

BBBBBBBB	AAAAAA	SSSSSSSS	PPPPPPPP	UU	UU	TTTTTTTTTT	
BBBBBBBB	AAAAAA	SSSSSSSS	PPPPPPPP	UU	UU	TTTTTTTTTT	
BB	AA	SS	PP	UU	UU	TT	
BB	AA	SS	PP	UU	UU	TT	
BB	AA	SS	PP	UU	UU	TT	
BB	AA	SS	PP	UU	UU	TT	
BBBBBBBB	AA	SSSSSS	PPPPPPPP	UU	UU	TT	
BBBBBBBB	AA	SSSSSS	PPPPPPPP	UU	UU	TT	
BB	AAAAAAAAAA	SS	PP	UU	UU	TT	
BB	AAAAAAAAAA	SS	PP	UU	UU	TT	
BB	AA	SS	PP	UU	UU	TT	
BB	AA	SS	PP	UU	UU	TT	
BBBBBBBB	AA	SSSSSSSS	PP	UUUUUUUUUU	UU	TT	
BBBBBBBB	AA	SSSSSSSS	PP	UUUUUUUUUU	UU	TT	

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 BAS$PUT (
2 0002 0 IDENT = '1-011'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY:
32 0032 1 Basic support library - user callable
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 This module is the UPI level of the Basic PUT construct. Initially,
37 0037 1 it contains only the code for sequential I/O. This module will set
38 0038 1 up the I/O data base for the LUN and go directly to the REC level.
39 0039 1
40 0040 1
41 0041 1 ENVIRONMENT:
42 0042 1 User access mode - AST reentrant.
43 0043 1
44 0044 1 AUTHOR: Donald G. Petersen, CREATION DATE: 19-Feb-79
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1
48 0048 1 DGP, 19-Feb-79 : VERSION 01
49 0049 1 1-001 - original. DGP 19-Feb-79
50 0050 1 1-002 - Add PUT with count. DGP 02-Mar-79
51 0051 1 1-003 - Add PUT_RECORD, PUT_REC_CNT. DGP 02-Mar-79
52 0052 1 1-004 - More work on relative I/O. DGP 05-Mar-79
53 0053 1 1-005 - Put in the junk for foreign buffers. DGP 28-Mar-79
54 0054 1 1-006 - Set up ISB$A_USER_FP. JBS 25-JUL-1979
55 0055 1 1-007 - Pass count to record level correctly. JBS 31-JUL-1979
56 0056 1 1-008 - Signal if file has been used for virtual I/O; set BLK_USE flag.
57 0057 1 DGP 16-Oct-79

```

! Basic PUT construct
 ! File: BASPUT.B32 Edit:FM1011

BAS\$PUT
1-011

H 1
16-Sep-1984 00:59:27 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:56:32 [BASRTL.SRC]BASPUT.B32;1

Page 2
(1)

```
: 58      0058 1 : 1-009 - Signal ILLIO CHA if channel passed is less than zero.
: 59      0059 1 : 1-010 - Pass to bas$$cb_push, lub$ ilun_min+2 so GET #0 BASIC statement
: 60      0060 1 : generate an error. FM 17-SEP-80
: 61      0061 1 : 1-011 - Lift the 1-010 restriction that was put on at V2.2. Use foreign
: 62      0062 1 : buffer mechanism for channel 0. FM 9-jul-81.
: 63      0063 1 : --
: 64      0064 1 :
: 65      0065 1 : <BLF/PAGE>
```



```

67 0066 1 |
68 0067 1 | SWITCHES
69 0068 1 |
70 0069 1 |
71 0070 1 | SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
72 0071 1 |
73 0072 1 |
74 0073 1 | LINKAGES
75 0074 1 |
76 0075 1 |
77 0076 1 | REQUIRE 'RTLIN:OTSLNK';           ! Define all linkages
78 0505 1 |
79 0506 1 |
80 0507 1 | TABLE OF CONTENTS:
81 0508 1 |
82 0509 1 |
83 0510 1 | FORWARD ROUTINE
84 0511 1 |     BASSPUT_REC CNT : NOVALUE,    ! UPI level Relative PUT with count
85 0512 1 |     BASSPUT_RECORD : NOVALUE,    ! UPI level Relative PUT
86 0513 1 |     BASSPUT_COUNT : NOVALUE,    ! UPI level Sequential PUT with count
87 0514 1 |     BASSPUT : NOVALUE;           ! UPI level Sequential PUT
88 0515 1 |
89 0516 1 |
90 0517 1 | INCLUDE FILES:
91 0518 1 |
92 0519 1 |
93 0520 1 | REQUIRE 'RTLML:OTSISB';           ! ISB definitions
94 0688 1 |
95 0689 1 | REQUIRE 'RTLML:BASPAR';           ! Basic specific parameters
96 0711 1 |
97 0712 1 | REQUIRE 'RTLML:OTSLUB';           ! LUB definitions
98 0852 1 |
99 0853 1 | REQUIRE 'RTLIN:RTLPSECT';         ! Define DECLARE_PSECTS macro
100 0948 1 |
101 0949 1 | LIBRARY 'RTLSTARLE';             ! Starlet system macros
102 0950 1 |
103 0951 1 |
104 0952 1 | MACROS:
105 0953 1 |
106 0954 1 |     NONE
107 0955 1 |
108 0956 1 | EQUATED SYMBOLS:
109 0957 1 |
110 0958 1 |     NONE
111 0959 1 |
112 0960 1 |
113 0961 1 | PSECT DECLARATIONS:
114 0962 1 |
115 0963 1 | DECLARE_PSECTS (BAS);
116 0964 1 |
117 0965 1 | OWN STORAGE:
118 0966 1 |
119 0967 1 |     NONE
120 0968 1 |
121 0969 1 | EXTERNAL REFERENCES:
122 0970 1 |
123 0971 1 |
```



