 ERASE_RESULT = $ERASE (FAB = FAB_BLOCK);
: 180 0839
: 181 0840 IF ( NOT .ERASE_RESULT)
: 182 0841 THEN
: 183 0842 BEGIN
: 184 0843 +
: 185 0844 Analyze the RMS error status to give a BASIC error message.
: 186 0845 -
: 187 0846 IF (.ERASE_RESULT NEQ RMS$_FLK)
: 188 0847 THEN
: 189 0848 BASS$STOP_RMS (.FILE_NAME, .FAB_BLOCK [FAB$_STS], .FAB_BLOCK [FAB$_STV])
: 190 0849 ELSE
: 191 0850 BEGIN
: 192 0851 +
: 193 0852 If this is a file locked error, perhaps the user is trying to KILL a file
: 194 0853 that is still open. See if the file is open on any LUN. If it is, set
: 195 0854 the LUB$_DELETE bit which will be acted upon at close time and do not
: 196 0855 return an error.
: 197 0856 -
: 198 0857 $PARSE (FAB = FAB_BLOCK);
: 199 0858 $SEARCH (FAB = FAB_BLOCK); ! get resultant name
: 200 0859 LUN_FLAG = 0;
: 201 0860 IF .STATUS THEN DO
: 202 0861 BEGIN
: 203 0862 BASS$NEXT_LUN (LUN_FLAG, UNIT); ! get next used LUN
: 204 0863 IF .LUN_FLAG NEQ 0
: 205 0864 THEN
: 206 0865 BEGIN
: 207 0866 +
: 208 0867 We have a unit which has been allocated by Basic.
: 209 0868 Call BASS$CB_PUSH and see if the file names match.
: 210 0869 -
: 211 0870 STATUS = PUSH_CCB (.UNIT);
: 212 0871 IF .STATUS AND (.CCB NEQ 0)
: 213 0872 THEN
: 214 0873 IF .CCB [LUB$_OPENED]
: 215 0874 THEN
: 216 0875 IF CH$EQL (.CCB [LUB$_RSL], .CCB [LUB$_RSN],
: 217 0876 .NAM_BLOCK [NAM$_RSS], .NAM_BLOCK [NAM$_RSA], %C ' ')
: 218 0877 THEN
: 219 0878 BEGIN
: 220 0879 +
: 221 0880 Found a match. Disable ASTs and make sure
: 222 0881 the file has not been closed since we first
: 223 0882 discovered the match. If still open, then
: 224 0883 set the delete bit in the LUB. If the file
: 225 0884 has already been closed, try $ERASE again.
: 226 0885 -
: 227 0886 LOCAL
: 228 0887 AST_STATUS;
: 229 0888 AST_STATUS = $SETAST (ENBFLG = 0); ! disable ASTs

```

```

: 230 0889 7
: 231 0890 7
: 232 0891 7
: 233 0892 7
: 234 0893 8
: 235 0894 8
: 236 0895 8
: 237 0896 8
: 238 0897 7
: 239 0898 7
: 240 0899 7
: 241 0900 7
: 242 0901 7
: 243 0902 7
: 244 0903 7
: 245 0904 7
: 246 0905 7
: 247 0906 7
: 248 0907 6
: 249 0908 6
: 250 0909 6
: 251 0910 6
: 252 0911 6
: 253 0912 5
: 254 0913 5
: 255 0914 4
: 256 0915 4
: 257 0916 4
: 258 0917 4
: 259 0918 4
: 260 0919 4
: 261 0920 4
: 262 0921 4
: 263 0922 4
: 264 0923 4
: 265 0924 4
: 266 0925 4
: 267 0926 4
: 268 0927 4
: 269 0928 4
: 270 0929 3
: 271 0930 3
: 272 0931 2
: 273 0932 2
: 274 0933 2
: 275 0934 1

```

```

IF CHSEQL (.CCB [LUB$B_RSL], .CCB [LUB$A_RSN],
.NAM_BLOCK [NAM$B_RSS], .NAM_BLOCK [NAM$L_RSA],
%C '-')
THEN
BEGIN
CCB [LUB$V_DELETE] = 1;
ERASE_RESULT = 1;
END
ELSE
ERASE_RESULT = $ERASE (FAB = FAB_BLOCK);
IF .AST_STATUS EQL SSS_WASSET
THEN
SSETAST (ENBFLG = 1); ! re-enable ASTs
!+
!- release the unit (CCB).
BAS$$CB_POP ();
EXITLOOP
END;

