

FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFF	111111	111111	111111	XXX
FFF	111111	111111	111111	XXX
FFF	111111	111111	111111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFFFFFFFFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111	111	111	XXX
FFF	111111111	111111111	111111111	XXX
FFF	111111111	111111111	111111111	XXX
FFF	111111111	111111111	111111111	XXX

\*\*FILE\*\*ID\*\*PMS

C 15

PPPPPPPP	MM	MM	SSSSSSS
PPPPPPPP	MM	MM	SSSSSSS
PP PP	MMMM	MMMM	SS
PP PP	MMMM	MMMM	SS
PP PP	MM MM	MM	SS
PP PP	MM MM	MM	SS
PPPPPPPP	MM	MM	SSSSS
PPPPPPPP	MM	MM	SSSSS
PP	MM	MM	SS
PP	MM	MM	SS
PP	MM	MM	SS
PP	MM	MM	SS
PP	MM	MM	SSSSSSS
PP	MM	MM	SSSSSSS

....

LL		SSSSSSS
LL		SSSSSSS
LL		SS
LL		SS
LL		SS
LL		SSSSS
LL		SSSSS
LL		SS
LLLLLLLLL		SSSSSSS
LLLLLLLLL		SSSSSSS

```
1 0001 0 MODULE PMS (
2 0002 0
3 0003 0 LANGUAGE (BLISS32),
4 0004 0 IDENT = 'V04-000'
5 0005 1 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1 ****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 ****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 1
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This module contains the local performance measurement data base
38 0038 1 and the performance metering routines.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1 STARLET operating system, including privileged system services
43 0043 1 and internal exec routines.
44 0044 1
45 0045 1 --
46 0046 1
47 0047 1
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 25-Aug-1977 11:30
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V03-002 ACG0408 Andrew C. Goldstein, 23-Mar-1984 11:18
53 0053 1 Make rest of global storage based
54 0054 1
55 0055 1 V03-001 CDS0001 Christian D. Saether 12-Dec-1983
56 0056 1 Use L_NORM linkage and BIND COMMON macro.
57 0057 1 Move all OWN and GLOBAL declarations to the
```