```

: 124 0972 1 EXTERNAL ROUTINE
: 125 0973 1   BAS$$STOP_IO : NOVALUE,      ! Signal fatal BASIC I/O errors
: 126 0974 1   BAS$$STOP- : NOVALUE,      ! Signal errors
: 127 0975 1   BAS$$REC_PSE : JSB_PUT NOVALUE, ! REC level processing - RMS interface
: 128 0976 1                                     ! PUT sequential
: 129 0977 1   BAS$$REC_PRE : JSB_PUT NOVALUE, ! REC level processing - RMS
: 130 0978 1                                     ! interface PUT relative
: 131 0979 1   BAS$$OPEN_ZERO : NOVALUE,    ! Open channel zero
: 132 0980 1   BAS$$CB_PUSH : JSB_CB_PUSH NOVALUE, ! Load register CCB
: 133 0981 1   BAS$$CB_POP : JSB_CB_POP NOVALUE; ! Done with register CCB
: 134 0982 1
: 135 0983 1 !+
: 136 0984 1 ! The following are the error codes used in this module.
: 137 0985 1 !-
: 138 0986 1
: 139 0987 1 EXTERNAL LITERAL
: 140 0988 1   BAS$K_ILLILLACC : UNSIGNED (8), ! Illegal or illogical access
: 141 0989 1   BAS$K_ILLIO_CHA : UNSIGNED (8), ! Illegal I/O channel
: 142 0990 1   BAS$K_IO_CHANOT : UNSIGNED (8); ! I/O channel not open
: 143 0991 1

```



```
145 0992 1 GLOBAL ROUTINE BASSPUT (          ! PUT sequential
146 0993 1     UNIT                          ! logical unit number
147 0994 1     ) : NOVALUE =
148 0995 1
149 0996 1 ++
150 0997 1 FUNCTIONAL DESCRIPTION:
151 0998 1
152 0999 1     This routine will set up the I/O data base for this LUN if necessary
153 1000 1     and then go to the REC level directly. When control is returned to
154 1001 1     this routine, it pops the CCB off of the I/O system. The actual inter-
155 1002 1     face to RMS is done at the REC level. One record is written.
156 1003 1
157 1004 1 FORMAL PARAMETERS:
158 1005 1
159 1006 1     UNIT.rlu.v      logical unit number
160 1007 1
161 1008 1 IMPLICIT INPUTS:
162 1009 1
163 1010 1     LUB$V_READ_ONLY      file is read only
164 1011 1     LUB$V_VA_USE        indicates virtual array usage
165 1012 1
166 1013 1 IMPLICIT OUTPUTS:
167 1014 1
168 1015 1     LUB$V_BLK_USE        indicates non-virtual array usage
169 1016 1
170 1017 1 COMPLETION CODES:
171 1018 1
172 1019 1     NONE
173 1020 1
174 1021 1 SIDE EFFECTS:
175 1022 1
176 1023 1     Signals:
177 1024 1     BASSK_IO_CHANOT and
178 1025 1     BASSK_ILC_IO_CHA for foreign buffers
179 1026 1     BASSK_ILL_ILC_ACC      illegal or illogical access
180 1027 1
181 1028 1 --
182 1029 1
183 1030 2 BEGIN
184 1031 2
185 1032 2 BUILTIN
186 1033 2     FP;
187 1034 2
188 1035 2 GLOBAL REGISTER
189 1036 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
190 1037 2
191 1038 2 LOCAL
192 1039 2     BUFFER_SIZE,
193 1040 2     FMP : REF BLOCK [, BYTE],
194 1041 2     ACTUAL_UNIT,          ! Unit number, without foreign buffer
195 1042 2     TEMP_RT1;            ! CCB for foreign buffer, or 0
196 1043 2
197 1044 2 ++
198 1045 2     If channel passed is less than zero then this is an error.
199 1046 2 --
200 1047 2
201 1048 2     IF (.UNIT LSS 0) THEN BASS$STOP (BASSK_ILLIO_CHA);
```



```
202 1049 2
203 1050 2 FMP = .FP;
204 1051 2
205 1052 2 !+
206 1053 2 Check for "foreign buffers". If the unit number exceeds 255 then a foreign
207 1054 2 buffer is specified. The foreign buffer is actually a unit number whose
208 1055 2 buffer is to be used to do the PUT. The "foreign buffer" unit
209 1056 2 is pushed to pick up the CB address which is passed to the REC level. Then
210 1057 2 the unit pointing to the file is pushed so that the CCB points to the log-
211 1058 2 ical unit which actually do the I/O. Upon return, the necessary RAB fields
212 1059 2 (USZ and UBF) have been restored and two CB POPs are done if necessary.
213 1060 2 Explicit use of channel zero e.g. GET #0, PUT #0.... are similar to
214 1061 2 foreign buffer in the sense that we use the buffer of input side of
215 1062 2 channel 0 to do the PUT but output side of channel 0 for other characteristics.
216 1063 2 -
217 1064 2 TEMP_R11 = 0;
218 1065 2 ACTUAL_UNIT = .UNIT;
219 1066 2 IF (.UNIT GTR LUB$K_LUN_MAX OR .UNIT EQL 0)
220 1067 2 THEN
221 1068 2 BEGIN
222 1069 2
223 1070 2 LOCAL
224 1071 2 FOREIGN_BUFFER;
225 1072 2
226 1073 2 IF .UNIT EQL 0
227 1074 2 THEN
228 1075 2
229 1076 2 !+
230 1077 2 This is a explicit channel 0 operation. Treat input side of channel zero
231 1078 2 as a foreign buffer.
232 1079 2 -
233 1080 2 BEGIN
234 1081 2 FOREIGN_BUFFER = LUB$K_LUN_INPU;
235 1082 2 ACTUAL_UNIT = LUB$K_LUN_BPRI;
236 1083 2 END
237 1084 2 ELSE
238 1085 2
239 1086 2 !+
240 1087 2 This is a regular foreign buffer operation.
241 1088 2 -
242 1089 2 BEGIN
243 1090 2 FOREIGN_BUFFER = .UNIT/BAS$K_LUN_MAX;
244 1091 2 ACTUAL_UNIT = .UNIT MOD BAS$K_LUN_MAX;
245 1092 2 END;
246 1093 2
247 1094 2 IF (.UNIT NEQ 0 AND .FOREIGN_BUFFER GTRU BAS$K_MAX_FOR_B) THEN BAS$$STOP (BAS$K_ILLIO_CHA);
248 1095 2
249 1096 2 BAS$$CB_PUSH (.FOREIGN_BUFFER, LUB$K_ILUN_MIN);
250 1097 2 CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
251 1098 2 BUFFER_SIZE = .CCB [LUB$W_RBUF_SIZE];
252 1099 2
253 1100 2 IF ( NOT .CCB [LUB$V_OPENED] AND .UNIT NEQ 0) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
254 1101 2
255 1102 2 TEMP_R11 = .CCB;
256 1103 2 END;
257 1104 2
258 1105 2 BAS$$CB_PUSH (.ACTUAL_UNIT, LUB$K_ILUN_MIN);
```



