


```

MM      MM      000000      MM      MM      SSSSSSSS      UU      UU      BBBB8888      SSSSSSSS
MM      MM      000000      MM      MM      SSSSSSSS      UU      UU      BBBB8888      SSSSSSSS
MMMM    MMMM    00          00    MMMM    MMMM    SS          UU      UU      BB      BB      SS
MMMM    MMMM    00          00    MMMM    MMMM    SS          UU      UU      BB      BB      SS
MM      MM      00          00    MM      MM      SS          UU      UU      BB      BB      SS
MM      MM      00          00    MM      MM      SS          UU      UU      BB      BB      SS
MM      MM      00          00    MM      MM      SSSSSS      UU      UU      BBBB8888      SSSSSS
MM      MM      00          00    MM      MM      SSSSSS      UU      UU      BBBB8888      SSSSSS
MM      MM      00          00    MM      MM      SS          UU      UU      BB      BB      SS
MM      MM      00          00    MM      MM      SS          UU      UU      BB      BB      SS
MM      MM      00          00    MM      MM      SS          UU      UU      BB      BB      SS
MM      MM      00          00    MM      MM      SS          UU      UU      BB      BB      SS
MM      MM      000000      MM      MM      SSSSSSSS      UUUUUUUUU      BBBB8888      SSSSSSSS
MM      MM      000000      MM      MM      SSSSSSSS      UUUUUUUUU      BBBB8888      SSSSSSSS

```

```

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

```

.....

```

1 0001 0 %TITLE 'Special service routines'
2 0002 C MODULE MOMSUBS (
3 0003 0 LANGUAGE (BLISS32),
4 0004 0 ADDRESSING_MODE (NONEXTERNAL=GENERAL),
5 0005 0 ADDRESSING_MODE (EXTERNAL=GENERAL),
6 0006 0 IDENT = 'V04-000'
7 0007 0 ) =
8 0008 1 BEGIN
9 0009 1
10 0010 1 *****
11 0011 1 *
12 0012 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
13 0013 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
14 0014 1 * ALL RIGHTS RESERVED. *
15 0015 1 *
16 0016 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
17 0017 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
18 0018 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
19 0019 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
20 0020 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
21 0021 1 * TRANSFERRED. *
22 0022 1 *
23 0023 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
24 0024 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
25 0025 1 * CORPORATION. *
26 0026 1 *
27 0027 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
28 0028 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
29 0029 1 *
30 0030 1 *
31 0031 1 *****
32 0032 1
33 0033 1
34 0034 1 **
35 0035 1 FACILITY: DECnet-VAX Network Maintenance Operations Module (MOM)
36 0036 1
37 0037 1 ABSTRACT:
38 0038 1 This module contains utility routines used for maintenance operations.
39 0039 1
40 0040 1 ENVIRONMENT: VAX/VMS Operating System
41 0041 1
42 0042 1 AUTHOR: Kathy Perko
43 0043 1
44 0044 1 CREATION DATE: 6-Jan-1983
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1 V03-005 MKP0005 Kathy Perko 26-June-1984
48 0048 1 If sending a BOOT message for a LOAD command, set the bit
49 0049 1 that tells the target to perform the load from this host.
50 0050 1
51 0051 1 V03-004 MKP0004 Kathy Perko 12-April-1984
52 0052 1 Change padding on SERVICE PASSWORD to zero instead of high
53 0053 1 byte.
54 0054 1
55 0055 1 V03-003 MKP0003 Kathy Perko 20-Jan-1984
56 0056 1 Add SERVICE NODE VERSION parameter.
57 0057 1 Pad the service password in the boot message with the

```

```
.. 58      0058 1 | high byte.  
.. 59      0059 1 |  
.. 60      0060 1 | V03-002 MKP0002      Kathy Perko      23-May-1983  
.. 61      0061 1 | When building the MOP Parameter Load with Transfer Address  
.. 62      0062 1 | message, mask out the area number if the target isn't on the  
.. 63      0063 1 | NI (this is a temporary way of identifying Phase III targets).  
.. 64      0064 1 |  
.. 65      0065 1 | V03-001 MKP0001      Kathy Perko      11-May-1983  
.. 66      0066 1 | Fix length of password put into MOP boot message.  
.. 67      0067 1 |  
.. 68      0068 1 | --  
.. 69      0069 1 |
```

```

71 0070 1 %SBTTL 'Declarations'
72 0071 1
73 0072 1
74 0073 1 ! TABLE OF CONTENTS:
75 0074 1 !
76 0075 1 !
77 0076 1 FORWARD ROUTINE
78 0077 1     mom$getsrvdata      : NOVALUE,
79 0078 1     mom$get_circuit_type : NOVALUE,
80 0079 1     mom$get_node_id     : NOVALUE,
81 0080 1     mom$getsrvtimer    : NOVALUE,
82 0081 1     mom$get_voldb_data,
83 0082 1     mom_get_circ_search2_key: NOVALUE,
84 0083 1     mom$bldmoprd$      : NOVALUE,
85 0084 1     mom$bldmopboot    : NOVALUE,
86 0085 1     mom$bldmopplt     : NOVALUE;
87 0086 1
88 0087 1 !
89 0088 1 ! INCLUDE FILES:
90 0089 1 !
91 0090 1 !
92 0091 1 LIBRARY 'LIB$:MOMLIB.L32';
93 0092 1 LIBRARY 'SHRLIB$:NMALIBRY.L32';
94 0093 1 LIBRARY 'SHRLIB$:NET.L32';
95 0094 1 LIBRARY 'SYSS$LIBRARY:STARLET.L32';
96 0095 1
97 0096 1 !
98 0097 1 ! OWN STORAGE:
99 0098 1 !
100 0099 1 !
101 0100 1 OWN
102 0101 1     mom$t_p2buffer : VECTOR [mom$k_p2_buf_len, BYTE]; ! P2 QIO buffer
103 0102 1
104 0103 1 BIND
105 0104 1     mom$q_p2_buf_dsc = UPLIT (mom$k_p2_buf_len, mom$t_p2buffer) : VECTOR [2];
106 0105 1
107 0106 1 !
108 0107 1 ! EXTERNAL REFERENCES:
109 0108 1 !
110 0109 1 !
111 0110 1 $mom_externals; ! Define external service data
112 0111 1
113 0112 1 EXTERNAL
114 0113 1     mom$npa_load,
115 0114 1     mom$npa_cirloop,
116 0115 1     mom$npa_trigger;
117 0116 1
118 0117 1 EXTERNAL ROUTINE
119 0118 1     mom$bld_reply,
120 0119 1     mom$buid_p2,
121 0120 1     mom$error,
122 0121 1     mom$debug_msg,
123 0122 1     mom$debug_txt,
124 0123 1     mom$netacp_qio;

```

```

126 0124 1 %SBTTL 'mom$getsrvdata Build the service data base'
127 0125 1 GLOBAL ROUTINE mom$getsrvdata : NOVALUE =
128 0126 1
129 0127 1 |++
130 0128 1 | FUNCTIONAL DESCRIPTION:
131 0129 1 |
132 0130 1 |     This routine gets the information needed for a maintenance operation
133 0131 1 |     from the target node's volatile data base entry.
134 0132 1 |
135 0133 1 | ROUTINE VALUE:
136 0134 1 | COMPLETION CODES:
137 0135 1 |
138 0136 1 |     Signal errors.
139 0137 1 |
140 0138 1 | --
141 0139 1 |
142 0140 2 BEGIN
143 0141 2
144 0142 2 LOCAL
145 0143 2     datptr,
146 0144 2     string_len,
147 0145 2     p4_buf_dsc : VECTOR [2],
148 0146 2     qio_p4_buffer : BBLOCK [mom$k_qio_buf_len];
149 0147 2
150 0148 2 IF .mom$gb_function NEQ nma$c_fnc_tes THEN
151 0149 2 BEGIN
152 0150 2 |
153 0151 2 |     Get the maintenance parameters from NETACPs node database entry for the
154 0152 2 |     target node.
155 0153 2 |
156 0154 2 |     p4_buf_dsc [0] = mom$k_qio_buf_len;
157 0155 2 |     p4_buf_dsc [1] = qio_p4_buffer;
158 0156 2 |
159 0157 2 |     mom$get_voldb_data (nfb$c_db_ndi, p4_buf_dsc);
160 0158 2 |
161 0159 2 |     Build the service data table. This table contains the values of longword
162 0160 2 |     parameters, and pointers to string parameters.
163 0161 2 |
164 0162 2 |     datptr = qio_p4_buffer;
165 0163 2 |
166 0164 2 |     Some parameters have already been extracted from the NICE or MOP message
167 0165 2 |     and inserted in the Service Data table. These take precedence over
168 0166 2 |     what's in the volatile database. So, move the rest of the service
169 0167 2 |     parameters from the QIOs P4 buffer into Service Data Table.
170 0168 2 |     The field IDs were put into the NFB in the order they are in in the
171 0169 2 |     Service Data Table. Extract the parameter values from the P4 buffer
172 0170 2 |     in the same order.
173 0171 2 |
174 0172 3 INCR i FROM 0 TO svd$c_entry_count DO
175 0173 4 BEGIN
176 0174 4 |
177 0175 4 |     If the parameter value is obtained from the remote node (NDI)
178 0176 4 |     database and it hasn't already been set by the NICE or MOP message,
179 0177 4 |     put it into the Service Data table.
180 0178 4 |
181 0179 4 |     IF .mom$ab_service_data [.i, svd$b_nfb_database]
182 0180 4 |         EQ[ nfb$c_db_ndi THEN

```

```

183 0181 5 BEGIN
184 0182 5 IF .mom$ab_service_data [.i, svd$b_nice_type] NEQ
185 0183 5 svd$k_string THEN
186 0184 5
187 0185 5 | If the parameter isn't a string and a value was returned
188 0186 5 | for it, move its value into the Service Data Table.
189 0187 5
190 0188 6 BEGIN
191 0189 6 IF ..datptr GTR -1 AND
192 0190 6 NOT .mom$ab_service_data [.i, svd$v_msg_param] T. EN
193 0191 6 mom$ab_service_data [.i, svd$l_param] = ..datptr;
194 0192 6 datptr = .datptr + 4;
195 0193 6 END
196 0194 5 ELSE
197 0195 5 | If the parameter is a string, and a value was returned for
198 0196 5 | it, move the string into Service Data Table.
199 0197 5
200 0198 5 BEGIN
201 0199 6 string_len = .(.datptr)<0,16>;
202 0200 6 IF .string_len GTR 0 AND
203 0201 6 NOT .mom$ab_service_data [.i, svd$v_msg_param] THEN
204 0202 6 BEGIN
205 0203 7 mom$ab_service_data [.i, svd$b_string_len] = .string_len;
206 0204 7 CHSMOVE (.string_len,
207 0205 7 (.datptr + 2),
208 0206 7 mom$ab_service_data [.i, svd$t_string]);
209 0207 7 END;
210 0208 6 datptr = .string_len + .datptr + 2;
211 0209 6 END;
212 0210 5 END;
213 0211 4 END;
214 0212 3
215 0213 3 | Get the Host node id for Loads and dumps.
216 0214 3
217 0215 3 IF .mom$gb_function EQL nma$c_fnc_loa OR
218 0216 3 .mom$gb_function EQL nma$c_fnc_dum THEN
219 0217 3 mom$get_node_id (svd$gk_pcno_1ho,
220 0218 3 svd$gk_pcno_1hna);
221 0219 3
222 0220 2 END;
223 0221 2
224 0222 2 | Determine if service circuit is an NI circuit. NI service operations
225 0223 2 | are different from point-to-point or multipoint at many points. For
226 0224 2 | autoservice this is determined elsewhere.
227 0225 2
228 0226 2
229 0227 2 IF NOT .mom$gl_service_flags [mom$v_autoservice] THEN
230 0228 2 mom$get_circuit_type ();
231 0229 1 END; ! End of mom$getsrvdata