58 0058 1 | COMMON module so that number of image sections can  
59 0059 1 | be reduced.  
60 0060 1 |  
61 0061 1 | X0100 ACG00001 Andrew C. Goldstein, 10-Oct-1978 20:00  
62 0062 1 | Previous revision history moved to [F11B.SRC]F11B.REV  
63 0063 1 | \*\*  
64 0064 1 |  
65 0065 1 |  
66 0066 1 LIBRARY 'SYSSLIBRARY:LIB.L32';  
67 0067 1 REQUIRE 'SRC\$:FCPDEF.B32';  
68 1058 1 |  
69 1059 1 |  
70 1060 1 FORWARD ROUTINE  
71 1061 1 PMS\_START : L\_NORM NOVALUE,! start measuring main function  
72 1062 1 PMS\_END : L\_NORM NOVALUE,! end measuring main function  
73 1063 1 PMS\_START\_SUB : L\_NORM NOVALUE,! start measuring subfunction  
74 1064 1 PMS\_END\_SUB : L\_NORM NOVALUE;! end measuring subfunction

```
76      1065 1 +
77      1066 1
78      1067 1 The performance measurement data base consists of the accounting array in
79      1068 1 system space, as well as some local storage to keep intermediate figures.
80      1069 1 The data accumulated per function (and also broken out for significant
81      1070 1 subfunctions) includes the number of functions executed, the number of
82      1071 1 modifiers, the number of disk reads, disk writes, and cache reads, the
83      1072 1 amount of CPU time, and the number of page faults.
84      1073 1 -
85      1074 1
86      1075 1
87      1076 1
88      1077 1 System space data array. Each measured parameter is contained in a vector
89      1078 1 with one entry per function.
90      1079 1
91      1080 1
92      1081 1 EXTERNAL
93      1082 1     PMSSGL_FCP2 : VECTOR ADDRESSING_MODE (ABSOLUTE);
94      1083 1           ! base of FCP measurement array
95      1084 1
96      1085 1 BIND
97      1086 1     PMSSGL_COUNT    = PMSSGL_FCP2 + 0      : VECTOR [10],
98      1087 1     PMSSGL_MCNT     = PMSSGL_FCP2 + 40     : VECTOR [10],
99      1088 1     PMSSGL_READ      = PMSSGL_FCP2 + 80     : VECTOR [10],
100     1089 1     PMSSGL_WRITE     = PMSSGL_FCP2 + 120    : VECTOR [10],
101     1090 1     PMSSGL_CACHE     = PMSSGL_FCP2 + 160    : VECTOR [10],
102     1091 1     PMSSGL_CPU       = PMSSGL_FCP2 + 200    : VECTOR [10],
103     1092 1     PMSSGL_PFA      = PMSSGL_FCP2 + 240    : VECTOR [10];
```

```
105      1093 1 GLOBAL ROUTINE PMS_START : L_NORM NOVALUE =
106
107
108
109      1095 1 ++
110      1096 1
111      1097 1 FUNCTIONAL DESCRIPTION:
112      1098 1     This routine initiates measurement for the main function being executed.
113      1101 1
114      1102 1 CALLING SEQUENCE:
115      1103 1     PMS_START ()
116      1104 1
117      1105 1 INPUT PARAMETERS:
118      1106 1     NONE
119      1107 1
120      1108 1 IMPLICIT INPUTS:
121      1109 1     NONE
122      1110 1
123      1111 1 OUTPUT PARAMETERS:
124      1112 1     NONE
125      1113 1
126      1114 1 IMPLICIT OUTPUTS:
127      1115 1     NONE
128      1116 1
129      1117 1 ROUTINE VALUE:
130      1118 1     NONE
131      1119 1
132      1120 1 SIDE EFFECTS:
133      1121 1     NONE
134      1122 1
135      1123 1 --
136      1124 1
137      1125 2 BEGIN
138
139      1126 2 LOCAL
140      1127 2     PROCESS_HEADER : REF BBLOCK; ! pointer to FCP process header
141
142      1128 2 BIND_COMMON;
143
144      1129 2 EXTERNAL
145      1130 2     CTL$GL_PHD : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
146      1131 2                         ! address of process header in control region
147
148      1132 2
149      1133 2
150      1134 2
151      1135 2
152      1136 2
153      1137 2
154      1138 2 To initialize measurement, we take copies of the running totals of all the
155      1139 2 parameters and stash them, so we can later compute the incremental usage.
156
157      1140 2
158      1141 2     PROCESS_HEADER = .CTL$GL_PHD; ! get address of own process header
159
160      1142 2
161      1143 2     PMS_FNC_READ = .PMS_TOT_READ;
162      1144 2     PMS_FNC_WRITE = .PMS_TOT_WRITE;
163      1145 2     PMS_FNC_CACHE = .PMS_TOT_CACHE;
164      1146 2     PMS_FNC_CPU = .PROCESS_HEADER[PHDSL_CPUTIM];
165      1147 2     PMS_FNC_PFA = .PROCESS_HEADER[PHDSL_PAGEFLTS];
166
167      1148 2
168      1149 1 END; ! end of routine PMS_START
```

```
.TITLE PMS
:IDENT \V04-000\

.EXTRN PMSSGL_FCP2, CTL$GL_PHD

.PSECT $CODE$,NOWRT,2

      50 00000000G 9F D0 00002   0000 00000
08F4  CA 08E8 CA 7D 00009
08FC  CA 08F0 CA D0 00010
0900  CA 38 A0 D0 00017
0904  CA 4C A0 D0 0001D   04 00023

.ENTRY PMS START, Save nothing : 1093
      MOVL @#CTL$GL_PHD, PROCESS HEADER : 1141
      MOVQ 2280(BASE), 2292(BASE) : 1143
      MOVL 2288(BASE), 2300(BASE) : 1145
      MOVL 56(PROCESS HEADER), 2304(BASE) : 1146
      MOVL 76(PROCESS_HEADER), 2308(BASE) : 1147
      RET : 1149
```