!+
!- No match. Release this unit and continue scanning units.
BAS$$CB_POP ();
END;

END
UNTIL .LUN_FLAG EQL 0; ! until no more LUNs

IF NOT .STATUS
THEN
!+
!- Signal error. File was not open on any channel.
BAS$$STOP (.STATUS)
ELSE
!+
!- No PUSH_CCB error. Any $ERASE error?
IF NOT .ERASE_RESULT
THEN
BAS$$STOP_RMS (.FILE_NAME, .FAB_BLOCK [FAB$L_STS], .FAB_BLOCK [FAB$L_STV]);
END;
END;

RETURN;
END; ! end of BASSKILL

```

```

.TITLE BASSKILL
.IDENT \1-007\

.EXTRN BAS$$STOP, BAS$$STOP_RMS
.EXTRN BAS$$NEXT_LUN, BAS$$CB_PUSH
.EXTRN BAS$$CB_POP, LIB$$SIG_TO_RET
.EXTRN SY$$ERASE, SY$$PARSE
.EXTRN SY$$SEARCH, SY$$SETAST

```



```

: 277      0935 1 ROUTINE PUSH_CCB (                               ! Call BASS$CB_PUSH and return with value
: 278      0936 1         UNIT) : CALL_CCB =
: 279      0937 1
: 280      0938 1 ++
: 281      0939 1 ABSTRACT:
: 282      0940 1
: 283      0941 1         Call BASS$CB_PUSH and return with a condition value as the function
: 284      0942 1         result. BASS$CB_PUSH is not called directly as it may signal when
: 285      0943 1         we do not desire it.
: 286      0944 1
: 287      0945 1 FORMAL PARAMETERS:
: 288      0946 1
: 289      0947 1         UNIT - the unit to be pushed
: 290      0948 1
: 291      0949 1 FUNCTION RESULT:
: 292      0950 1
: 293      0951 1         Either SS$NORMAL or the condition code of an error which was
: 294      0952 1         signalled by BASS$CB_PUSH.
: 295      0953 1
: 296      0954 1 --
: 297      0955 1
: 298      0956 2 BEGIN
: 299      0957 2
: 300      0958 2     EXTERNAL REGISTER
: 301      0959 2         CCB;
: 302      0960 2
: 303      0961 2     ENABLE
: 304      0962 2         LIB$SIG_TO_RET;                               ! Converts signals to return values
: 305      0963 2
: 306      0964 2     BASS$CB_PUSH (.UNIT, LUB$K_ILUN_MIN);           ! Push the CCB
: 307      0965 2
: 308      0966 2     RETURN 1;                                       ! Success
: 309      0967 2
: 310      0968 1     END;                                             ! end of PUSH_CCB

```

```

                                0004 0000 PUSH_CCB:
                                .WORD   Save R2
                                MOVAL   1$, (FP)
                                MNEGL   #8, R0
                                MOVL    UNIT, R2
                                JSB     BASS$CB_PUSH
                                MOVL    #1, R0
                                RET
                                .WORD   Save nothing
                                CLRL    -(SP)
                                PUSHL   SP
                                MOVQ   4(AP), -(SP)
                                CALLS  #3, LIB$SIG_TO_RET
                                RET

```

; Routine Size: 42 bytes, Routine Base: _BAS\$CODE + 014B

BASSKILL
1-007

H 12
16-Sep-1984 00:41:22
14-Sep-1984 11:55:13

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASKILL.B32;1

Page 9
(4)

```
: 311      0969 1  
: 312      0970 1 END  
: 313      0971 1  
: 314      0972 0 ELUDOM
```

! end of module BASSKILL

PSECT SUMMARY

```
: Name          Bytes          Attributes  
: _BASSCODE     373 NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)
```

Library Statistics

```
: File          Total  Symbols  Percent  Pages  Processing  
:              -----  Loaded  -----  Mapped  Time  
: _$255$DUA28:[SYSLIB]STARLET.L32;1  9776      65      0      581    00:01.1
```

COMMAND QUALIFIERS

```
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:BASKILL/OBJ=OBJ$:BASKILL MSRC$:BASKILL/UPDATE=(ENH$:BASKILL)
```

```
: Size:          373 code + 0 data bytes  
: Run Time:      00:12.2  
: Elapsed Time: 00:28.3  
: Lines/CPU Min: 4768  
: Lexemes/CPU-Min: 40150  
: Memory Used: 173 pages  
: Compilation Complete
```