```
259 1106 2 CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
260 1107 2
261 1108 2 + If we are on a default unit (unit number less than zero) then
262 1109 2 we can open it if it is not already open. Otherwise it must
263 1110 2 be open already.
264 1111 2 -
265 1112 2
266 1113 2 IF ( NOT .CCB [LUB$V_OPENED])
267 1114 2 THEN
268 1115 2
269 1116 2 IF (.ACTUAL_UNIT LSS 0)
270 1117 2 THEN
271 1118 2 BEGIN
272 1119 2 BASS$OPEN_ZERO (.FMP [SF$L_SAVE_FP])
273 1120 2 END
274 1121 2 ELSE
275 1122 2 BEGIN
276 1123 2 BASS$STOP_IO (BASS$K_IO_CHANOT);
277 1124 2 END;
278 1125 2
279 1126 2 IF (.TEMP_R11 EQA 0) THEN BUFFER_SIZE = .CCB [LUB$W_RBUF_SIZE];
280 1127 2
281 1128 2 +
282 1129 2 Now that the data base is in place, store the statement type and go
283 1130 2 directly to the REC level.
284 1131 2 -
285 1132 2 CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_PSE;
286 1133 2 +
287 1134 2 Check for virtual array usage and set block usage
288 1135 2 -
289 1136 2
290 1137 2 IF .CCB [LUB$V_VA_USE] OR .CCB [LUB$V_READ_ONLY] THEN BASS$STOP_IO (BASS$K_ILLILLACC);
291 1138 2
292 1139 2 CCB [LUB$V_BLK_USE] = 1;
293 1140 2 BASS$REC_PSE (.BUFFER_SIZE, .TEMP_R11);
294 1141 2 +
295 1142 2 Now that the PUT has been done, pop the CCB off the I/O system.
296 1143 2 -
297 1144 2 BASS$CB_POP ();
298 1145 2 +
299 1146 2 Pop the "foreign buffer" CB if necessary. It is kept on the CB stack until
300 1147 2 now to guard against an AST closing the foreign buffer channel.
301 1148 2 -
302 1149 2
303 1150 2 IF (.TEMP_R11 NEQA 0)
304 1151 2 THEN
305 1152 2 BEGIN
306 1153 2 CCB = .TEMP_R11;
307 1154 2 BASS$CB_POP ();
308 1155 2 END;
309 1156 2
310 1157 1 END;
```

!End of BASSPUT

.TITLE BASSPUT
.IDENT \1-011\

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
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

BAS\$PUT
1-011

B 2
16-Sep-1984 00:59:27 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:56:32 [BASRTL.SRC]BASPUT.B32;1

Page 9
(3)

		0C	A3	DD	000A9	PUSHL	12(FMP)	:	1119
	00000000G	00	01	FB	000AC	CALLS	#1, BAS\$\$OPEN_ZERO	:	
			07	11	000B3	BRB	9\$	:	1118
	7E	00G	8F	9A	000B5	MOVZBL	#BAS\$K_IO_CHANOT, -(SP)	:	1123
	68		01	FB	000B9	CALLS	#1, BAS\$\$STOP_IO	:	
			56	D5	000BC	TSTL	TEMP_R11	:	1126
			04	12	000BE	BNEQ	10\$	:	
	57	D2	AB	3C	000C0	MOVZWL	-46(CCB), BUFFER_SIZE	:	
	FF71	CB	1D	90	000C4	MOVB	#29, -143(CCB)	:	1132
		FF	AB	E8	000C9	BLBS	-1(CCB), 11\$	:	1137
07	FC	AB	02	E1	000CD	BBC	#2, -4(CCB), 12\$	:	
		00G	8F	9A	000D2	MOVZBL	#BAS\$K_ILLILLACC, -(SP)	:	
			01	FB	000D6	CALLS	#1, BAS\$\$STOP_IO	:	
	FF	AB	02	88	000D9	BISB2	#2, -1(CCB)	:	1139
		50	56	7D	000DD	MOVQ	TEMP_R11, R0	:	1140
		00000000G	00	16	000E0	JSB	BAS\$\$REC_PSE	:	
		00000000G	00	16	000E6	JSB	BAS\$\$CB_POP	:	1144
			56	D5	000EC	TSTL	TEMP_R1T	:	1150
			09	13	000EE	BEQL	13\$	:	
	5B		56	D0	000F0	MOVL	TEMP_R11, CCB	:	1153
		00000000G	00	16	000F3	JSB	BAS\$\$CB_POP	:	1154
			04	000F9	13\$:	RET		:	1157

; Routine Size: 250 bytes, Routine Base: _BAS\$CODE + 0000

; 311 1158 1


```
313 1159 1 GLOBAL ROUTINE BASSPUT_RECORD (          ! PUT sequential
314 1160 1     UNIT,                                ! logical unit number
315 1161 1     RECORD_NUM                          ! relative record number
316 1162 1 ) : NOVALUE =
317 1163 1
318 1164 1 ++
319 1165 1 FUNCTIONAL DESCRIPTION:
320 1166 1
321 1167 1     This routine will set up the I/O data base for this LUN if necessary
322 1168 1     and then go to the REC level directly. When control is returned to
323 1169 1     this routine, it pops the CCB off of the I/O system. The actual inter-
324 1170 1     face to RMS is done at the REC level. One record is written.
325 1171 1
326 1172 1 FORMAL PARAMETERS:
327 1173 1
328 1174 1     UNIT.rlu.v          logical unit number
329 1175 1     RECORD_NUM.rl.v   relative record number
330 1176 1
331 1177 1 IMPLICIT INPUTS:
332 1178 1
333 1179 1     LUB$V_VA_USE          indicates virtual array usage
334 1180 1     LUB$V_READ_ONLY      file is read only
335 1181 1
336 1182 1 IMPLICIT OUTPUTS:
337 1183 1
338 1184 1     LUB$V_BLK_USE          indicates non-virtual I/O usage
339 1185 1
340 1186 1 COMPLETION CODES:
341 1187 1
342 1188 1     NONE
343 1189 1
344 1190 1 SIDE EFFECTS:
345 1191 1
346 1192 1     Signals:
347 1193 1     BASSK_ILLIO CHA and
348 1194 1     BASSK_IO_CHANOT for foreign buffers
349 1195 1     BASSK_ILCILLACC
350 1196 1
351 1197 1 --
352 1198 1
353 1199 2 BEGIN
354 1200 2
355 1201 2 BUILTIN
356 1202 2     FP;
357 1203 2
358 1204 2 GLOBAL REGISTER
359 1205 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
360 1206 2
361 1207 2 LOCAL
362 1208 2     BUFFER_SIZE,
363 1209 2     FMP : REF BLOCK [, BYTE],
364 1210 2     ACTUAL_UNIT,          ! Unit number, without foreign buffer
365 1211 2     TEMP_RT1;             ! CCB for foreign buffer, or 0
366 1212 2
367 1213 2     FMP = .FP;
368 1214 2 ++
369 1215 2 ! Check for "foreign buffers". If the unit number exceeds 255 then a foreign
```