```

.TITLE MOMSUBS Special service routines
.IDENT \V04-000\
.PSECT \$PLITS,NOWRT,NOEXE,2

00000068 00000 P.AAA: .LONG 104

00000000' 00004

.ADDRESS MOMST_P2BUFFER

.PSECT \$OWNS,NOEXE,2

00000 MOMST_P2BUFFER:

.BLKB 104

MOMSQ_P2_BUF_DSC= P.AAA

- .EXTRN MOM\$GL_LOGMASK, MOM\$GL_SVD_INDEX
- .EXTRN MOM\$AB_SERVICE_DATA
- .EXTRN MOM\$GB_FUNCTION
- .EXTRN MOM\$GB_OPTION_BYTE
- .EXTRN MOM\$GB_ENTITY_CODE
- .EXTRN MOM\$AB_ENTITY_BUF
- .EXTRN MOM\$GQ_ENTITY_BUF_DSC
- .EXTRN MOM\$GL_SERVICE_FLAGS
- .EXTRN MOM\$AB_NPARSE_BLK
- .EXTRN MOM\$AB_NICE_RCV_BUF
- .EXTRN MOM\$AB_NICE_XMIT_BUF
- .EXTRN MOM\$GQ_NICE_RCV_BUF_DSC
- .EXTRN MOM\$GL_NICE_RCV_MSG_LEN
- .EXTRN MOM\$GQ_NICE_XMIT_BUF_DSC
- .EXTRN MOM\$AB_MSGBLOCK
- .EXTRN MOM\$AB_ACPQIO_BUFFER
- .EXTRN MOM\$GQ_ACPQIO_BUF_DSC
- .EXTRN MOM\$AB_CIB, MOM\$AB_LOOP_CIB
- .EXTRN MOM\$AB_TRIGGER_CIB
- .EXTRN MOM\$AB_MOP_XMIT_BUF
- .EXTRN MOM\$GQ_MOP_XMIT_BUF_DSC
- .EXTRN MOM\$AB_MOP_RCV_BUF
- .EXTRN MOM\$GQ_MOP_RCV_BUF_DSC
- .EXTRN MOM\$AB_MOP_MSG, MOM\$GQ_MOP_MSG_DSC
- .EXTRN MOM\$GW_EVT_CODE
- .EXTRN MOM\$GB_EVT_POPR
- .EXTRN MOM\$GB_EVT_PRSN
- .EXTRN MOM\$GB_EVT_PSER
- .EXTRN SVDSGK_PCNO_ADD
- .EXTRN SVDSGK_PCNO_SDV
- .EXTRN SVDSGK_PCNO_CPU
- .EXTRN SVDSGK_PCNO_STY
- .EXTRN SVDSGK_PCNO_DAD
- .EXTRN SVDSGK_PCNO_DCT
- .EXTRN SVDSGK_PCNO_IHO
- .EXTRN SVDSGK_PCNO_NNA
- .EXTRN SVDSGK_PCNO_SLI
- .EXTRN SVDSGK_PCNO_SPA
- .EXTRN SVDSGK_PCNO_HWA
- .EXTRN SVDSGK_PCNO_SNV
- .EXTRN SVDSGK_PCNO_LOA
- .EXTRN SVDSGK_PCNO_SLO
- .EXTRN SVDSGK_PCNO_TLO
- .EXTRN SVDSGK_PCNO_DFL
- .EXTRN SVDSGK_PCNO_SID
- .EXTRN SVDSGK_PCNO_DUM
- .EXTRN SVDSGK_PCNO_SDU
- .EXTRN SVDSGK_PCNO_\$HNA
- .EXTRN SVDSGK_PCNO_\$HHW


```

                                07FC 00000
                                SA 00000000G 00 9E 00002
                                59 00000000G 00 9E 00009
                                SE      FDF8  CE 9E 00010
                                12              6A 91 00015
                                           03 12 00018
                                           008C 31 0001A
                                FB  AD      0200 8F 3C 0001D 1$:
                                FC  AD              6E 9E 00023
                                           F8  AD 9F 00027
                                           02 DD 0002A
                                00000000V 00      02 FB 0002C
                                57      6E 9E 00033
                                56      01 CE 00036
                                           46 11 00039
                                50 56 00000089 8F C5 0003B 2$:
                                02      FC A940 91 00043
                                           37 12 00048
                                03      FF A940 91 0004A
                                           15 13 0004F
                                           67 D5 00051
                                           0C 19 00053
                                07      6940      00 E0 00055
                                           02 A940 9F 0005A
                                           67 D0 0005E
                                9E      04 C0 00061 3$:
                                57      1B 11 00064
                                           58      67 3C 00066 4$:
                                           11 15 00069
                                0C      6940      00 E0 0006B
                                01 A940      58 90 00070
                                02 A7        58 28 00075

.EXTRN  SVD$GK_PCNO_SFTY
.EXTRN  SVD$GK_PCNO_PHA
.EXTRN  SVD$GK_PCNO_$DA
.EXTRN  SVD$GK_PCNO_LPC
.EXTRN  SVD$GK_PCNO_LPL
.EXTRN  SVD$GK_PCNO_LPD
.EXTRN  SVD$GK_PCNO_LPH
.EXTRN  SVD$GK_PCNO_LPA
.EXTRN  SVD$GK_PCNO_LPN
.EXTRN  SVD$GK_PCNO_$LNA
.EXTRN  SVD$GK_PCNO_$LNH
.EXTRN  SVD$GK_PCNO_LAN
.EXTRN  SVD$GK_PCNO_$LNN
.EXTRN  SVD$GK_PCNO_$LAH
.EXTRN  SVD$GK_PCLI_STI
.EXTRN  SVD$C_ENTRY_COUNT
.EXTRN  MOM$NPA_LOAD, MOM$NPA_CIRLOOP
.EXTRN  MOM$NPA_TRIGGER
.EXTRN  MOM$BLD_REPLY, MOM$BUILD_P2
.EXTRN  MOM$ERROR, MOM$DEBUG_MSG
.EXTRN  MOM$DEBUG_TXT, MOM$NETACP_QIO

.PSECT  $CODE$,NOWRT,2

.ENTRY  MOM$GETSRVDATA, Save R2,R3,R4,R5,R6,R7,R8,- 0125
        R9,R10
        MOVAB  MOM$GB_FUNCTION, R10
        MOVAB  MOM$AB_SERVICE_DATA+7, R9
        MOVAB  -520(SP), SP
        CMPB   MOM$GB_FUNCTION, #18
        BNEQ   1$
        BRW    8$
        MOVZWL #512, P4_BUF_DSC
        MOVAB  QIO_P4_BUFFER, P4_BUF_DSC+4
        PUSHAB P4_BUF_DSC
        PUSHL  #2
        CALLS  #2, MOM$GET_VOLDB_DATA
        MOVAB  QIO_P4_BUFFER, DATPTR
        MNEGL  #1, I
        BRB   6$
        MULL3  #137, I, R0
        CMPB   MOM$AB_SERVICE_DATA+3[R0], #2
        BNEQ   6$
        CMPB   MOM$AB_SERVICE_DATA+6[R0], #3
        BEQL   4$
        TSTL  (DATPTR)
        BLSS  3$
        BBS   #0, MOM$AB_SERVICE_DATA+7[R0], 3$
        PUSHAB MOM$AB_SERVICE_DATA+9[R0]
        MOVL  (DATPTR), 2(SP)+
        ADDL2 #4, DATPTR
        BRB   6$
        MOVZWL (DATPTR), STRING_LEN
        BLEQ  5$
        BBS   #0, MOM$AB_SERVICE_DATA+7[R0], 5$
        MOVB  STRING_LEN, MOM$AB_SERVICE_DATA+8[R0]
        MOVC3 STRING_LEN, 2(DATPTR), -

```

B2	57	02 A748	9E	0007C	5\$:	MOVAB	MOM\$AB SERVICE DATA+9[R0]	:	0209
	56	00000000G	8F	F3 00081	6\$:	AOBLEQ	2(DATPTR)[STRING LEN], DATPTR	:	0172
	50		6A	9A 00089		MOVZBL	#SVD\$C_ENTRY_COUNT, I, 2\$:	0216
	0F		50	91 0008C		CMPB	MOM\$GB_FUNCTION, R0	:	
			05	13 0008F		BEQL	R0, #15	:	
	10		50	91 00091		CMPB	R0, #16	:	0217
			13	12 00094		BNEQ	8\$:	
		00000000G	8F	DD 00096	7\$:	PUSHL	#SVD\$GK_PCNO_SHNA	:	0218
		00000000G	8F	DD 0009C		PUSHL	#SVD\$GK_PCNO_IHO	:	
00000000V	00		02	FB 000A2		CALLS	#2, MOM\$GET_NODE_ID	:	
	07	00000000G	00	E8 000A9	8\$:	BLBS	MOM\$GL_SERVICE_FLAGS, 9\$:	0227
00000000V	00		00	FB 000B0		CALLS	#0, MOM\$GET_CIRCUIT_TYPE	:	0228
			04	000B7	9\$:	RET		:	0229

; Routine Size: 184 bytes, Routine Base: \$CODE\$ + 0000

```

233 0230 1 %SBTTL 'mom$get_circuit_type See if Circuit is on Ethernet'
234 0231 1 GLOBAL ROUTINE mom$get_circuit_type : NOVALUE =
235 0232 1
236 0233 1 :++
237 0234 1 : FUNCTIONAL DESCRIPTION:
238 0235 1 : This routine looks the service circuit up in the volatile database
239 0236 1 : to determine if it's an NI circuit or not.
240 0237 1
241 0238 1 : ROUTINE VALUE:
242 0239 1 : COMPLETION CODES:
243 0240 1 :
244 0241 1 : Signal errors.
245 0242 1 :
246 0243 1 :--
247 0244 1
248 0245 2 BEGIN
249 0246 2
250 P 0247 2 $nfbdsc (mom_q_cirtyp_nfbdsc, show, , cri
251 P 0248 2 ,nam, : Search key one = circuit name, oper, = eql
252 P 0249 2 ,:typ : Null search key two.
253 P 0250 2 ,:typ : Circuit type
254 0251 2 );
255 0252 2
256 0253 2 LOCAL
257 0254 2 len,
258 0255 2 msgsize,
259 0256 2 p2dsc: VECTOR [2],
260 0257 2 p3,
261 0258 2 err detail,
262 0259 2 status;
263 0260 2
264 0261 2 :
265 0262 2 : If there isn't any service circuit for the node, return an error to NCP.
266 0263 2 : (There is always a service circuit for autoservice functions).
267 0264 2 :
268 0265 2 len = .mom$ab_service_data [svd$gk_pcno_sli, svd$b_string_len];
269 0266 2 IF .len EQL 0 THEN
270 0267 2 BEGIN
271 0268 2 mom$ab_msgblock [msb$l_flags] = msb$m_det_fld;
272 0269 2 mom$ab_msgblock [msb$b_code] = nma$c_sts_pms;
273 0270 2 mom$ab_msgblock [msb$w_detail] = nma$c_pcno_sli;
274 0271 2 mom$blb_reply (mom$ab_msgblock, msgsize);
275 0272 2 $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
276 0273 2 END;
277 0274 2 :
278 0275 2 : Get the circuit type from NETACPs CRI database to determine if it's
279 0276 2 : an NI (Ethernet) circuit.
280 0277 2 :
281 0278 2 mom$build_p2 (.len,
282 0279 2 mom$ab_service_data [svd$gk_pcno_sli, svd$b_string],
283 0280 2 -1, 0,
284 0281 2 mom$q_p2_buf_dsc, p2dsc);
285 0282 2 status = mom$netacp_qio (mom_q_cirtyp_nfbdsc,
286 0283 2 p2dsc,
287 0284 2 p3,
288 0285 2 mom$qq_acpqio_buf_dsc);
289 0286 2 IF NOT .status THEN