; Routine Size: 36 bytes,    Routine Base: \$CODE\$ + 0000

```
: 163    1150 1 GLOBAL ROUTINE PMS-END : L_NORM NOVALUE =
: 164
: 165
: 166
: 167    1152 1 ++
: 168    1153 1
: 169    1154 1 FUNCTIONAL DESCRIPTION:
: 170    1155 1
: 171    1156 1 This routine ends measurement for the current main function. It
: 172    1157 1 subtracts the stored base values from the running totals of the
: 173    1158 1 parameters and accumulates the delta in the system space cells
: 174    1159 1 for the particular function.
: 175    1160 1
: 176    1161 1
: 177    1162 1 CALLING SEQUENCE:
: 178    1163 1     PMS-END ()
: 179    1164 1
: 180    1165 1 INPUT PARAMETERS:
: 181    1166 1     NONE
: 182    1167 1
: 183    1168 1 IMPLICIT INPUTS:
: 184    1169 1     IO_PACKET: address of I/O packet of this function
: 185    1170 1
: 186    1171 1 OUTPUT PARAMETERS:
: 187    1172 1     NONE
: 188    1173 1
: 189    1174 1 IMPLICIT OUTPUTS:
: 190    1175 1     NONE
: 191    1176 1
: 192    1177 1 ROUTINE VALUE:
: 193    1178 1     NONE
: 194    1179 1
: 195    1180 1 SIDE EFFECTS:
: 196    1181 1     measurement data base updated
: 197    1182 1
: 198    1183 1 --.
: 199    1184 1
: 200    2 BEGIN
: 201    2 BIND      FUNCTAB      = UPLIT BYTE    ! table to translate function to array index
: 202    2           : VECTOR [,BYTE];
: 203    2           : BBLOCK[4],   ! value of parameter change
: 204    2           : array index
: 205    2           : REF BBLOCK;  ! I/O function code, including modifiers
: 206    2           : REF BBLOCK;  ! address of own process header
: 207    2           : CTL$GL_PHD   ! address of process header in control region
: 208
: 209    2 LOCAL      D,
: 210    2           FUNCTION : BBLOCK[4],   ! value of parameter change
: 211    2           J,
: 212    2           PROCESS_HEADER : REF BBLOCK;  ! I/O function code, including modifiers
: 213    2           : array index
: 214    2           : REF BBLOCK;  ! address of own process header
: 215    2 EXTERNAL    CTL$GL_PHD   : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
: 216    2           : address of process header in control region
: 217
: 218    2           : BIND_COMMON;
```

```
: 220      1207 2
: 221      1208 2  | If there is a subfunction open, close it first.
: 222      1209 2
: 223      1210 2
: 224      1211 2 IF .PMS_SUB_NEST NEQ 0
: 225      1212 2 THEN
: 226          BEGIN
: 227          PMS_SUB_NEST = 1;
: 228          PMS_END_SUB ();
: 229          END;
: 230
: 231      1217 2
: 232      1218 2 ! Derive the table index from the function code by searching the function
: 233      1219 2 code table. If the code is not found, we do not record data.
: 234      1220 2
: 235      1221 2 J =
: 236          BEGIN
: 237          INCR I FROM 0 TO 5 DO
: 238          IF .IO_PACKET[IRPSV_FCODE] EQL .FUNCTAB[.I]
: 239          THEN EXITLOOP .I
: 240          END;
: 241
: 242      1228 2
: 243      1229 2 IF .J EQL -1 THEN RETURN;
: 244
: 245      1231 2 ! Compute the deltas and accumulate them into the system space array.
: 246      1232 2 Note that we explicitly compute the change for each parameter and then
: 247      1233 2 add it into the data base cell, to prevent windows in which the value of
: 248      1234 2 the parameter is held in a local. This is because we cannot count on the
: 249      1235 2 compiler to generate simple stores which would be hazard free.
: 250
: 251      1238 2 PROCESS HEADER = .CTL$GL_PHD;
: 252      1239 2 FUNCTION = .IO_PACKET[IRPSW_FUNC];
: 253
: 254      1241 2 PMSSGL COUNT[.J] = .PMSSGL_COUNT[.J] + 1;
: 255      1242 2 IF .FUNCTION[IOSV_ACCESS]
: 256      1243 2 THEN PMSSGL MCNT[.J] = .PMSSGL_MCNT[.J] + 1;
: 257      1244 2 IF .FUNCTION[IOSV_CREATE]
: 258      1245 2 THEN PMSSGL MCNT[.J] = .PMSSGL_MCNT[.J] + 1;
: 259      1246 2 IF .FUNCTION[IOSV_DELETE]
: 260      1247 2 THEN PMSSGL MCNT[.J] = .PMSSGL_MCNT[.J] + 1;
: 261      1248 2 D = .PMS_TOT_READ - .PMS_FNC_READ;
: 262      1249 2 PMSSGL READ[.J] = .PMSSGL_READ[.J] + .D;
: 263      1250 2 D = .PMS_TOT_WRITE - .PMS_FNC_WRITE;
: 264      1251 2 PMSSGL WRITE[.J] = .PMSSGL_WRITE[.J] + .D;
: 265      1252 2 D = .PMS_TOT_CACHE - .PMS_FNC_CACHE;
: 266      1253 2 PMSSGL CACHE[.J] = .PMSSGL_CACHE[.J] + .D;
: 267      1254 2 D = .PROCESS HEADER[PHDSL_CPUTIM] - .PMS_FNC_CPU;
: 268      1255 2 PMSSGL CPU[.J] = .PMSSGL_CPU[.J] + .D;
: 269      1256 2 D = .PROCESS HEADER[PHDSL_PAGEFLTS] - .PMS_FNC_PFA;
: 270      1257 2 PMSSGL_PFA[.J] = .PMSSGL_PFA[.J] + .D;
: 271      1258 2
: 272      1259 1 END;
:                                ! end of routine PMS-END
```