```
370 1216 2 | buffer is specified. The foreign buffer is actually a unit number whose
371 1217 2 | buffer is to receive the record which is read. The "foreign buffer" unit
372 1218 2 | is pushed to pick up the CB address which is passed to the REC level. Then
373 1219 2 | the unit pointing to the file is pushed so that the CCB points to the log-
374 1220 2 | ical unit which actually do the I/O. Upon return, the necessary RAB fields
375 1221 2 | (USZ and UBF) have been restored and two CB_POPs are done if necessary.
376 1222 2 |
377 1223 2 | TEMP R11 = 0;
378 1224 2 | ACTUAL_UNIT = .UNIT;
379 1225 2 |
380 1226 2 | IF (.UNIT GTR LUB$K_LUN_MAX)
381 1227 2 | THEN
382 1228 2 | BEGIN
383 1229 2 |
384 1230 2 | LOCAL
385 1231 2 | FOREIGN_BUFFER;
386 1232 2 |
387 1233 2 | FOREIGN_BUFFER = .UNIT/BAS$K_LUN_MAX;
388 1234 2 | ACTUAL_UNIT = .UNIT MOD BAS$K_LUN_MAX;
389 1235 2 |
390 1236 2 | IF (.FOREIGN_BUFFER GTRU BAS$K_MAX_FOR_B) THEN BAS$$STOP (BAS$K_ILLIO_CHA);
391 1237 2 |
392 1238 2 | BAS$$CB_PUSH (.FOREIGN_BUFFER, LUB$K_LUN_MIN);
393 1239 2 | CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
394 1240 2 |
395 1241 2 | IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
396 1242 2 |
397 1243 2 | BUFFER_SIZE = .CCB [LUB$W_RBUF_SIZE];
398 1244 2 | TEMP_RT1 = .CCB;
399 1245 2 | END;
400 1246 2 |
401 1247 2 | BAS$$CB_PUSH (.ACTUAL_UNIT, LUB$K_ILUN_MIN);
402 1248 2 | CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
403 1249 2 |
404 1250 2 | IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
405 1251 2 |
406 1252 2 | IF (.TEMP_R11 EQLA 0) THEN BUFFER_SIZE = .CCB [LUB$W_RBUF_SIZE];
407 1253 2 |
408 1254 2 | +
409 1255 2 | Now that the data base is in place, store the statement type, store the key, and go
410 1256 2 | directly to the REC level.
411 1257 2 | -
412 1258 2 | CCB [LUB$L_LOG_RECNO] = .RECORD_NUM;
413 1259 2 | CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_PRE;
414 1260 2 | +
415 1261 2 | Check for virtual array usage and set block usage
416 1262 2 | -
417 1263 2 |
418 1264 2 | IF .CCB [LUB$V_VA_USE] OR .CCB [LUB$V_READ_ONLY] THEN BAS$$STOP_IO (BAS$K_ILLILLACC);
419 1265 2 |
420 1266 2 | CCB [LUB$V_BLK_USE] = 1;
421 1267 2 | BAS$$REC_PRE (.BUFFER_SIZE, .TEMP_R11);
422 1268 2 | +
423 1269 2 | Now that the PUT has been done, pop the CCB off the I/O system.
424 1270 2 | -
425 1271 2 | BAS$$CB_POP ();
426 1272 2 | +
```



```

: 427      1273 2  ! Pop the "foreign buffer" CB if necessary. It is kept on the CB stack until
: 428      1274 2  ! now to guard against an AST closing the foreign buffer channel.
: 429      1275 2  !-
: 430      1276 2
: 431      1277 3      IF (.TEMP_R11 NEQA 0)
: 432      1278 3      THEN
: 433      1279 3          BEGIN
: 434      1280 3              CCB = .TEMP_R11;
: 435      1281 3              BASS$CB_POP-();
: 436      1282 3          END;
: 437      1283 2
: 438      1284 1      END;

```

!End of BASSPUT

				OBFC 00000		.ENTRY	BASSPUT_RECORD, Save R2,R3,R4,R5,R6,R7,R8,-		1159
				59 00000000G	00 9E 00002	MOVAB	R9,R11		
				58 00000000G	00 9E 00009	MOVAB	BASS\$CB_POP, R9		
				57 00000000G	00 9E 00010	MOVAB	BASS\$CB_PUSH, R8		
				53	5D D0 00017	MOVAB	BASS\$STOP_IO, R7		
					55 D4 0001A	MOVL	FP, FMP		1213
					54	CLRL	TEMP_R11		1223
					AC D0 0001C	MOVL	UNIT, ACTUAL_UNIT		1224
					AC D1 00020	CMPL	UNIT, #119		1226
					48 15 00028	BLEQ	3\$		
					8F C7 0002A	DIVL3	#256, UNIT, FOREIGN_BUFFER		1233
					01 7A 00033	EMUL	#1, UNIT, #0, -(SP)		1234
					8F 7B 00039	EDIV	#256, (SP)+, ACTUAL_UNIT, ACTUAL_UNIT		
					52 D1 00042	CMPL	FOREIGN_BUFFER, #127		1236
					0B 1B 00049	BLEQU	1\$		
					8F 9A 0004B	MOVZBL	#BASS\$K_ILLIO_CHA, -(SP)		
					01 FB 0004F	CALLS	#1, BASS\$STOP		
					50 D4 00056	CLRL	R0		1238
					68 16 00058	JSB	BASS\$CB_PUSH		
					A3 D0 0005A	MOVL	12(FMP), -180(CCB)		1239
					AB E8 00060	BLBS	-4(CCB), 2\$		1241
					8F 9A 00064	MOVZBL	#BASS\$K_IO_CHANOT, -(SP)		
					01 FB 00068	CALLS	#1, BASS\$STOP_IO		
					AB 3C 0006B	MOVZWL	-46(CCB), BUFFER_SIZE		1243
					5B D0 0006F	MOVL	CCB, TEMP_R11		1244
					08 CE 00072	MNEGL	#8, R0		1247
					54 D0 00075	MOVL	ACTUAL_UNIT, R2		
					68 16 00078	JSB	BASS\$CB_PUSH		
					A3 D0 0007A	MOVL	12(FMP), -180(CCB)		1248
					AB E8 00080	BLBS	-4(CCB), 4\$		1250
					8F 9A 00084	MOVZBL	#BASS\$K_IO_CHANOT, -(SP)		
					01 FB 00088	CALLS	#1, BASS\$STOP_IO		
					55 D5 0008B	TSTL	TEMP_R11		1252
					04 12 0008D	BNEQ	5\$		
					AB 3C 0008F	MOVZWL	-46(CCB), BUFFER_SIZE		
					AC D0 00093	MOVL	RECORD_NUM, -32(CCB)		1258
					27 90 00098	MOVAB	#39, -T43(CCB)		1259
					AB E8 0009D	BLBS	-1(CCB), 6\$		1264
					02 E1 000A1	BBC	#2, -4(CCB), 7\$		
					8F 9A 000A6	MOVZBL	#BASS\$K_ILLILLACC, -(SP)		

BASSPUT
1-011

F 2
16-Sep-1984 00:59:27
14-Sep-1984 11:56:32

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

Page 13
(4)

FF	67	01	FB	000AA	CALLS	#1, BASS\$STOP_IO	:	
	AB	02	88	000AD	BISB2	#2, -1(CCB)	:	1266
	50	55	7D	000B1	MOVQ	TEMP_R11, R0	:	1267
		00	16	000B4	JSB	BASS\$REC_PRE	:	
	00000000G	69	16	000BA	JSB	BASS\$CB_POP	:	1271
		55	D5	000BC	TSTL	TEMP_R1T	:	1277
		05	13	000BE	BEQL	8\$	:	
	5B	55	D0	000C0	MOVL	TEMP_R11, CCB	:	1280
		69	16	000C3	JSB	BASS\$CB_POP	:	1281
		04	00	000C5	RET		:	1284

; Routine Size: 198 bytes, Routine Base: _BASS\$CODE + 00FA

; 439 1285 1