```

```

290 0287 3 BEGIN
291 0288 mom$bld_reply (mom$ab_msgblock, msgsize);
292 0289 $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
293 0290 END;
294 0291 IF (.mom$gq_acpqio_buf_dsc [1]) EQL nma$c_cirty_ni THEN
295 0292 BEGIN
296 0293 mom$gl_service_flags [mom$V_ni_circ] = true;
297 0294 err_detail = 0;
298 0295
299 0296     If it's an NI circuit, and the NICE command was LOAD VIA, TRIGGER VIA,
300 0297     it must also specify a physical address. If it's LOOP CIRCUIT it must
301 0298     specify a physical address or a node id. This is because the circuit
302 0299     id is not sufficient to uniquely identify a target on the NI.
303 0300
304 0301 IF NOT .mom$gl_service_flags [mom$V_autoservice] AND
305 0302 NOT .mom$ab_service_data [svd$gk_pcno_pha, svd$V_msg_param] THEN
306 0303 BEGIN
307 0304     IF .mom$gb_entity_code EQL mom$c_circuit THEN
308 0305 BEGIN
309 0306     IF .mom$gb_function NEQ nma$c_fnc_tes AND
310 0307     NOT .mom$ab_service_data [svd$gk_pcno_add, svd$V_msg_param] AND
311 0308     NOT .mom$ab_service_data [svd$gk_pcno_nna, svd$V_msg_param] THEN
312 0309     err_detail = nma$c_pcno_pha
313 0310 ELSE
314 0311     IF NOT .mom$ab_service_data [svd$gk_pcno_lpn, svd$V_msg_param] AND
315 0312     NOT .mom$ab_service_data [svd$gk_pcno_$lna, svd$V_msg_param] AND
316 0313     NOT .mom$ab_service_data [svd$gk_pcno_lan, svd$V_msg_param] AND
317 0314     NOT .mom$ab_service_data [svd$gk_pcno_$lnn, svd$V_msg_param] THEN
318 0315     err_detail = nma$c_pcno_pha;
319 0316 END
320 0317 ELSE
321 0318
322 0319     If it's an NI circuit, and the NICE command was LOAD NODE or
323 0320     TRIGGER NODE with no PHYSICAL ADDRESS specified, there must
324 0321     be a hardware address in the volatile database.
325 0322
326 0323 BEGIN
327 0324     IF .mom$ab_service_data [svd$gk_pcno_hwa, svd$b_string_len]
328 0325     EQ 0 THEN
329 0326     err_detail = nma$c_pcno_hwa;
330 0327 END;
331 0328 IF .err_detail NEQ 0 THEN
332 0329 BEGIN
333 0330     mom$ab_msgblock [msb$l_flags] = msb$m_det_fld;
334 0331     mom$ab_msgblock [msb$b_code] = nma$c_ets_pms;
335 0332     mom$ab_msgblock [msb$w_detail] = .err_detail;
336 0333     mom$bld_reply (mom$ab_msgblock, msgsize);
337 0334     $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
338 0335     END;
339 0336 END;
340 0337 END;
341 0338 1 END;

```

! of mom\$get_circuit_type

.PSECT SPLITS,NOWRT,NOEXE,2

```

0000001C 00008 P.AAB: .LONG 28
00000000' 0000C .ADDRESS U.1
                                     .PSECT $OWNS,NOEXE,2

22 00068 ; NFB
      U.1: .BYTE 34
           .BYTE 0
           .BYTE 4
           .BYTE 0
04020041 0006C .LONG 67240001
00000001 00070 .LONG 1
           .BYTE 0
           .BYTE 0
           .WORD 0
04010020 00078 .LONG 67174432
00000000 0007C .LONG 0
           00080 .BLKB 4

                                     U.2= P.AAB

.PSECT $CODE$,NOWRT,2
      .ENTRY MOM$GET_CIRCUIT_TYPE, Save R2,R3,R4,R5,R6,- R7 0231
57 00000000G 00 9E 00002 MOVAB MOM$GL_SERVICE_FLAGS, R7
56 00000000G 00 9E 00009 MOVAB LIB$SIGNAL, R6
55 00000000G 00 9E 00010 MOVAB MOM$AB_NICE_XMIT_BUF, R5
54 00000000G 00 9E 00017 MOVAB MOM$BLD_REPLY, R4
53 00000000G 00 9E 0001E MOVAB MOM$AB_MSGBLOCK, R3
5E 00000000* 10 C2 00025 SUBL2 #16, SP
52 00000000* 00 9A 00028 MOVZBL <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_SLI*137>-
                                     >+8>, LEN 0265
                                     BNEQ 1$ 0266
04 63 22 12 0002F MOVL #2, MOM$AB_MSGBLOCK 0268
08 A3 02 D0 00031 MNEGB #29, MOM$AB_MSGBLOCK+4 0269
A3 6E 1D 8E 00034 MOVZBW #110, MOM$AB_MSGBLOCK+8 0270
      04 AE 9F 0003D PUSHAB MSGSIZE 0271
      53 DD 00040 PUSHL R3
64 04 02 FB 00042 CALLS #2, MOM$BLD_REPLY
      AE DD 00045 PUSHL MSGSIZE 0272
      55 DD 00048 PUSHL R5
66 02070000 8F DD 0004A PUSHL #34013184
      03 FB 00050 CALLS #3, LIB$SIGNAL
      08 AE 9F 00053 1$: PUSHAB P2DSC 0279
00000000' 00 9F 00056 PUSHAB MOM$Q_P2_BUF_DSC
      7E D4 0005C CLRL -(SP)
7E 00000000* 01 CE 0005E MNEGL #1, -(SP) 0280
      00 9F 00061 PUSHAB <<MOM$AB_SERVICE_DATA+<SVD$GK_PCNO_SLI*137>-
                                     >+9> 0279
00000000G 00 52 DD 00067 PUSHL LEN
      06 FB 00069 CALLS #6, MOM$BUILD_P2
00000000G 00 9F 00070 PUSHAB MOM$Q_ACPQIO_BUF_DSC
      04 AE 9F 00076 PUSHAB P3
      10 AE 9F 00079 PUSHAB P2DSC
00000000' 00 9F 0007C PUSHAB U.2 0282

```

00000000G	00		G4	FB	00082	CALLS	#4, MOM\$NETACP_QIO				
	16		50	EB	00089	BLBS	STATUS, 2\$	0286			
		04	AE	9F	0008C	PUSHAB	MSGSIZE	0288			
			53	DD	0008F	PUSHL	R3				
	64		02	FB	00091	CALLS	#2, MOM\$BLD_REPLY				
		04	AE	DD	00094	PUSHL	MSGSIZE	0289			
			55	DD	00097	PUSHL	R5				
		02070000	8F	DD	00099	PUSHL	#34013184				
	66		03	FB	0009F	CALLS	#3, LIB\$SIGNAL				
	50	00000000G	00	D0	000A2	2\$:	MOVL	MOM\$GB_ACPQIO_BUF_DSC+4, R0	0291		
	06		60	D1	000A9	CMPL	(R0), #6				
			01	13	000AC	BEQL	3\$				
				04	000AE	RET					
	67		02	88	000AF	3\$:	BISB2	#2, MOM\$GL_SERVICE_FLAGS	0293		
			50	D4	000B2	CLRL	ERR_DETAIL	0294			
	79		67	EB	000B4	PLBS	MOM\$GL_SERVICE_FLAGS, 8\$	0301			
	72	00000000*	00	EB	000B7	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_PHA*137>->+7>, 8\$	0302			
		02	00000000G	00	91	000BE	CMPB	MOM\$GB_ENTITY_CODE, #2	0304		
				38	12	000C5	BNEQ	6\$			
	12	00000000G	00	91	000C7	CMPB	MOM\$GB_FUNCTION, #18	0306			
			0E	13	000CE	BEQL	4\$				
	07	00000000*	00	EB	000D0	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_ADD*137>->+7>, 4\$	0307			
		1C	00000000*	00	E9	000D7	BLBC	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_NNA*137>->+7>, 5\$	0308		
		26	00000000*	00	EB	000DE	4\$:	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_LPN*137>->+7>, 7\$	0311	
		1F	00000000*	00	EB	000E5	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_\$LNA*-137>>+7>, 7\$	0312		
		18	00000000*	00	EB	000EC	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_LAN*137>->+7>, 7\$	0313		
		11	00000000*	00	EB	000F3	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_\$LNN*-137>>+7>, 7\$	0314		
		50		0A	D0	000FA	5\$:	MOVL	#10, ERR_DETAIL	0315	
				0C	11	000FD	BRB	7\$	0304		
			00000000*	00	95	000FF	6\$:	TSTB	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_HWA*137>->+8>	0325	
				04	12	00105	BNEQ	7\$			
		50		72	8F	9A	00107	MOVZBL	#114, ERR_DETAIL	0326	
					50	D5	0010B	7\$:	TSTL	ERR_DETAIL	0328
					21	13	0010D	BEQL	8\$		
		63			02	D0	0010F	MOVL	#2, MOM\$AB_MSGBLOCK	0330	
	04	A3			1D	8E	00112	MNEGB	#29, MOM\$AB_MSGBLOCK+4	0331	
	08	A3			50	B0	00116	MOVW	ERR_DETAIL, MOM\$AB_MSGBLOCK+8	0332	
				04	AE	9F	0011A	PUSHAB	MSGSIZE	0333	
					53	DD	0011D	PUSHL	R3		
		64			02	FB	0011F	CALLS	#2, MOM\$BLD_REPLY		
				04	AE	DD	00122	PUSHL	MSGSIZE	0334	
					55	DD	00125	PUSHL	R5		
			02070000		8F	DD	00127	PUSHL	#34013184		
		66			03	FB	0012D	CALLS	#3, LIB\$SIGNAL		
					04	00130	8\$:	RET		0338	