				FUNCTAB=	P.AAA		
52	20	A1	0908 0000V	53 00000000G 000C 00000 0908 CA D5 00009 0A 13 0000D 01 D0 0000F 00 FB 00014 50 D4 00019 1\$: 51 52 90 AA DO 0001B 2\$: 52 52 D7 AF40 9A 0001F 00 ED 00024 07 13 0002A 50 F3 0002C 01 CE 00030 50 D1 00033 3\$: 50 69 13 0003A 52 00000000G 9F DO 0003C 51 90 AA DO 00043 51 20 A1 3C 00047 D8 A340 D6 0004B 51 06 E1 0004F 6340 D6 00053 51 95 00056 4\$: 51 03 18 00058 6340 D6 0005A 51 08 E1 0005D 5\$: 51 6340 D6 00061 51 08E8 CA 08F4 CA C3 00064 6\$: 51 28 A340 51 C0 0006C 51 08EC CA 08F8 CA C3 00071 50 A340 51 C0 00079 51 08F0 CA 08FC CA C3 0007E 78 A340 51 C0 00086 51 38 A2 0900 CA C3 0008B 00A0 C340 51 C0 00092 51 4C A2 0904 CA C3 00098 00C8 C340 51 C0 0009F 04 000A5 7\$: 04	.ENTRY MOVAB TSTL BEQL MOVL CALLS CLRL MOVL MOVZBL CMPZV BEQL AOBLEQ MNEGL CMPL BEQL MOVL MOVL MOVZWL INCL BBC INCL TSTB BGEQ INCL BBC INCL SUBL3 ADDL2 SUBL3 ADDL2 SUBL3 ADDL2 SUBL3 ADDL2 SUBL3 ADDL2 SUBL3 ADDL2 RET	PMS END, Save R2,R3 @#PMSSGL MCNT, R3 2312(BASE) 1\$ #1, 2312(BASE) #0, PMS_END_SUB I -112(BASE), R1 FUNCTAB[I], R2 #0, #6, 32(R1), R2 3\$ #5, I, 2\$ #1, J J # -1 7\$ @#CTLSGL PHD, PROCESS_HEADER -112(BASE), R1 32(R1), FUNCTION PMSSGL COUNT[J] #6, FUNCTION, 4\$ PMSSGL MCNT[J] FUNCTION 5\$ PMSSGL MCNT[J] #8, FUNCTION, 6\$ PMSSGL MCNT[J] 2292(BASE), 2280(BASE), D D, PMSSGL READ[J] 2296(BASE), 2284(BASE), D D, PMSSGL WRITE[J] 2300(BASE), 2288(BASE), D D, PMSSGL CACHE[J] 2304(BASE), 56(PROCESS_HEADER), D D, PMSSGL CPU[J] 2308(BASE), 76(PROCESS_HEADER), D D, PMSSGL_PFA[J]	1150 1211 1214 1215 1224 1225 1224 1229 1238 1239 1241 1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1259