```

441 1286 1 GLOBAL ROUTINE BASS$PUT_COUNT (      ! PUT sequential with count
442 1287 1     UNIT,                          ! logical unit number
443 1288 1     COUNT,                          ! No. of bytes in record
444 1289 1     ) : NOVALUE =
445 1290 1
446 1291 1 ++
447 1292 1 FUNCTIONAL DESCRIPTION:
448 1293 1
449 1294 1     This routine will set up the I/O data base for this LUN if necessary
450 1295 1     and then go to the REC level directly. When control is returned to
451 1296 1     this routine, it pops the CCB off of the I/O system. The actual inter-
452 1297 1     face to RMS is done at the REC level. One record is written.
453 1298 1
454 1299 1 FORMAL PARAMETERS:
455 1300 1
456 1301 1     UNIT.rlu.v      logical unit number
457 1302 1     COUNT.rl.v     no.of bytes in record
458 1303 1
459 1304 1 IMPLICIT INPUTS:
460 1305 1
461 1306 1     LUB$V_READ_ONLY    file is read only
462 1307 1     LUB$V_VA_USE      virtual array I/O usage
463 1308 1
464 1309 1 IMPLICIT OUTPUTS:
465 1310 1
466 1311 1     LUB$V_BLK_USE      non-virtual array I/O usage
467 1312 1
468 1313 1 COMPLETION CODES:
469 1314 1
470 1315 1     NONE
471 1316 1
472 1317 1 SIDE EFFECTS:
473 1318 1
474 1319 1     Signals:
475 1320 1     BASS$K_IO_CHANOT and
476 1321 1     BASS$K_ILC_IO_CHA for foreign buffers
477 1322 1     BASS$K_ILLLILCACC
478 1323 1
479 1324 1 --
480 1325 1
481 1326 2 BEGIN
482 1327 2
483 1328 2 BUILTIN
484 1329 2     FP;
485 1330 2
486 1331 2 GLOBAL REGISTER
487 1332 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
488 1333 2
489 1334 2 LOCAL
490 1335 2     FMP : REF BLOCK [, BYTE],
491 1336 2     ACTUAL_UNIT,
492 1337 2     TEMP_RT1;
493 1338 2
494 1339 2     FMP = .FP;
495 1340 2 ++
496 1341 2 ! Check for "foreign buffers". If the unit number exceeds 255 then a foreign
497 1342 2 ! buffer is specified. The foreign buffer is actually a unit number whose

```



```

498 1343 2 1 buffer is used for the PUT operation. The "foreign buffer" unit
499 1344 2 1 is pushed to pick up the CB address which is passed to the REC level. Then
500 1345 2 1 the unit pointing to the file is pushed so that the CCB points to the log-
501 1346 2 1 ical unit which actually do the I/O. Upon return, the necessary RAB fields
502 1347 2 1 (USZ and UBF) have been restored and two CB POPs are done if necessary.
503 1348 2 1 Explicit channel zero operation is similar to foreign buffers in the sense
504 1349 2 1 that it uses input side of channel zero for buffer but output side of channel
505 1350 2 1 zero for everything else.
506 1351 2 1 -
507 1352 2 1     TEMP_R11 = 0;
508 1353 2 1     ACTUAL_UNIT = .UNIT;
509 1354 2 1
510 1355 2 1     IF (.UNIT GTR LUB$K_LUN_MAX OR .UNIT EQL 0)
511 1356 2 1     THEN
512 1357 2 1         BEGIN
513 1358 2 1
514 1359 2 1             LOCAL
515 1360 2 1                 FOREIGN_BUFFER;
516 1361 2 1
517 1362 2 1             IF .UNIT EQL 0
518 1363 2 1             THEN
519 1364 2 1
520 1365 2 1         +
521 1366 2 1         1 This is a explicit channel 0 operation. Treat input side of channel zero
522 1367 2 1         1 as a foreign buffer.
523 1368 2 1         1 -
524 1369 2 1             BEGIN
525 1370 2 1                 FOREIGN_BUFFER = LUB$K_LUN_INPU;
526 1371 2 1                 ACTUAL_UNIT = LUB$K_LUN_BPRI;
527 1372 2 1             END
528 1373 2 1         ELSE
529 1374 2 1
530 1375 2 1         +
531 1376 2 1         1 This is a regular foreign buffer operation.
532 1377 2 1         1 -
533 1378 2 1             BEGIN
534 1379 2 1                 FOREIGN_BUFFER = .UNIT/BAS$K_LUN_MAX;
535 1380 2 1                 ACTUAL_UNIT = .UNIT MOD BAS$K_LUN_MAX;
536 1381 2 1             END;
537 1382 2 1
538 1383 2 1             IF (.UNIT NEQ 0 AND .FOREIGN_BUFFER GTRU BAS$K_MAX_FOR_B) THEN BAS$$STOP (BAS$K_ILLIO_CHA);
539 1384 2 1
540 1385 2 1             BAS$$CB PUSH (.FOREIGN_BUFFER, LUB$K_ILUN_MIN);
541 1386 2 1             CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
542 1387 2 1
543 1388 2 1             IF ( NOT .CCB [LUB$V_OPENED] AND .UNIT NEQ 0) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
544 1389 2 1
545 1390 2 1             TEMP_R11 = .CCB;
546 1391 2 1             END;
547 1392 2 1
548 1393 2 1             BAS$$CB PUSH (.ACTUAL_UNIT, LUB$K_ILUN_MIN);
549 1394 2 1             CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
550 1395 2 1
551 1396 2 1         +
552 1397 2 1         1 If we are on a default unit (unit number less than zero) then
553 1398 2 1         1 we can open it if it is not already open. Otherwise it must
554 1399 2 1         1 be open already.

```