; Routine Size: 305 bytes, Routine Base: \$CODE\$ + 00B8

```

343 0339 1 %SBTTL 'mom$get_node_id Get the name of the host node'
344 0340 1 GLOBAL ROUTINE mom$get_node_id (node_add_svd,
345 0341 1 node_name_svd,
346 0342 1 NI_hwa_svd) : NOVALUE =
347 0343 1
348 0344 1 +-
349 0345 1 FUNCTIONAL DESCRIPTION:
350 0346 1 This routine gets the node name and node address needed for a
351 0347 1 load, dump, or loop circuit operation. It uses the SVD indices
352 0348 1 to determine what node name or address is already known (from
353 0349 1 the NICE command or the volatile database), and gets the node
354 0350 1 name, address, and NI hardware address for that node. If no
355 0351 1 node name or address is already known, the executor node is used.
356 0352 1
357 0353 1 FORMAL PARAMETERS:
358 0354 1 NODE_ADD_SVD = Service Data (SVD) table index of entry for node
359 0355 1 address.
360 0356 1 NODE_NAME_SVD = Service Data (SVD) table index of entry for node
361 0357 1 name.
362 0358 1 NI_HWA_SVD = Service Data (SVD) table index of NI hardware
363 0359 1 address for node. Set up only for loop functions.
364 0360 1
365 0361 1 ROUTINE VALUE:
366 0362 1 COMPLETION CODES:
367 0363 1 Signal errors.
368 0364 1
369 0365 1 --
370 0366 1
371 0367 1
372 0368 2 BEGIN
373 0369 2
374 0370 2 $nfbdsc(nfbdsc, show, , ndi
P 0371 2 .add, Search key 1 = node address, oper1 = eql
P 0372 2 .nfbdc_wildcard, Search key 2 = wildcard, oper2 = eql
P 0373 2 .tad, Node address
P 0374 2 .nna, Node name
0375 2 .hwa); NI hardware address
376 0376 2
377 0377 2 MAP
378 0378 2 nfbdsc: VECTOR;
379 0379 2
380 0380 2 LOCAL
381 0381 2 search_key,
382 0382 2 search_len,
383 0383 2 search_value,
384 0384 2 status,
385 0385 2 p2_dsc: VECTOR [2],
386 0386 2 p2_buf_dsc: VECTOR [2],
387 0387 2 p2_buffer: BBLOCK [mom$sk_p2_buf_len],
388 0388 2 nfb: REF BBLOCK,
389 0389 2 p4_dsc: VECTOR [2],
390 0390 2 p4_buffer: BBLOCK [32],
391 0391 2 ptr,
392 0392 2 length;
393 0393 2
394 0394 2
395 0395 2

```

```
400 0396 2 ! If the node name was supplied in the NICE command, use it to get the address.
401 0397 2
402 0398 2 IF .mom$ab_service_data [.node_name_svd, svd$v_msg_param] THEN
403 0399 2 BEGIN
404 0400 2 search_len = .mom$ab_service_data [.node_name_svd, svd$b_string_len];
405 0401 2 search_value = mom$ab_service_data [.node_name_svd, svd$t_string];
406 0402 2 search_key = nfb$c_ndi_nna;
407 0403 2 END
408 0404 2 ELSE
409 0405 2 BEGIN
410 0406 2 search_key = nfb$c_ndi_tad;
411 0407 2 search_len = 0;
412 0408 2
413 0409 2 ! If the node address was supplied in the NICE command, use it to get
414 0410 2 ! the name. Otherwise, get the executor's name and address (this works
415 0411 2 ! because the SVD$L_PARAM is initialized to 0).
416 0412 2
417 0413 2 search_value = .mom$ab_service_data [.node_add_svd, svd$l_param];
418 0414 2 END;
419 0415 2
420 0416 2 ! Get the name and address of the node from the volatile data base.
421 0417 2 ! If it is not found then report an error in the node identification parameter.
422 0418 2
423 0419 2 p2_buf_dsc [0] = mom$k_p2_buf_len;
424 0420 2 p2_buf_dsc [1] = p2_buffer;
425 0421 2 mom$build_p2 (.search_len,
426 0422 2 .search_value,
427 0423 2 -1, 0,
428 0424 2 p2_buf_dsc, p2_dsc);
429 0425 2 nfb = .nfbdsc [1];
430 0426 2 nfb [nfb$l_srch_key] = .search_key;
431 0427 2 p4_dsc [0] = 32;
432 0428 2 p4_dsc [1] = p4_buffer;
433 0429 2 IF mom$netacp_qio ( nfbdsc,
434 0430 2 p2_dsc,
435 0431 2 0,
436 0432 2 p4_dsc) THEN
437 0433 2 BEGIN
438 0434 2 ptr = p4_buffer;
439 0435 2
440 0436 2 ! If the node name and/or address were not supplied in the NICE command,
441 0437 2 ! take the ones returned from the volatile database, and put them into
442 0438 2 ! the service data.
443 0439 2
444 0440 2 IF NOT .mom$ab_service_data [.node_add_svd, svd$v_msg_param] THEN
445 0441 2 mom$ab_service_data [.node_add_svd, svd$l_param] = ..ptr;
446 0442 2 ptr = .ptr + 4;
447 0443 2 length = (.ptr) < 0, 16 >;
448 0444 2 IF NOT .mom$ab_service_data [.node_name_svd, svd$v_msg_param] THEN
449 0445 2 BEGIN
450 0446 2 CH$MOVE (.length, (.ptr + 2),
451 0447 2 mom$ab_service_data [.node_name_svd, svd$t_string]);
452 0448 2 mom$ab_service_data [.node_name_svd, svd$b_string_len] = .length;
453 0449 2 END;
454 0450 2 ptr = .ptr + 2 + .length;
455 0451 2
456 0452 2 ! If it's a LOOP CIRCUIT function, also return the NI hardware address.
```



```

: 457 0453 3 ! For LOAD, TRIGGER, and DUMP functions, the hardware address is obtained
: 458 0454 3 ! with the rest of the service data.
: 459 0455 3 !
: 460 0456 3 IF .mom$gb_function EQL nma$c_fnc_tes THEN
: 461 0457 4 BEGIN
: 462 0458 4 length = .(.ptr)<0,16>;
: 463 0459 4 CH$MOVE (.length, (.ptr + 2),
: 464 0460 4 mom$ab_service_data [.ni_hwa_svd, svd$t_string]);
: 465 0461 4 mom$ab_service_data [.ni_hwa_svd, svd$b_string_len] = .length;
: 466 0462 3 END;
: 467 0463 3 END
: 468 0464 2 ELSE
: 469 0465 2 mom$error (nma$c_sts_ide, nma$c_ent_nod);
: 470 0466 2
: 471 0467 1 END;
! End of mom$get_node_id

```

```

.PSECT $SPLITS$,NOWRT,NOEXE,2
00000024 00010 P.AAC: .LONG 36
00000000 00014 .ADDRESS U.3
.PSECT $SOWNS$,NOEXE,2
22 00084 ; NFB
U.3: .BYTE 34
00 00085 .BYTE 0
02 00086 .BYTE 2
00 00087 .BYTE 0
02010012 00088 .LONG 33619986
00000001 0008C .LONG 1
00 00090 .BYTE 0
00 00091 .BYTE 0
0000 00092 .WORD 0
02010010 00094 .LONG 33619984
02020043 00098 .LONG 33685571
02020057 0009C .LONG 33685591
00000000 000A0 .LONG 0
000A4 .BLKR 4

```

U.4=

P.AAC

```

.PSECT $CODE$,NOWRT,2
03FC 00000 .ENTRY MOM$GET_NODE_ID, Save R2,R3,R4,R5,R6,R7,R8,-; 0340
R9
56 08 59 00000000G 00 9E 00002 MOVAB MOM$AB_SERVICE_DATA+9, R9
5E FF60 CE 9E 00009 MOVAB -160(SP), SP
AC 00000089 8F C5 0000E MULL3 #137, NODE_NAME_SVD, R6 0398
54 FE A946 9E 00017 MOVAB MOM$AB_SERVICE_DATA+7[R0], R4
12 64 E9 0001C BLBC (R4), T$
52 FF A946 9A 0001F MOVZBL MOM$AB_SERVICE_DATA+8[R6], SEARCH_LFN 0400
56 59 C1 00024 ADDL3 R9, R6, SEARCH_VALUE 0401
53 02020043 8F D0 00028 MOVL #33685571, SEARCH_KEY 0402
18 11 0002F BRB 2$ 0398

```

		53	02010010	8F	D0	00031	1\$:	MOVL	#33619984, SEARCH_KEY	:	0406	
				52	D4	00038		CLRL	SEARCH_LEN	:	0407	
50	04	AC	00000089	8F	C5	0003A		MULL3	#137, NODE_ADD_SVD, R0	:	0413	
				6940	9F	00043		PUSHAB	MOM\$AB_SERVICE_DATA+9[R0]	:		
		51		9E	D0	00046		MOVL	@(SP)+, SEARCH_VALUE	:		
	F0	AD	68	8F	9A	00049	2\$:	MOVZBL	#104, P2_BUF_DSC	:	0419	
	F4	AD	28	AE	9E	0004E		MOVAB	P2_BUFFER, P2_BUF_DSC+4	:	0420	
			F8	AD	9F	00053		PUSHAB	P2_DSC	:	0421	
			F0	AD	9F	00056		PUSHAB	P2_BUF_DSC	:		
			7E	D4	00059			CLRL	-(SP)	:		
		7E		01	CE	0005B		MNEGL	#1, -(SP)	:	0423	
				51	DD	0005E		PUSHL	SEARCH_VALUE	:	0422	
				52	DD	00060		PUSHL	SEARCH_LEN	:	0421	
00000000G	00			06	FB	00062		CALLS	#6, MOM\$BUILD_P2	:		
	50	00000000'		00	D0	00069		MOVL	NFB_DSC+4, NFB	:	0425	
	04	AO		53	D0	00070		MOVL	SEARCH_KEY, 4(NFB)	:	0426	
	20	AE		20	D0	00074		MOVL	#32, P2_DSC	:	0427	
	24	AE		6E	9E	00078		MOVAB	P4_BUFFER, P4_DSC+4	:	0428	
			20	AE	9F	0007C		PUSHAB	P4_DSC	:	0429	
				7E	D4	0007F		CLRL	-(SP)	:		
			F8	AD	9F	00081		PUSHAB	P2_DSC	:		
			00000000'	00	9F	00084		PUSHAB	NFB_DSC	:		
00000000G	00			04	FB	0008A		CALLS	#4, MOM\$NETACP_Q10	:		
				52	50	E9	00091	BLBC	R0, 5\$:		
				57	6E	9E	00094	MOVAB	P4_BUFFER, PTR	:	0434	
50	04	AC	00000089	8F	C5	00097		MULL3	#137, NODE_ADD_SVD, R0	:	0440	
06	FE	A940		00	E0	000A0		BBS	#0, MOM\$AB_SERVICE_DATA+7[R0], 3\$:		
				6940	9F	000A6		PUSHAB	MOM\$AB_SERVICE_DATA+9[R0]	:	0441	
				9E	D0	000A9		MOVL	(PTR), @(SP)+	:		
				57	04	C0	000AC	3\$:	ADDL2	#4, PTR	:	0442
				58	67	3C	000AF	MOVZWL	(PTR), LENGTH	:	0443	
				08	64	E8	000B2	BLBS	(R4), 4\$:	0444	
6946	02	A7		58	28	000B5		MOVCS	LENGTH, 2(PTR), MOM\$AB_SERVICE_DATA+9[R6]	:	0447	
	FF	A946		58	90	C00BB		MOVAB	LENGTH, MOM\$AB_SERVICE_DATA+8[R6]	:	0448	
			G2 A847	9E	000C0		4\$:	MOVAB	2(LENGTH)[PTR], PTR	:	0450	
			00000000G	00	91	000C5		CMPB	MOM\$GB_FUNCTION, #18	:	0456	
				24	12	000CC		BNEQ	6\$:		
				58	67	3C	000CE	MOVZWL	(PTR), LENGTH	:	0458	
56	0C	AC	00000089	8F	C5	000D1		MULL3	#137, NI_HWA_SVD, R6	:	0460	
6946	02	A7		58	28	000DA		MOVCS	LENGTH, 2(PTR), MOM\$AB_SERVICE_DATA+9[R6]	:		
	FF	A946		58	90	000E0		MOVAB	LENGTH, MOM\$AB_SERVICE_DATA+8[R6]	:	0461	
					04	000E5		RET		:	0429	
				7E	D4	000E6	5\$:	CLRL	-(SP)	:	0465	
				09	CE	000E8		MNEGL	#9, -(SP)	:		
00000000G	00			02	FB	000EB		CALLS	#2, MOM\$ERROR	:		
				04	000F2		6\$:	RET		:	0467	