; Routine Size: 166 bytes.    Routine Base: \$CODE\$ + 002A

```
: 274      1260 1 GLOBAL ROUTINE PMS_START_SUB (INDEX) : L_NORM NOVALUE =
: 275      1261 1
: 276      1262 1 ++
: 277      1263 1
: 278      1264 1 FUNCTIONAL DESCRIPTION:
: 279      1265 1
: 280      1266 1 This routine starts metering for the indicated subfunction.
: 281      1267 1
: 282      1268 1
: 283      1269 1 CALLING SEQUENCE:
: 284      1270 1     PMS_START_SUB (ARG1)
: 285      1271 1
: 286      1272 1 INPUT PARAMETERS:
: 287      1273 1     ARG1: index of measurement array to use
: 288      1274 1
: 289      1275 1 IMPLICIT INPUTS:
: 290      1276 1     NONE
: 291      1277 1
: 292      1278 1 OUTPUT PARAMETERS:
: 293      1279 1     NONE
: 294      1280 1
: 295      1281 1 IMPLICIT OUTPUTS:
: 296      1282 1     NONE
: 297      1283 1
: 298      1284 1 ROUTINE VALUE:
: 299      1285 1     NONE
: 300      1286 1
: 301      1287 1 SIDE EFFECTS:
: 302      1288 1     NONE
: 303      1289 1
: 304      1290 1 --
: 305      1291 1
: 306      1292 2 BEGIN
: 307      1293 2
: 308      1294 2 LOCAL
: 309      1295 2     PROCESS_HEADER : REF BBLOCK; ! pointer to FCP process header
: 310      1296 2
: 311      1297 2 BIND_COMMON;
: 312      1298 2
: 313      1299 2 EXTERNAL
: 314      1300 2     CTL$GL_PHD : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
: 315      1301 2           ! address of process header in control region
: 316      1302 2
: 317      1303 2 | We copy the current running totals into subfunction holding cells to
: 318      1304 2 | compute the deltas later. Note that since the extend subfunction can be
: 319      1305 2 | reentered, we do nothing if the depth count is already non-zero.
: 320      1306 2 |
: 321      1307 2
: 322      1308 2 PMS_SUB_NEST = .PMS_SUB_NEST + 1;
: 323      1309 2 IF .PMS_SUB_NEST NEG 1 THEN RETURN;
: 324      1310 2 PROCESS_HEADER = .CTL$GL_PHD;
: 325      1311 2
: 326      1312 2 PMS_SUB_FUNC = .INDEX;
: 327      1313 2 PMS_SUB_READ = .PMS_TOT_READ;
: 328      1314 2 PMS_SUB_WRITE = .PMS_TOT_WRITE;
: 329      1315 2 PMS_SUB_CACHE = .PMS_TOT_CACHE;
: 330      1316 2 PMS_SUB_CPU = .PROCESS_HEADER[PHDSL_CPUTIM];
```

: 331 1317 2 PMS\_SUB\_PFA = .PROCESS\_HEADER[PHD\$L\_PAGEFLTS];  
: 332 1318 2  
: 333 1319 1 END;  
                                ! end of routine PMS\_START\_SUB