```
555 1400 2
556 1401 3 IF ( NOT .CCB [LUB$V_OPENED])
557 1402 2 THEN
558 1403 2
559 1404 3 IF (.ACTUAL_UNIT LSS 0)
560 1405 2 THEN
561 1406 3 BEGIN
562 1407 3 BASS$OPEN_ZERO (.FMP [SF$L_SAVE_FP])
563 1408 2 END
564 1409 2 ELSE
565 1410 3 BEGIN
566 1411 3 BASS$STOP_IO (BASS$K_IO_CHANOT);
567 1412 2 END;
568 1413 2
569 1414 2 +
570 1415 2 Now that the data base is in place, store the statement type and go
571 1416 2 directly to the REC level.
572 1417 2 -
573 1418 2 CCB [ISB$B_STTM_TYPE] = ISB$K_ST_TY_PSE;
574 1419 2 +
575 1420 2 Check for virtual array usage and set block usage
576 1421 2 -
577 1422 2
578 1423 2 IF .CCB [LUB$V_VA_USE] OR .CCB [LUB$V_READ_ONLY] THEN BASS$STOP_IO (BASS$K_ILLILLACC);
579 1424 2
580 1425 2 CCB [LUB$V_BLK_USE] = 1;
581 1426 2 BASS$REC_PSE (.COUNT, .TEMP_R11);
582 1427 2 +
583 1428 2 Now that the PUT has been done, pop the CCB off the I/O system.
584 1429 2 -
585 1430 2 BASS$CB_POP ();
586 1431 2 +
587 1432 2 Pop the "foreign buffer" CB if necessary. It is kept on the CB stack until
588 1433 2 now to guard against an AST closing the foreign buffer channel.
589 1434 2 -
590 1435 2
591 1436 2 IF (.TEMP_R11 NEQA 0)
592 1437 2 THEN
593 1438 3 BEGIN
594 1439 3 CCB = .TEMP_R11;
595 1440 3 BASS$CB_POP-();
596 1441 2 END;
597 1442 2
598 1443 1 END;
```

!End of BASSPUT_COUNT

OBFC 00000				.ENTRY	BASSPUT_COUNT, Save R2,R3,R4,R5,R6,R7,R8,-	:	1286
59	00000000G	00	9E 00002	MOVAB	R9,R11	:	
58	00000000G	00	9E 00009	MOVAB	BASS\$CB_POP, R9	:	
57	00000000G	00	9E 00010	MOVAB	BASS\$CB_PUSH, R8	:	
53		5D	D0 00017	MOVAB	BASS\$STOP_IO, R7	:	
		56	D4 0001A	MOVL	FP, FMP	:	1339
50	04	AC	D0 0001C	CLRL	TEMP_R11	:	1352
				MOVL	UNIT, R0	:	1353

				50	D0	00020	MOVL	R0, ACTUAL_UNIT		
	00000077	54		50	D1	00023	CMPL	R0, #119	1355	
		8F		04	14	0002A	BGTR	1\$		
				50	D5	0002C	TSTL	R0		
				5A	12	0002E	BNEQ	6\$		
				50	D5	00030	TSTL	R0	1362	
				08	12	00032	BNEQ	2\$		
		52		07	CE	00034	MNEGL	#7, FOREIGN_BUFFER	1370	
		54		08	CE	00037	MNEGL	#8, ACTUAL_UNIT	1371	
				16	11	0003A	BRB	3\$	1362	
	52	50	00000100	8F	C7	0003C	DIVL3	#256, R0, FOREIGN_BUFFER	1379	
7E	00	50		01	7A	00044	EMUL	#1, R0, #0, -(SP)	1380	
54	54	8E	00000100	8F	7B	00049	EDIV	#256, (SP)+, ACTUAL_UNIT, ACTUAL_UNIT		
				55	D4	00052	CLRL	R5	1383	
				50	D5	00054	TSTL	R0		
				16	13	00056	BEQL	4\$		
				55	D6	00058	INCL	R5		
	0000007F	8F		52	D1	0005A	CMPL	FOREIGN_BUFFER, #127		
				0B	1B	00061	BLEQU	4\$		
		7E	00G	8F	9A	00063	MOVZBL	#BAS\$K_ILLIO_CHA, -(SP)		
	00000000G	00		01	FB	00067	CALLS	#1, BAS\$\$STOP		
		50		08	CE	0006E	MNEGL	#8, R0	1385	
				68	16	00071	JSB	BAS\$\$CB_PUSH		
	FF4C	CB	OC	A3	D0	00073	MOVL	12(FMP), -180(CCB)	1386	
		0A	FC	AB	E8	00079	BLBS	-4(CCB), 5\$	1388	
		07		55	E9	0007D	BLBC	R5, 5\$		
		7E	00G	8F	9A	00080	MOVZBL	#BAS\$K_IO_CHANOT, -(SP)		
		67		01	FB	00084	CALLS	#1, BAS\$\$STOP_IO		
		56		5B	D0	00087	MOVL	CCB, TEMP_R11	1390	
		50		08	CE	0008A	MNEGL	#8, R0	1393	
		52		54	D0	0008D	MOVL	ACTUAL_UNIT, R2		
				68	16	00090	JSB	BAS\$\$CB_PUSH		
	FF4C	CB	OC	A3	D0	00092	MOVL	12(FMP), -180(CCB)	1394	
		17	FC	AB	E8	00098	BLBS	-4(CCB), 8\$	1401	
				54	D5	0009C	TSTL	ACTUAL_UNIT	1404	
				0C	18	0009E	BGEQ	7\$		
			OC	A3	DD	000A0	PUSHL	12(FMP)	1407	
	00000000G	00		01	FB	000A3	CALLS	#1, BAS\$\$OPEN_ZERO		
				07	11	000AA	BRB	8\$	1406	
		7E	00G	8F	9A	000AC	MOVZBL	#BAS\$K_IO_CHANOT, -(SP)	1411	
		67		01	FB	000B0	CALLS	#1, BAS\$\$STOP_IO		
	FF71	CB		1D	90	000B3	MOVB	#29, -143(CCB)	1418	
		05	FF	AB	E8	000B8	BLBS	-1(CCB), 9\$	1423	
		FC		02	E1	000BC	BBC	#2, -4(CCB), 10\$		
07		7E	00G	8F	9A	000C1	MOVZBL	#BAS\$K_ILLILLACC, -(SP)		
		67		01	FB	000C5	CALLS	#1, BAS\$\$STOP_IO		
		AB		02	88	000C8	BISB2	#2, -1(CCB)	1425	
	FF	50		56	D0	000CC	MOVL	TEMP_R11, R0	1426	
		51	08	AC	D0	000CF	MOVL	COUNT, R1		
			00000000G	00	16	000D3	JSB	BAS\$\$REC_PSE		
				69	16	000D9	JSB	BAS\$\$CB_POP	1430	
				56	D5	000DB	TSTL	TEMP_R11	1436	
				05	13	000DD	BEQL	11\$		
		5B		56	D0	000DF	MOVL	TEMP_R11, CCB	1439	
				69	16	000E2	JSB	BAS\$\$CB_POP	1440	
				04	000E4	11\$:	RET		1443	

BASSPUT
1-011

K 2
16-Sep-1984 00:59:27
14-Sep-1984 11:56:32

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

Page 18
(5)

; Routine Size: 229 bytes, Routine Base: _BASSCODE + 01C0

; 599 1444 1