; Routine Size: 243 bytes, Routine Base: \$CODE\$ + 01E9

; 472 0468 1

```

474 0469 1 %SBTTL 'mom$getsrvtimer Get the service timer'
475 0470 1 GLOBAL ROUTINE mom$getsrvtimer: NOVALUE =
476 0471 1
477 0472 1 |**
478 0473 1 | FUNCTIONAL DESCRIPTION:
479 0474 1 |
480 0475 1 |     This routine gets the service timer of the circuit to be used.
481 0476 1 |     Since service timer is a line parameter, the routine must access
482 0477 1 |     the volatile data base of the line which corresponds to the
483 0478 1 |     target node's service circuit.
484 0479 1 |
485 0480 1 | FORMAL PARAMETERS:
486 0481 1 |
487 0482 1 | IMPLICIT INPUTS:
488 0483 1 |     Service Data Table (MOM$AB_SERVICE_DATA)
489 0484 1 |
490 0485 1 | ROUTINE VALUE:
491 0486 1 | COMPLETION CODES:
492 0487 1 |
493 0488 1 |     Signal errors.
494 0489 1 |
495 0490 1 | --
496 0491 1 |
497 0492 2 BEGIN
498 0493 2
499 0494 2 LOCAL
500 0495 2     p4_buf_dsc : VECTOR [2],
501 0496 2     qio_p4_buffer : BBLOCK [mom$k_qio_buf_len],
502 0497 2     status;
503 0498 2
504 0499 2 |
505 0500 2 | Get the maintenance parameters from NETACPs node database entry for the
506 0501 2 | target node.
507 0502 2 |
508 0503 2 p4_buf_dsc [0] = mom$k_qio_buf_len;
509 0504 2 p4_buf_dsc [1] = qio_p4_buffer;
510 0505 2
511 0506 2 status = mom$get_voldb_data (nfb$c_db_pli, p4_buf_dsc);
512 0507 2 IF .status THEN
513 0508 2 |
514 0509 2 | Return the service timer value. If the parameter is not set then
515 0510 2 | the value will be -1. This is a suitable value for infinity.
516 0511 2 | Note that the service timer is defaulted to -1 when MOM is initializing.
517 0512 2 |
518 0513 2     mom$ab_service_data [svd$gk_pcli_sti, svd$l_param] = .qio_p4_buffer;
519 0514 1 END;

```

				0000 0000	.ENTRY	MOM\$GETSRVTIMER, Save nothing	: 0470
				CE 9E 00002	MOVAB	-520(SP), SP	: 0503
F8	5E	FDF8	CE	8F 3C 00007	MOVZWL	#512, P4_BUF_DSC	: 0504
FC	AD	0200	6E	9E 0000D	MOVAB	QIO_P4_BUFFER, P4_BUF_DSC+4	: 0506
			F8	AD 9F 00011	PUSHAB	P4_BUF_DSC	: 0506
			05	DD 00014	PUSHL	#5	: 0506

MOMSUBS
V04-000

Special service routines
mom\$getsrvtimer Get the service timer

F 15
16-Sep-1984 02:08:44
14-Sep-1984 12:44:37

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

Page 18
(6)

MO
VO

00000000V	00	02	FB	00016	CALLS	#2, MOM\$GET_VOLDB_DATA	:	
	07	50	E9	0001D	BLBC	STATUS, 1\$:	0507
00000000*	00	6E	D0	00020	MOVL	QIO P4 BUFFER, <<MOM\$AB_SERVICE_DATA+-	:	0513
						<SVB\$GR_PCLI_STI*137>>+9>	:	
		04	00027	1\$:	RET		:	0514

: Routine Size: 40 bytes, Routine Base: \$CODE\$ + 02DC

: 520 0515 1

```

522 0516 1 %SBTTL 'mom$get_voldb_data Get data from volatile database'
523 0517 1 GLOBAL ROUTINE mom$get_voldb_data (database, p4_buf_dsc) : =
524 0518 1
525 0519 1 |++
526 0520 1 | FUNCTIONAL DESCRIPTION:
527 0521 1 | This routine builds the QIO buffers to get information about the
528 0522 1 | target from the volatile data base specified. It issues the
529 0523 1 | QIO to NETACP.
530 0524 1 |
531 0525 1 | Inputs:
532 0526 1 | DATABASE - Database id to use when building the NFB and to determine
533 0527 1 | which of the parameters in the Service Data Table
534 0528 1 | to request.
535 0529 1 | P4_BUF_DSC - P4 buffer descriptor in which to return information.
536 0530 1 |
537 0531 1 | IMPLICIT INPUTS:
538 0532 1 | MOM$GB_ENTITY_CODE
539 0533 1 | MOM$GQ_ENTITY_BUF_DSC
540 0534 1 |
541 0535 1 | OUTPUTS:
542 0536 1 | The P4 buffer described by P4_BUF_DSC contains the maintenance
543 0537 1 | information from the specified database.
544 0538 1 |
545 0539 1 |--
546 0540 1 |
547 0541 2 BEGIN
548 0542 2
549 0543 2 MAP
550 0544 2     p4_buf_dsc : REF VECTOR;
551 0545 2
552 0546 2 LOCAL
553 0547 2     status,
554 0548 2     p2_dsc : VECTOR [2],
555 0549 2     key,
556 0550 2     length,
557 0551 2     address,
558 0552 2     line_len,
559 0553 2     period_ptr,
560 0554 2     nfb : REF BBLOCK,
561 0555 2     nfb_dsc : VECTOR [2],
562 0556 2     nfb_buffer : BBLOCK [mom$k_qio_buf_len],
563 0557 2     msgsize;
564 0558 2
565 0559 2 |
566 0560 2 | Build the NFB, which tells NETACP which information you want returned.
567 0561 2 |
568 0562 2 CH$FILL (0, mom$k_qio_buf_len, nfb_buffer);
569 0563 2 nfb = nfb_buffer;
570 0564 2 nfb [nfb$b_fct] = nfb$c_fc_show;
571 0565 2 nfb [nfb$b_database] = .database;
572 0566 2 nfb [nfb$b_oper] = nfb$c_op_eql;
573 0567 2 nfb [nfb$l_srch2_key] = nfb$c_wildcard;
574 0568 2 nfb [nfb$b_oper2] = nfb$c_op_eql;
575 0569 2
576 0570 2 |
577 0571 2 | Build the P2 buffer for the specified entity. The P2 buffer identifies
578 0572 2 | the specific circuit or node for which information is being requested.

```

```

0579 0573 2 !
0580 0574 2 SELECTONEU .database OF
0581 0575 2 SET
0582 0576 2 [nfb$c_db_ndi]:
0583 0577 2 SELECTIONEU .mom$gb_entity_code OF
0584 0578 2 SET
0585 0579 2
0586 0580 2 [mom$c_circuit]:
0587 0581 2 BEGIN
0588 0582 2 nfb [nfb$l_srch_key] = nfb$c_ndi_sli;
0589 0583 2
0590 0584 2     Figure out what the second search key should be. It's
0591 0585 2     either the node address or the hardware address, depending
0592 0586 2     on whether the physical address is the UNA hardware address
0593 0587 2     or the hiord (node address with DEC NI address space constant)
0594 0588 2     address.
0595 0589 2
0596 0590 2     mom_get_circ_search2 key (key, length, address);
0597 0591 2     nfb [nfb$l_srch2_key] = .key;
0598 0592 2     mom$build_p2 ( .mom$gq_entity_buf_dsc [0],
0599 0593 2                   .mom$gq_entity_buf_dsc [1],
0600 0594 2                   .length, .address,
0601 0595 2                   mom$q_p2_buf_dsc, p2_dsc);
0602 0596 2
0603 0597 2     END;
0604 0598 2 [mom$c_node]:
0605 0599 2 BEGIN
0606 0600 2 nfb [nfb$l_srch_key] = nfb$c_ndi_add;
0607 0601 2 mom$build_p2 ( 0,
0608 0602 2                 .(.mom$gq_entity_buf_dsc [1])<0,16>,
0609 0603 2                 -1, 0,
0610 0604 2                 mom$q_p2_buf_dsc, p2_dsc);
0611 0605 2
0612 0606 2     END;
0613 0607 2 [mom$c_nodebyname]:
0614 0608 2 BEGIN
0615 0609 2 nfb [nfb$l_srch_key] = nfb$c_ndi_nna;
0616 0610 2 mom$build_p2 ( .mom$gq_entity_buf_dsc [0],
0617 0611 2                 .mom$gq_entity_buf_dsc [1],
0618 0612 2                 -1, 0,
0619 0613 2                 mom$q_p2_buf_dsc, p2_dsc);
0620 0614 2
0621 0615 2     END;
0622 0616 2 TES;
0623 0617 2 [nfb$c_db_pli]:
0624 0618 2 BEGIN
0625 0619 2 nfb [nfb$l_srch_key] = nfb$c_pli_nam;
0626 0620 2
0627 0621 2     If the service circuit for the target node is multidrop (eg. DMP-0.1),
0628 0622 2     the corresponding line name will include the period and tributary
0629 0623 2     number. If so, before using the circuit name to access the ACPs line
0630 0624 2     database, eliminate the period and tributary number from the end of the
0631 0625 2     circuit name to get the line name.
0632 0626 2
0633 0627 2     line_len = .mom$ab_service_data [svd$gk_pcno_sli, svd$b_string_len];
0634 0628 2     period_ptr = CH$FIND_CH (.line_len,
0635 0629 2                             mom$ab_service_data [svd$gk_pcno_sli, svd$t_string],

```

```

636 0630      %C'.');
637 0631      IF NOT CH$FAIL (.period_ptr) THEN
638 0632          line_len = .period_ptr - mom$ab_service_data [svd$gk_pcno_sli,
639 0633                          svd$t_string];
640 0634
641 0635      mom$build_p2 (.line_len,
642 0636          mom$ab_service_data [svd$gk_pcno_sli, svd$t_string],
643 0637          -1, 0,
644 0638          mom$q_p2_buf_dsc, p2_dsc);
645 0639      END;
646 0640
647 0641      TES;
648 0642
649 0643      !
650 0644      ! Step through the Service Data Table to find all parameters in the requested
651 0645      ! database. Move these parameter's field IDs into the NFB so that NETACP
652 0646      ! will return their values in the P4 buffer.
653 0647
654 0648      INCR svd index FROM 0 TO svd$e_entry_count DO
655 0649          BEGIN
656 0650              IF .mom$ab_service_data [.svd_index, svd$b_nfb_database]
657 0651                  EQL .database THEN
658 0652                  BEGIN
659 0653                      nfb [nfb$l_fldid] = .mom$ab_service_data [.svd_index, svd$l_nfb_id];
660 0654                      nfb = .nfb + 4;
661 0655                  END;
662 0656          END;
663 0657      nfb [nfb$l_fldid] = 0;
664 0658
665 0659      nfbdsc [0] = nfb [nfb$l_fldid] + 4 - nfb_buffer;
666 0660      nfbdsc [1] = nfb_buffer;
667 0661
668 0662      ! If there is an entry in the volatile data base then NETACP will return the
669 0663      ! data requested in the NFB. Return this data to the calling routine.
670 0664
671 0665      STATUS = mom$netacp_qio (nfbdsc,
672 0666          p2_dsc,
673 0667          p4_buf_dsc [0],
674 0668          .p4_buf_dsc);
675 0669
676 0670      IF NOT .status THEN
677 0671          BEGIN
678 0672              mom$bld_reply (mom$ab_msgblock, msgsize);
679 0673              $signal_msg (mom$ab_nice_xmit_buf, .msgsize);
680 0674          END;
681 0675
682 0676      RETURN .status;
683 0677      ! of mom$get_voldb_data