				.ENTRY	PMS START SUB Save nothing	
			50 0908 CA 9E 0000 0000	MOVAB	2312(BASE), R0	: 1260
			60 D6 00007	INCL	(R0)	: 1295
		01	60 D1 00009	CMPL	(R0), #1	: 1308
			27 12 0000C	BNEQ	1S	: 1309
			50 0000000G 9F D0 0000E	MOVL	@#CTL\$GL PHD, PROCESS_HEADER	: 1310
	090C	CA 04	AC D0 00015	MOVL	INDEX, 2316(BASE)	: 1312
	0910	CA 08E8	CA 7D 0001B	MOVQ	2280(BASE), 2320(BASE)	: 1313
	0918	CA 08F0	CA D0 00022	MOVL	2288(BASE), 2328(BASE)	: 1315
	091C	CA 38	A0 D0 00029	MOVL	56(PROCESS_HEADER), 2332(BASE)	: 1316
	0920	CA 4C	A0 D0 0002F	MOVL	76(PROCESS_HEADER), 2336(BASE)	: 1317
			04 00035 1\$:	RET		: 1319

: Routine Size: 54 bytes, Routine Base: \$CODE\$ + 0000

```

335 1320 1 GLOBAL ROUTINE PMS_END_SUB : L_NORM NOVALUE =
336 1321 1
337 1322 1 !++
338 1323 1
339 1324 1 FUNCTIONAL DESCRIPTION:
340 1325 1
341 1326 1 This routine ends metering for the currently active subfunction.
342 1327 1
343 1328 1
344 1329 1 CALLING SEQUENCE:
345 1330 1 PMS_END_SUB ()
346 1331 1
347 1332 1 INPUT PARAMETERS:
348 1333 1 NONE
349 1334 1
350 1335 1 IMPLICIT INPUTS:
351 1336 1 NONE
352 1337 1
353 1338 1 OUTPUT PARAMETERS:
354 1339 1 NONE
355 1340 1
356 1341 1 IMPLICIT OUTPUTS:
357 1342 1 NONE
358 1343 1
359 1344 1 ROUTINE VALUE:
360 1345 1 NONE
361 1346 1
362 1347 1 SIDE EFFECTS:
363 1348 1 measurement data base updated
364 1349 1
365 1350 1 ---
```

2 BEGIN

2 LOCAL

2 J,   | array index

2 D,   | parameter difference

2 PROCESS\_HEADER : REF BBLOCK;           | pointer to FCP process header

2 BIND\_COMMON;

2 EXTERNAL

2 CTL\_SGL\_PHD : REF BBLOCK ADDRESSING\_MODE (ABSOLUTE);

2   | address of process header in control region

2 ! Decrement the nesting count. If non-zero, we are in a nested extend and

2 do nothing.

2 !

2 PMS\_SUB\_NEST = .PMS\_SUB\_NEST - 1;

2 IF .PMS\_SUB\_NEST NEQ 0 THEN RETURN;

2 ! Now compute the delta for each parameter by subtracting the base from the

2 running total. Record it by adding into the system array. Also deduct

2 the delta from the charge for the main function by adding it into the

2 main function base.

```

: 392      1377 2
: 393      1378 2 PROCESS HEADER = .CTL$GL_PHD;
: 394      1379 2 J = .PMS_SUB_FUNC;           ! get array index
: 395      1380 2
: 396      1381 2 PMSSGL_COUNT[.J] = .PMSSGL_COUNT[.J] + 1;
: 397      1382 2
: 398      1383 2 D = .PMS_TOT_READ - .PMS_SUB_READ;
: 399      1384 2 PMSSGL_READ[.J] = .PMSSGL_READ[.J] + .D;
: 400      1385 2 PMS_FNC_READ = .PMS_FNC_READ + .D;
: 401      1386 2
: 402      1387 2 D = .PMS_TOT_WRITE - .PMS_SUB_WRITE;
: 403      1388 2 PMSSGL_WRITE[.J] = .PMSSGL_WRITE[.J] + .D;
: 404      1389 2 PMS_FNC_WRITE = .PMS_FNC_WRITE + .D;
: 405      1390 2
: 406      1391 2 D = .PMS_TOT_CACHE - .PMS_SUB_CACHE;
: 407      1392 2 PMSSGL_CACHE[.J] = .PMSSGL_CACHE[.J] + .D;
: 408      1393 2 PMS_FNC_CACHE = .PMS_FNC_CACHE + .D;
: 409      1394 2
: 410      1395 2 D = .PROCESS_HEADER[PHD$L_CPUTIM] - .PMS_SUB_CPU;
: 411      1396 2 PMSSGL_CPU[.J] = .PMSSGL_CPU[.J] + .D;
: 412      1397 2 PMS_FNC_CPU = .PMS_FNC_CPU + .D;
: 413      1398 2
: 414      1399 2 D = .PROCESS_HEADER[PHD$L_PAGEFLTS] - .PMS_SUB_PFA;
: 415      1400 2 PMSSGL_PFA[.J] = .PMSSGL_PFA[.J] + .D;
: 416      1401 2 PMS_FNC_PFA = .PMS_FNC_PFA + .D;
: 417      1402 2
: 418      1403 1 END;                      ! end of routine PMS-END_SUB