```
601 1445 1 GLOBAL ROUTINE BASSPUT_REC_CNT (      ! PUT relative with count
602 1446 1      UNIT,                          ! logical unit number
603 1447 1      RECORD_NUM,                    ! relative record number
604 1448 1      COUNT                          ! No. of bytes in record
605 1449 1      ) : NOVALUE =
606 1450 1
607 1451 1 !++
608 1452 1 FUNCTIONAL DESCRIPTION:
609 1453 1
610 1454 1      This routine will set up the I/O data base for this LUN if necessary
611 1455 1      and then go to the REC level directly. When control is returned to
612 1456 1      this routine, it pops the CCB off of the I/O system. The actual inter-
613 1457 1      face to RMS is done at the REC level. One record is written.
614 1458 1
615 1459 1 FORMAL PARAMETERS:
616 1460 1
617 1461 1      UNIT.rlu.v      logical unit number
618 1462 1      RECORD_NUM.rl.v  relative record number
619 1463 1      COUNT.fl.v      no.of bytes in record
620 1464 1
621 1465 1 IMPLICIT INPUTS:
622 1466 1
623 1467 1      LUB$V_READ_ONLY    file is read only
624 1468 1      LUB$V_VA_USE      virtual array usage
625 1469 1
626 1470 1 IMPLICIT OUTPUTS:
627 1471 1
628 1472 1      LUB$V_BLK_USE      non-virtual array use
629 1473 1
630 1474 1 COMPLETION CODES:
631 1475 1
632 1476 1      NONE
633 1477 1
634 1478 1 SIDE EFFECTS:
635 1479 1
636 1480 1      Signals:
637 1481 1      BASSK_ILLO CHA and
638 1482 1      BASSK_IO CHANOT for foreign buffers
639 1483 1      BASSK_ILCILLACC
640 1484 1
641 1485 1 !--
642 1486 1
643 1487 2 BEGIN
644 1488 2
645 1489 2 BUILTIN
646 1490 2     FP;
647 1491 2
648 1492 2 GLOBAL REGISTER
649 1493 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
650 1494 2
651 1495 2 LOCAL
652 1496 2     FMP : REF BLOCK [, BYTE],
653 1497 2     ACTUAL_UNIT,
654 1498 2     TEMP_RT1;
655 1499 2
656 1500 2     FMP = .FP;
657 1501 2 !+
```

! Unit number, without foreign buffer
! CCB for foreign buffer, or 0


```

658 1502 2 | Check for "foreign buffers". If the unit number exceeds 255 then a foreign
659 1503 2 | buffer is specified. The foreign buffer is actually a unit number whose
660 1504 2 | buffer is to receive the record which is read. The "foreign buffer" unit
661 1505 2 | is pushed to pick up the CB address which is passed to the REC level. Then
662 1506 2 | the unit pointing to the file is pushed so that the CCB points to the log-
663 1507 2 | ical unit which actually do the I/O. Upon return, the necessary RAB fields
664 1508 2 | (USZ and UBF) have been restored and two CB_POPs are done if necessary.
665 1509 2 |
666 1510 2 | TEMP_R11 = 0;
667 1511 2 | ACTUAL_UNIT = .UNIT;
668 1512 2 |
669 1513 2 | IF (.UNIT GTR LUB$K_LUN_MAX)
670 1514 2 | THEN
671 1515 2 | BEGIN
672 1516 2 |
673 1517 2 | LOCAL
674 1518 2 | FOREIGN_BUFFER;
675 1519 2 |
676 1520 2 | FOREIGN_BUFFER = .UNIT/BASS$K_LUN_MAX;
677 1521 2 | ACTUAL_UNIT = .UNIT MOD BASS$K_LUN_MAX;
678 1522 2 |
679 1523 2 | IF (.FOREIGN_BUFFER GTRU BASS$K_MAX_FOR_B) THEN BASS$$STOP (BASS$K_ILLIO_CHA);
680 1524 2 |
681 1525 2 | BASS$CB PUSH (.FOREIGN_BUFFER, LUB$K_LUN_MIN);
682 1526 2 | CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
683 1527 2 |
684 1528 2 | IF ( NOT .CCB [LUB$V_OPENED]) THEN BASS$$STOP_IO (BASS$K_IO_CHANOT);
685 1529 2 |
686 1530 2 | TEMP_R11 = .CCB;
687 1531 2 | END;
688 1532 2 |
689 1533 2 | BASS$CB PUSH (.ACTUAL_UNIT, LUB$K_ILUN_MIN);
690 1534 2 | CCB [ISB$A_USER_FP] = .FMP [SF$L_SAVE_FP];
691 1535 2 |
692 1536 2 | IF ( NOT .CCB [LUB$V_OPENED]) THEN BASS$$STOP_IO (BASS$K_IO_CHANOT);
693 1537 2 |
694 1538 2 | +
695 1539 2 | Now that the data base is in place, store the statement type, store the index, and go
696 1540 2 | directly to the REC level.
697 1541 2 | -
698 1542 2 | CCB [LUB$L_LOG_RECNO] = .RECORD_NUM;
699 1543 2 | CCB [ISB$B_STMT_TYPE] = ISB$K_ST_TY_PRE;
700 1544 2 | +
701 1545 2 | Check for virtual array usage and set block usage
702 1546 2 | -
703 1547 2 |
704 1548 2 | IF .CCB [LUB$V_VA_USE] OR .CCB [LUB$V_READ_ONLY] THEN BASS$$STOP_IO (BASS$K_ILLILLACC);
705 1549 2 |
706 1550 2 | CCB [LUB$V_BLK_USE] = 1;
707 1551 2 | BASS$REC_PRE (.COUNT, .TEMP_R11);
708 1552 2 | +
709 1553 2 | Now that the PUT has been done, pop the CCB off the I/O system.
710 1554 2 | -
711 1555 2 | BASS$CB_POP ();
712 1556 2 | +
713 1557 2 | Pop the "foreign buffer" CB if necessary. It is kept on the CB stack until
714 1558 2 | now to guard against an AST closing the foreign buffer channel.
```


BASSPUT
1-011

N 2
16-Sep-1984 00:59:27
14-Sep-1984 11:56:32

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

Page 21
(6)