```

0200	8F	00	56	00000000'	00	007C	00000	.ENTRY	MOM\$GET VOLDB DATA, Save R2,R3,R4,R5,R6	:	0517
			5E	FDE0	CE	9E	00002	MOVAB	MOM\$Q_P2_BUF_DSC, R6	:	
			6E		00	2C	0000E	MOVAB	-544(SP), SP	:	
								MOVCS	#0, (SP), #0, #512, NFB_BUFFER	:	0562

		10	AE	00015					
	52	10	AE	9E 00017	MOVAB	NFB_BUFFER, NFB			0563
	62		22	90 0001B	MOVB	#34, (NFB)			0564
	54	04	AC	D0 0001E	MOVL	DATABASE, R4			0565
02	A2		54	9B 00022	MOVZBW	R4, 2(NFB)			
08	A2		01	D0 00026	MOVL	#1, 8(NFB)			0567
		0C	A2	94 0002A	CLRB	12(NFB)			0568
	02		54	D1 0002D	CMPL	R4, #2			0576
			79	12 00030	BNEQ	4\$			
	50	00000000G	00	9A 00032	MOVZBL	MOM\$GB_ENTITY_CODE, R0			0577
	02		50	91 00039	CMPB	R0, #2			0580
			29	12 0003C	BNEQ	1\$			
	04	A2 02020044	8F	D0 0003E	MOVL	#33685572, 4(NFB)			0582
			5E	DD 00046	PUSHL	SP			0590
		08	AE	9F 00048	PUSHAB	LENGTH			
		10	AE	9F 0004B	PUSHAB	KEY			
00000000V	00		03	FB 0004E	CALLS	#3, MOM_GET_CIRC_SEARCH2_KEY			
08	A2	08	AE	D0 00055	MOVL	KEY, 8(NFB)			0591
		F8	AD	9F 0005A	PUSHAB	P2_DSC			0592
			56	DD 0005D	PUSHL	R6			
		08	AE	DD 0005F	PUSHL	ADDRESS			0594
		10	AE	DD 00062	PUSHL	LENGTH			
			3B	11 00065	BRB	3\$			0593
			50	D5 00067	TSTL	R0			0598
			20	12 00069	BNEQ	2\$			
	04	A2 02010012	8F	D0 0006B	MOVL	#33619986, 4(NFB)			0600
		F8	AD	9F 00073	PUSHAB	P2_DSC			0601
			56	DD 00076	PUSHL	R6			
			7E	D4 00078	CLRL	-(SP)			
	7E		01	CE 0007A	MNEGL	#1, -(SP)			0603
	50	00000000G	00	D0 0007D	MOVL	MOM\$GQ_ENTITY_BUF_DSC+4, R0			0602
	7E		60	3C 00084	MOVZWL	(R0), -(SP)			
			7E	D4 00087	CLRL	-(SP)			0601
			61	11 00089	BRB	7\$			
	01		50	91 0008B	CMPB	R0, #1			0607
			63	12 0008E	BNEQ	8\$			
	04	A2 02020043	8F	D0 00090	MOVL	#33685571, 4(NFB)			0609
		F8	AD	9F 00098	PUSHAB	P2_DSC			0610
			56	DD 0009B	PUSHL	R6			
			7E	D4 0009D	CLRL	-(SP)			
	7E		01	CE 0009F	MNEGL	#1, -(SP)			0612
	7E	00000000G	00	7D 000A2	MOVQ	MOM\$GQ_ENTITY_BUF_DSC, -(SP)			0610
			41	11 000A9	BRB	7\$			
	05		54	D1 000AB	CMPL	R4, #5			0617
			43	12 000AE	BNEQ	8\$			
	04	A2 05020041	8F	D0 000B0	MOVL	#84017217, 4(NFB)			0619
	53	00000000*	00	9A 000B8	MOVZBL	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_SLI*137>->+8>, LINE_LEN			0627
00000000*	00		53	2E 3A 000BF	LOCC	#46, LINE_LEN, <<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_SLI*137>->+9>			0629
			02	12 000C7	BNEQ	5\$			
			51	D4 000C9	CLRL	R1			
			51	D5 000CB	TSTL	PERIOD_PTR			0631
			08	13 000CD	BEQL	6\$			
	50	000G0000*	00	9E 000CF	MOVAB	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_SLI*137>->+9>, R0			0633
53			51	50 C3 000D6	SUBL3	R0, PERIOD_PTR, LINE_LEN			

		FB	AD	9F	000DA	6\$:	PUSHAB	P2_DSC		0636
			56	DD	000DD		PUSHL	R6		
			7E	D4	000DF		CLRL	-(SP)		
	7E		01	CE	000E1		MNEGL	#1, -(SP)		0637
		00000000*	00	9F	000E4		PUSHAB	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_SLI+137>->+9>		0636
			53	DD	000EA		PUSHL	LINE_LEN		
	00000000G	00	06	FB	000EC	7\$:	CALLS	#6, MOM\$BUILD_P2		
		50	01	CE	000F3	8\$:	MNEGL	#1, SVD_INDEX		0648
			22	11	000F6		BRB	10\$		
	51	50	8F	C5	000F8	9\$:	MULL3	#137, SVD_INDEX, R1		0650
54	00000000G0041	08	00	ED	00100		CMPZV	#0, #8, MOM\$AB_SERVICE_DATA+3[R1], R4		0651
			0E	12	0010A		BNEQ	10\$		
		00000000G0041	9F	0010C			PUSHAB	MOM\$AB_SERVICE_DATA[R1]		0653
		10	A2	DO	00113		MOVL	@(SP)+, 16(NFB)		
			52	C0	00117		ADDL2	#4, NFB		0654
	D6	50	8F	F3	0011A	10\$:	AOBLEQ	#SVD\$C_ENTRY_COUNT, SVD_INDEX, 9\$		0648
			10	A2	00122		CLRL	16(NFB)		0657
		50	10	AE	00125		MOVAB	NFB_BUFFER, R0		0659
		52	50	C2	00129		SUBL2	R0, R2		
		F0	AD	9E	0012C		MOVAB	20(R2), NFB\$DSC		
		F4	AD	9E	00131		MOVAB	NFB_BUFFER, NFB\$DSC+4		0660
			08	AC	00136		PUSHL	P4_BUF_DSC		0668
			08	AC	00139		PUSHL	P4_BUF_DSC		0667
			F8	AD	0013C		PUSHAB	P2_DSC		0665
			F0	AD	0013F		PUSHAB	NFB\$DSC		
	00000000G	00	04	FB	00142		CALLS	#4, MOM\$NETACP_QIO		0667
		52	50	DO	00149		MOVL	R0, STATUS		
		26	52	E8	0014C		BLBS	STATUS, 11\$		0670
			0C	AE	0014F		PUSHAB	MSG\$SIZE		0672
		00000000G	00	9F	00152		PUSHAB	MOM\$AB_MSGBLOCK		
	00000000G	00	02	FB	00158		CALLS	#2, MOM\$BLD_REPLY		
			0C	AE	0015F		PUSHL	MSG\$SIZE		0673
		00000000G	00	9F	00162		PUSHAB	MOM\$AB_NICE_XMIT_BUF		
		02070000	8F	DD	00168		PUSHL	#34013T84		
	00000000G	00	03	FB	0016E		CALLS	#3, LIB\$SIGNAL		
		50	52	DO	00175	11\$:	MOVL	STATUS, R0		0676
			04	00178			RET			0677

; Routine Size: 377 bytes, Routine Base: \$CODE\$ + 0304

```
0678 1 %SBTTL 'mom_get_circ_search2_key'
0679 1 GLOBAL ROUTINE mom_get_circ_search2_key (key, length, address) : NOVALUE =
0680 1
0681 1 |++
0682 1 | FUNCTIONAL DESCRIPTION:
0683 1 | This routine is called when preparing to get service data for
0684 1 | the target from the volatile database. At this point the entity
0685 1 | is always MOMSC CIRCUIT, and the operation is a TRIGGER VIA, a
0686 1 | LOAD VIA, or autoservice. In these three cases, there is no node
0687 1 | ID with which to locate the target in the node volatile database.
0688 1 | For point to point circuits, it is sufficient to look for a node
0689 1 | with a service circuit matching the one from the command. For
0690 1 | NI circuits, this routine sets up the second search key to match
0691 1 | in the database.
0692 1
0693 1 | FORMAL PARAMETERS:
0694 1 | KEY      Address to return search key two ID
0695 1 | LENGTH  Address to return search key two length
0696 1 | ADDRESS Address to return search key two address.
0697 1 |
0698 1 | --
0699 1
0700 2 BEGIN
0701 2
0702 2 LOCAL
0703 2     physical_addr_ptr:
0704 2
0705 2 |
0706 2 | At this point the NICE message (operservice) or initial MOP message (auto-
0707 2 | service) has been parsed, and the only parameters in the Service Data table
0708 2 | are from this message. Therefore, the presence of the NI physical address
0709 2 | in the SVD is an indication that the service circuit is an NI.
0710 2
0711 2 IF .mom$ab_service_data [svd$gk_pcno_pha, svd$sv_msg_param] THEN
0712 2 |
0713 2 | If the Physical Address begins with the DEC assigned NI prefix, then
0714 2 | the last word of the Physical Address is the target node's address.
0715 2 | Extract it and use it as the second search key to find the target in
0716 2 | the volatile database (it would actually be sufficient by itself).
0717 2 |
0718 2 | BEGIN
0719 2 |     physical_addr_ptr = mom$ab_service_data [svd$gk_pcno_pha, svd$st_string];
0720 2 | IF .physical_addr_ptr EQL mom$sk_ni_prefix THEN
0721 2 |     BEGIN
0722 2 |         .key = nfb$sc_ndi_add;
0723 2 |         .length = 0;
0724 2 |         .address = .(physical_addr_ptr + 4)<0,16>;
0725 2 |     END
0726 2 | ELSE
0727 2 | |
0728 2 | | Build a P2 buffer that uses the NI hardware address (the entire
0729 2 | | physical address) to find the target's entry in NETACP's node
0730 2 | | database.
0731 2 | |
0732 2 | | BEGIN
0733 2 | |     .key = nfb$sc_ndi_hwa;
0734 2 | |     .length = 6;
```

```

: 742 0735 4      .address = .physical_addr_ptr;
: 743 0736 3      END;
: 744 0737 3      END
: 745 0738 2 ELSE
: 746 0739 2
: 747 0740 2      . The circuit is point-to-point or multipoint. The service circuit
: 748 0741 2      IDs in the node volatile database must be unique for these.
: 749 0742 2
: 750 0743 3 BEGIN
: 751 0744 3 .key = nfb$c_wildcard;
: 752 0745 3 .length = -1;
: 753 0746 3 .address = 0;
: 754 0747 2 END;
: 755 0748 1 END;
! End of mom_get_circ_search2_key