```

					ENTRY	PMS END SUB, Save R2,R3	1320
		53 00000000G	9F 9E 00002		MOVAB	@#PMSSGL_COUNT, R3	1357
		50 0908	CA 9E 00009		MOVAB	2312(BASE), R0	1369
			60 D7 0000E		DECL	(R0)	1370
			60 D5 00010		TSTL	(R0)	
			6A 12 00012		BNEQ	1\$	
		52 00000000G	9F D0 00014		MOVL	@#CTL\$GL_PHD, PROCESS_HEADER	1378
		50 090C	CA D0 0001B		MOVL	2316(BASE), J	1379
			6340 D6 00020		INCL	PMSSGL_COUNT[J]	1381
51	08E8 CA	0910	CA C3 00023		SUBL3	2320(BASE), 2280(BASE), D	1383
	50 A340		51 C0 0002B		ADDL2	D, PMSSGL_READ[J]	1384
	08F4 CA		51 C0 00030		ADDL2	D, 2292(BASE)	1385
51	08EC CA	0914	CA C3 00035		SUBL3	2324(BASE), 2284(BASE), D	1387
	78 A340		51 C0 0003D		ADDL2	D, PMSSGL_WRITE[J]	1388
	08F8 CA		51 C0 00042		ADDL2	D, 2296(BASE)	1389
51	08F0 CA	0918	CA C3 00047		SUBL3	2328(BASE), 2288(BASE), D	1391
	00A0 C340		51 C0 0004F		ADDL2	D, PMSSGL_CACHE[J]	1392
	08FC CA		51 C0 00055		ADDL2	D, 2300(BASE)	1393
51	38 A2	091C	CA C3 0005A		SUBL3	2332(BASE), 56(PROCESS_HEADER), D	1395
	00C8 C340		51 C0 00061		ADDL2	D, PMSSGL_CPU[J]	1396
	0900 CA		51 C0 00067		ADDL2	D, 2304(BASE)	1397
51	4C A2	0920	CA C3 0006C		SUBL3	2336(BASE), 76(PROCESS_HEADER), D	1399
	00F0 C340		51 C0 00073		ADDL2	D, PMSSGL_PFA[J]	1400
	0904 CA		51 C0 00079		ADDL2	D, 2308(BASE)	1401
			04 0007E 1\$:		RET		1403

; Routine Size: 127 bytes, Routine Base: \$CODE\$ + 0106

; 419 1404 1  
; 420 1405 1 END  
; 421 1406 0 ELUDOM

#### PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	389	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

#### Library Statistics

File	Total	Symbols	Pages Mapped	Processing Time
\$_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	30	0	00:02.0

#### COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:PMS/OBJ=OBJ\$:PMS MSRC\$:PMS/UPDATE=(ENH\$:PMS)

Size: 383 code + 6 data bytes  
Run Time: 00:41.6  
Elapsed Time: 01:19.0  
Lines/CPU Min: 2029  
Lexemes/CPU-Min: 66236  
Memory Used: 227 pages  
Compilation Complete

0171 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

LOCKDB  
LIS

LOCKERS  
LIS

MAKACC  
LIS

MAKPTR  
LIS

MATCHNAME  
LIS

MPWIND  
LIS

PARSNM  
LIS

QUOTAUTIL  
LIS

100ONE  
LIS

LOCKON  
LIS

MAPUBN  
LIS

MODIFY  
LIS

MOUNT  
LIS

NXTHDR  
LIS

MAKNMB  
LIS

MAKSTR  
LIS