```

: 715      1559 2 :-
: 716      1560
: 717      1561 IF (.TEMP_R11 NEQA 0)
: 718      1562 THEN
: 719      1563 BEGIN
: 720      1564 CCB = .TEMP_R11;
: 721      1565 BASS$CB_POP-();
: 722      1566 END;
: 723      1567
: 724      1568 1 END;
```

!End of BASSPUT_REC_CNT

				09FC 00000	.ENTRY	BASSPUT_REC_CNT, Save R2,R3,R4,R5,R6,R7,R8,-;	
				58 00000000G 00 9E 00002	MOVAB	R11	1445
				57 00000000G 00 9E 00009	MOVAB	BASS\$CB_POP, R8	
				56 00000000G 00 9E 00010	MOVAB	BASS\$CB_PUSH, R7	
				53 5D D0 00017	MOVAB	BASS\$STOP_IO, R6	
				55 D4 0001A	MOVL	FP, FMP	1500
				54 04 AC D0 0001C	CLRL	TEMP_R11	1510
				8F 04 AC D1 00020	MOVL	UNIT, ACTUAL_UNIT	1511
				44 15 00028	CMPL	UNIT, #119	1513
				8F C7 0002A	BLEQ	3\$	
				01 7A 00033	DIVL3	#256, UNIT, FOREIGN_BUFFER	1520
				8F 7B 00039	EMUL	#1, UNIT, #0, -(SP)	1521
				52 D1 00042	EDIV	#256, (SP)+, ACTUAL_UNIT, ACTUAL_UNIT	
				0B 1B 00049	CMPL	FOREIGN_BUFFER, #127	1523
				8F 9A 0004B	BLEQU	1\$	
				01 FB 0004F	MOVZBL	#BASSK ILLIO CHA, -(SP)	
				50 D4 00056 1\$:	CALLS	#1, BASS\$STOP	
				67 16 00058	CLRL	R0	1525
				A3 D0 0005A	JSB	BASS\$CB_PUSH	
				AB E8 00060	MOVL	12(FMP), -180(CCB)	1526
				8F 9A 00064	BLBS	-4(CCB), 2\$	1528
				01 FB 00068	MOVZBL	#BASSK IO CHANOT, -(SP)	
				5B D0 0006B 2\$:	CALLS	#1, BASS\$STOP_IO	
				08 CE 0006E 3\$:	MOVL	CCB, TEMP_R11	1530
				54 D0 00071	MNEGL	#8, R0	1533
				67 16 00074	MOVL	ACTUAL_UNIT, R2	
				A3 D0 00076	JSB	BASS\$CB_PUSH	
				AB E8 0007C	MOVL	12(FMP), -180(CCB)	1534
				8F 9A 00080	BLBS	-4(CCB), 4\$	1536
				01 FB 00084	MOVZBL	#BASSK IO CHANOT, -(SP)	
				AC D0 00087 4\$:	CALLS	#1, BASS\$STOP_IO	
				27 90 0008C	MOVL	RECORD_NUM, -32(CCB)	1542
				AB E8 00091	MOVB	#39, -T43(CCB)	1543
				02 E1 00095	BLBS	-1(CCB), 5\$	1548
				8F 9A 0009A 5\$:	BBC	#2, -4(CCB), 6\$	
				01 FB 0009E 6\$:	MOVZBL	#BASSK ILLILLACC, -(SP)	
				55 D0 000A5	CALLS	#1, BASS\$STOP_IO	
				AC D0 000AB	BISB2	#2, -1(CCB)	1550
				00 16 000AC	MOVL	TEMP_R11, R0	1551
				68 16 000B2	MOVL	COUNT, R1	
					JSB	BASS\$REC_PRE	
					JSB	BASS\$CB_POP	1555

BAS\$PUT
1-011

B 3
16-Sep-1984 00:59:27 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 11:56:32 [BASRTL.SRC]BASPUT.B32;1

Page 22
(6)

5B	55 D5 000B4	TSTL	TEMP_R11	: 1561
	05 13 000B6	BEQL	7\$	: 1564
	55 D0 000B8	MOVL	TEMP_R11, CCB	: 1565
	68 16 000BB	JSB	BAS\$\$CB_POP	: 1568
	04 000BD 7\$:	RET		

; Routine Size: 190 bytes, Routine Base: _BAS\$CODE + 02A5

: 725 1569 1
: 726 1570 1 END
: 727 1571 1
: 728 1572 0 ELUDOM

!End of module - BAS\$PUT

PSECT SUMMARY

Name	Bytes	Attributes
_BAS\$CODE	867	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	1	0	581	00:01.2

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/NOTRACE/LIS=LIS\$:BASPUT/OBJ=OBJ\$:BASPUT MSRC\$:BASPUT/UPDATE=(ENH\$:BASPUT)

; Size: 867 code + 0 data bytes
; Run Time: 00:19.7
; Elapsed Time: 00:45.6
; Lines/CPU Min: 4782
; Lexemes/CPU-Min: 27660
; Memory Used: 159 pages
; Compilation Complete

0030 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