```

			0000 0000	.ENTRY	MOM GET CIRC SEARCH2 KEY, Save nothing	0679
	32	00000000*	00 E9 00002	BLBC	<<MOM\$AB_SERVICE_DATA+<SVDS\$GK_PCNO_PHA*137>-	0711
					>+7>, 2\$	
	50	00000000*	00 9E 00009	MOVAB	<<MOM\$AB_SERVICE_DATA+<SVDS\$GK_PCNO_PHA*137>-	0719
					>+9>, PHYSICAL_ADDR_PTR	
000400AA	8F		60 D1 00010	CMPL	(PHYSICAL_ADDR_PTR), #262314	0720
			11 12 00017	BNEQ	1\$	
04	BC	02010012	8F D0 00019	MOVL	#33619986, @KEY	0722
		08	BC D4 00021	CLRL	@LENGTH	0723
0C	BC	04	A0 3C 00024	MOVZWL	4(PHYSICAL_ADDR_PTR), @ADDRESS	0724
			04 00029	RET		0720
04	BC	02020057	8F D0 0002A 1\$:	MOVL	#33685591, @KEY	0733
08	BC		06 D0 00032	MOVL	#6, @LENGTH	0734
0C	BC		50 D0 00036	MOVL	PHYSICAL_ADDR_PTR, @ADDRESS	0735
			04 0003A	RET		0711
04	BC		01 D0 0003B 2\$:	MOVL	#1, @KEY	0744
08	BC		01 CE 0003F	MNEGL	#1, @LENGTH	0745
		0C	BC D4 00043	CLRL	@ADDRESS	0746
			04 00046	RET		0748

; Routine Size: 71 bytes, Routine Base: \$CODE\$ + 047D

```

: 757 0749 1 %SBTTL 'mom$bldmoprds Build MOP mode running message'
: 758 0750 1 GLOBAL ROUTINE mom$bldmoprds (msgdsc) : NOVALUE =
: 759 0751 1
: 760 0752 1 !++
: 761 0753 1 | FUNCTIONAL DESCRIPTION:
: 762 0754 1 |
: 763 0755 1 |     This routine builds a 'MOP Request Dump Service' message in the
: 764 0756 1 |     MOP transmit buffer.
: 765 0757 1 |
: 766 0758 1 | FORMAL PARAMETERS:
: 767 0759 1 |
: 768 0760 1 |     MSGDSC
: 769 0761 1 |
: 770 0762 1 | --
: 771 0763 1 |
: 772 0764 2 BEGIN
: 773 0765 2
: 774 0766 2 MAP
: 775 0767 2     msgdsc : REF VECTOR;
: 776 0768 2 |
: 777 0769 2 | Move the 'MOP request dump service' function code into the buffer.
: 778 0770 2 |
: 779 0771 2 CH$wCHAR (mop$_fct_rds, mom$ab_mop_xmit_buf);
: 780 0772 2 |
: 781 0773 2 | Set up the descriptor for the return.
: 782 0774 2 |
: 783 0775 2 msgdsc [0] = 1;
: 784 0776 2 msgdsc [1] = mom$ab_mop_xmit_buf;
: 785 0777 2
: 786 0778 1 END;

```

! End of MOM\$BLDMOPRDS

			0004 0000	.ENTRY	MOM\$BLDMOPRDS, Save R2	: 0750
	52	00000000G	00 9E 00002	MOVAB	MOM\$AB_MOP_XMIT_BUF, R2
	62		0C 90 00009	MOVB	#12, MOM\$AB_MOP_XMIT_BUF	: 0771
	50	04	AC D0 0000C	MOVL	MSGDSC, R0	: 0775
	60		01 D0 00010	MOVL	#1, (R0)
04	A0		62 9E 00013	MOVAB	MOM\$AB_MOP_XMIT_BUF, 4(R0)	: 0776
			04 00017	RET		: 0778

; Routine Size: 24 bytes, Routine Base: \$CODES + 04C4

B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{|}~

```

: 788 0779 1 %SBTTL 'mom$bldmopboot Build enter MOP mode message'
: 789 0780 1 GLOBAL ROUTINE mom$bldmopboot (msgdsc) : NOVALUE =
: 790 0781 1
: 791 0782 1 !++
: 792 0783 1 ! FUNCTIONAL DESCRIPTION:
: 793 0784 1 !
: 794 0785 1 !     This routine builds the 'Boot' (trigger) message in the
: 795 0786 1 !     MOP transmit buffer.  This is the old 'Enter MOP Mode' message.
: 796 0787 1 !
: 797 0788 1 ! FORMAL PARAMETERS:
: 798 0789 1 !
: 799 0790 1 !     MSGDSC
: 800 0791 1 !
: 801 0792 1 ! --
: 802 0793 1
: 803 0794 2 BEGIN
: 804 0795 2
: 805 0796 2 MAP
: 806 0797 2     msgdsc : REF VECTOR;
: 807 0798 2
: 808 0799 2 LOCAL
: 809 0800 2     db_passwd_len,
: 810 0801 2     msg_passwd_len,
: 811 0802 2     ptr,
: 812 0803 2     status;
: 813 0804 2
: 814 0805 2 !
: 815 0806 2 ! Build the 'Boot' message.
: 816 0807 2 !
: 817 0808 2 ptr = mom$ab_mop_xmit_buf;
: 818 0809 2 CH$WCHAR_A (mop$fct_emm, ptr);
: 819 0810 2 !
: 820 0811 2 ! Move the service password from the Service Data base to the MOP message.
: 821 0812 2 ! If no password is set then zeros will be used.  The MOP trigger password
: 822 0813 2 ! is always four bytes for point to point and 8 bytes for NI.
: 823 0814 2 !
: 824 0815 2 db_passwd_len = .mom$ab_service_data [svd$gk_pcno_spa, svd$b_string_len];
: 825 0816 2 msg_passwd_len = .db_passwd_len;
: 826 0817 2 IF .mom$gl_service_flags [mom$vn_ni_circ] THEN
: 827 0818 2     msg_passwd_len = 8
: 828 0819 2 ELSE
: 829 0820 2     msg_passwd_len = 4;
: 830 0821 2 ptr = (R$COPY 7.db_passwd_len,
: 831 0822 2     mom$ab_service_data [svd$gk_pcno_spa, svd$t_string],
: 832 0823 2     0, .msg_passwd_len, .ptr);
: 833 0824 2 !
: 834 0825 2 ! The MOP V2.1 Boot message has an 8 byte password (the old version has a
: 835 0826 2 ! 4 byte one) and some extra fields.  Add those extra fields.
: 836 0827 2 !
: 837 0828 2 IF .msg_passwd_len GTR 4 THEN
: 838 0829 3     BEGIN
: 839 0830 3     IF .mom$gl_service_flags [mom$vn_console_carrier_load] THEN
: 840 0831 3         CH$WCHAR_A (mop$c_pro_com, ptr)          ! Load communications processor
: 841 0832 3     ELSE
: 842 0833 3         CH$WCHAR_A (mop$c_pro_sys, ptr);          ! Load system processor
: 843 0834 3     IF .mom$gb_function EQL nma$c_fnc_tri THEN
: 844 0835 3         !

```

```

: 845 0836      | Control: Boot server = system default,
: 846 0837      |           Boot device = system default
: 847 0838      | CH$WCHAR_A (0, ptr)
: 848 0839      | ELSE
: 849 0840      |
: 850 0841      | For load triggers, tell the target to request the load from this
: 851 0842      | system (as opposed to multicasting the load request).
: 852 0843      | Control: Boot server = requesting system,
: 853 0844      |           Boot device = system default
: 854 0845      | CH$WCHAR_A (1, ptr);
: 855 0846      |
: 856 0847      | Software ID - always boot for operating system. I don't see any way
: 857 0848      | for me to tell if I'm loading diagnostics or not.
: 858 0849      |
: 859 0850      | CH$WCHAR_A (-1, ptr);
: 860 0851      | END;
: 861 0852      |
: 862 0853      | Set up the descriptor for the return.
: 863 0854      |
: 864 0855      |
: 865 0856      | msgdsc [0] = .ptr - mom$ab_mop_xmit_buf;
: 866 0857      | msgdsc [1] = mom$ab_mop_xmit_buf;
: 867 0858      |
: 868 0859      | 1 END;

```

! End of mom\$bldmopboot

Address	OpCode	OpCode Hex	OpCode Dec	OpCode Hex	OpCode Dec	Instruction	Address
58	00000000G	00	9E	00002		MOVAB MOM\$GL_SERVICE_FLAGS, R8	0780
57	00000000G	00	9E	00009		MOVAB MOM\$AB_MOP_XMIT_BUF, R7	
53		67	9E	00010		MOVAB MOM\$AB_MOP_XMIT_BUF, PTR	0808
83		06	90	00013		MOVB #6, (PTR)+	0809
50	00000000*	00	9A	00016		MOVZBL <<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_SPA*137>-	0815
						>+8>, DB_PASSWD_LEN	
56		50	D0	0001D		MOVL DB_PASSWD_LEN, MSG_PASSWD_LEN	0816
05		68	01	E1	00020	BEC #1, MOM\$GL_SERVICE_FLAGS, 1\$	0817
		56	08	D0	00024	MOVL #8, MSG_PASSWD_LEN	0818
			03	11	00027	BRB 2\$	
56		04	D0	00029	1\$:	MOVL #4, MSG_PASSWD_LEN	0820
56	00000000*	00	50	2C	0002C	MOVCS DB_PASSWD_LEN, <<MOM\$AB_SERVICE_DATA+-	0823
			63		00035	<SVD\$GK_PCNO_SPA*137>>+9>, #0, =	
						MSG_PASSWD_LEN, (PTR)	
						MSG_PASSWD_LEN, #4	0828
		04	D1	00036		CMLP 7\$	
			22	15	00039	BLEQ #6, MOM\$GL_SERVICE_FLAGS, 3\$	0830
05		68	06	E1	0003B	BBC #1, (PTR)	0831
		63	01	90	0003F	MOVB #1, (PTR)	0833
			02	11	00042	BRB 4\$	
			63	94	00044	CLRB (PTR)	0831
			53	D6	00046	INCL PTR	0834
11	00000000G	00	91	00048		CMPB MOM\$GB_FUNCTION, #17	0838
			04	12	0004F	BNEQ 5\$	0845
			63	94	00051	CLRB (PTR)	
			03	11	00053	BRB 6\$	
63		01	90	00055	5\$:	MOVB #1, (PTR)	0838
			53	D6	00058	INCL PTR	

MOMSUBS
V04-000

Special service routines
mom\$bldmopboot Build enter MOP mode message

D 16
16-Sep-1984 02:08:44
14-Sep-1984 12:44:37

VAX-11 @liss-32 V4.0-742
DISK\$VMSMASTER:[MOM.SRC]MOMSUBS.B32;1 Page 29
(10)

	83		01	8E	0005A	MNEGB	#1, (PTR)+	: 0851
	50	04	AC	D0	0005D	MOVL	MSGDSC, R0	: 0856
	51		67	9E	00061	MOVAB	MOM\$AB_MOP_XMIT_BUF, R1	:
6C	53		51	C3	00064	SUBL3	R1, PTR, (R0)	:
	04	A0	67	9E	00068	MOVAB	MOM\$AB_MOP_XMIT_BUF, 4(R0)	: 0857
			04	0006C		RET		: 0859

; Routine Size: 109 bytes, Routine Base: \$CODE\$ + 04DC

```

870 0860 1 %SBTTL 'mom$bldmopplt Build MOP Parameter Load with Transfer Address message'
871 0861 1 GLOBAL ROUTINE mom$bldmopplt (plt_msg_dsc, load_seg_num,
872 0862 1 transfer_addr) : NOVALUE =
873 0863 1
874 0864 1 |++
875 0865 1 | FUNCTIONAL DESCRIPTION:
876 0866 1 | This routine is called to build the MOP Parameter Load with
877 0867 1 | Transfer message which is sent to the target node at the end of
878 0868 1 | a down line load.
879 0869 1 |
880 0870 1 | FORMAL PARAMETERS:
881 0871 1 | PLT_MSG_DSC - Descriptor of buffer for MOP Parameter Load with
882 0872 1 | Transfer message.
883 0873 1 | LOAD_SEG_NUM - Number of load segments loaded modulo 256.
884 0874 1 | TRANSFER_ADDR - Address to start executing image just loaded.
885 0875 1 |
886 0876 1 | IMPLICIT OUTPUTS:
887 0877 1 |
888 0878 1 | ROUTINE VALUE:
889 0879 1 | COMPLETION CODES:
890 0880 1 | --
891 0881 1
892 0882 2 BEGIN
893 0883 2
894 0884 2 MAP
895 0885 2     plt_msg_dsc : REF VECTOR,
896 0886 2     load_seg_num: BYTE;
897 0887 2
898 0888 2 LOCAL
899 0889 2     len,
900 0890 2     ptr,
901 0891 2     node_addr: WORD,
902 0892 2     date_time : VECTOR [7, WORD],
903 0893 2     century,
904 0894 2     year;
905 0895 2
906 0896 2 |
907 0897 2 | If the load file was a bootstrap then send an empty memory load with
908 0898 2 | transfer address message.
909 0899 2 |
910 0900 2 IF .mom$ab_service_data [svd$gk_pcno_sty, svd$_param] NEQU nma$_soft_osys THEN
911 0901 2 BEGIN
912 0902 2
913 0903 2     ptr = mom$ab_mop_xmit_buf;
914 0904 2
915 0905 2     CH$W(HAR_A (mop$_fct_mlt, ptr);           ! Function code
916 0906 2     CH$W(HAR_A (.load_seg_num, ptr);         ! Load segment number
917 0907 2     (.PTR)<0,32> = 0;                          ! Zero load address
918 0908 2     ptr = .ptr + 4;                             ! Skip load address
919 0909 2
920 0910 2 | Output the MOP message to the debug log.
921 0911 2 |
922 0912 2     mom$debug_txt (dbg$_srvtrc,
923 0913 2     $ASCII ('transmitting empty memory load with transfer address.')}
924 0914 2     );
925 0915 2
926 0916 2 END

```



```

0917 2 ELSE
0918 BEGIN
0919
0920 The load file was the system image so send a parameter load with transfer
0921 address message.
0922
0923 ptr = mom$ab_mop_xmit_buf;
0924 CH$WCHAR_A (mop$fct_plt, ptr); ! Function code
0925 CH$WCHAR_A (.load_seg_num, ptr); ! Load segment number
0926
0927 If target node name specified then add it to message.
0928
0929 len = .mom$ab_service_data [svd$gk_pcno_nna, svd$b_string_len];
0930 IF .len NEQ 0 THEN
0931 BEGIN
0932 CH$WCHAR_A (mop$c_par_nna, ptr); ! Parameter code
0933 CH$WCHAR_A (.len, ptr); ! Name length
0934 PTR = CH$MOVE (.len, ! Name
0935 mom$ab_service_data [svd$gk_pcno_nna, svd$t_string],
0936 .ptr);
0937 END;
0938
0939 Add target node address to message. If address not specified then
0940 program error.
0941
0942 CH$WCHAR_A (mop$c_par_nad, ptr); ! Parameter code
0943 CH$WCHAR_A (2, ptr); ! Address length
0944 node_addr = .mom$ab_service_data [svd$gk_pcno_add, svd$l_param];
0945
0946 ! If it's a phase III node, mask out the area number in the node address.
0947 ! DECnet Phase III did not include areas.
0948
0949 IF .mom$ab_service_data [svd$gk_pcno_snv, svd$l_param] EQL nma$c_nodsnu_ph3
0950 THEN
0951 BEGIN
0952 MAP node_addr: BBLOCK;
0953 node_addr [nma$v_area] = 0;
0954 END;
0955 ptr = CH$MOVE (2, node_addr, .ptr);
0956
0957
0958 If the host node name is specified then add it to the message.
0959
0960 len = .mom$ab_service_data [svd$gk_pcno_$hna, svd$b_string_len];
0961 IF .len NEQ 0 THEN
0962 BEGIN
0963 CH$WCHAR_A (mop$c_par_hna, ptr); ! Parameter code
0964 CH$WCHAR_A (.len, ptr); ! Name length
0965 PTR = CH$MOVE (.len, ! Name
0966 mom$ab_service_data [svd$gk_pcno_$hna, svd$t_string],
0967 .ptr);
0968 END;
0969
0970
0971 If the host address is specified then add it to the message.
0972
0973 IF .mom$ab_service_data [svd$gk_pcno_iho, svd$l_param] NEQ 0 THEN

```

```

984 0974 4 BEGIN
985 0975 4 CH$WCHAR_A (mop$C_par_had, ptr); ! Parameter code
986 0976 4 CH$WCHAR_A (2, ptr); ! Address length
987 0977 4 node_addr = .mom$ab_service_data [svd$gk_pcno_iho, svd$l_param];
988 0978 4
989 0979 4 ! If it's a phase III node, mask out the area number in the node address.
990 0980 4
991 0981 4 IF .mom$ab_service_data [svd$gk_pcno_snv, svd$l_param] EQL
992 0982 4 nma$C_nodsnn_ph3 THEN
993 0983 5 BEGIN
994 0984 5 MAP node_addr: BBLOCK;
995 0985 5 node_addr [nma$V_area] = 0;
996 0986 4 END;
997 0987 4 ptr = CH$MOVE (2, node_addr, .ptr);
998 0988 4 END;
999 0989 4
1000 0990 4
1001 0991 4 ! If it's not a phase III node, add the system time to the message
1002 0992 4
1003 0993 4 IF .mom$ab_service_data [svd$gk_pcno_snv, svd$l_param] NEQ
1004 0994 4 nma$C_nodsnn_ph3 THEN
1005 0995 4 BEGIN
1006 0996 4 CH$WCHAR_A (mop$C_par_hti, ptr);
1007 0997 4 CH$WCHAR_A (10, ptr);
1008 0998 4 $NUMTIM (TIMBUF = date_time);
1009 0999 4
1010 1000 4 ! The parameter load with transfer message requires that the year be
1011 1001 4 ! broken up into a century and a year. Do that.
1012 1002 4
1013 1003 4 century = .date_time [0] /100;
1014 1004 4 year = .date_time [0] MOD 100;
1015 1005 4
1016 1006 4 ! The rest of the date/time string required in the MOP Parameter Load with
1017 1007 4 ! Transfer message is in the same order as that returned by the $NUMTIM
1018 1008 4 ! system service. Put the string into the MOP message, converting the words
1019 1009 4 ! to bytes.
1020 1010 4
1021 1011 4 CH$WCHAR_A (.century, ptr);
1022 1012 4 CH$WCHAR_A (.year, ptr);
1023 1013 4 INCR i FROM 1 TO 6 DO
1024 1014 4 CH$WCHAR_A (.date_time [.i], ptr);
1025 1015 4
1026 1016 4 ! Fill in the Time Differential Factor hours and minutes as 0. VMS
1027 1017 4 ! doesn't keep Greenwich Mean Time around for figuring these out with.
1028 1018 4
1029 1019 4 ptr = CH$FILL (0, 2, .ptr);
1030 1020 4 END;
1031 1021 4
1032 1022 4
1033 1023 4 ! Add the end mark.
1034 1024 4
1035 1025 4 CH$WCHAR_A (0, ptr);
1036 1026 4
1037 1027 4 ! Output the trace message.
1038 1028 4
1039 1029 4 mom$debug txt ( dbg$C_srvtrc,
1040 1030 4 $ASCII ('Transmitting parameter load with transfer address.')
```


Address	OpCode	Parameter	Load	with T	OpCode	Instruction	Address	
63 00000000*	00		56	28	0003F	MOVC3	LEN, <<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_NNA--*137>>+9>, (PTR)	0936
	83	0202	8F	80	00047	2\$: MOVW	#514, (PTR)+	0942
	57	00000000*	00	80	0004C	MOVW	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_ADD*137>->+9>, NODE_ADDR	0944
	58	00000000*	00	D0	00053	MOVL	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_SNV*137>->+9>, R8	0949
			59	D4	0005A	CLRL	R9	
			58	D5	0005C	TSTL	R8	
			07	12	0005E	BNEQ	3\$	
			59	D6	00060	INCL	R9	
	57	FC00	8F	AA	00062	BICW2	#64512, NODE_ADDR	0953
	83		57	80	00067	3\$: MOVW	NODE_ADDR, (PTR)+	0955
	56	00000000*	00	9A	0006A	MOVZBL	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_\$HNA+-137>>+8>, LEN	0960
			0E	13	00071	BEQL	4\$	0961
	83		03	90	00073	MOVB	#3, (PTR)+	0963
	83		56	90	00076	MOVB	LEN, (PTR)+	0965
63 00000000*	00		56	28	00079	MOVC3	LEN, <<MOM\$AB_SERVICE_DATA+-<SVD\$GK_PCNO_\$HNA*1375>>+9>, (PTR)	0968
	50	00000000*	00	D0	00081	4\$: MOVL	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_IHO*137>->+9>, R0	0973
			13	13	00088	BEQL	6\$	
	83	0204	8F	80	0008A	MOVW	#516, (PTR)+	0975
	57		50	80	0008F	MOVW	R0, NODE_ADDR	0977
	05		59	E9	00092	BLBC	R9, 5\$	0981
	57	FC00	8F	AA	00095	BICW2	#64512, NODE_ADDR	0985
	83		57	80	0009A	5\$: MOVW	NODE_ADDR, (PTR)+	0987
			58	D5	0009D	6\$: TSTL	R8	0993
			3F	13	0009F	BEQL	8\$	
	83	0A05	8F	80	000A1	MOVW	#2565, (PTR)+	0996
		04	7E	D4	000A6	CLRL	-(SP)	0998
			AE	9F	000A8	PUSHAB	DATE TIME	
00000000G	00		02	FB	000AB	CALLS	#2, SYSSNUMTIM	
	51		6E	3C	000B2	MOVZWL	DATE TIME, CENTURY	1003
	51	00000064	8F	C6	000B5	DIVL2	#100, CENTURY	
	50		6E	3C	000BC	MOVZWL	DATE TIME, YEAR	1004
7E	00		01	7A	000BF	EMUL	#1, YEAR, #0, -(SP)	
50	50		8E	7B	000C4	EDIV	#100, (SP)+, YEAR, YEAR	
	83	00000064	51	90	000CD	MOVB	CENTURY, (PTR)+	1011
	83		50	90	000D0	MOVB	YEAR, (PTR)+	1012
	50		01	D0	000D3	MOVL	#1, I	1014
	83		6E40	33	000D6	7\$: CVTWB	DATE TIME[I], (PTR)+	
F8	50		06	F3	000DA	AOBLEQ	#6, I, 7\$	
			83	B4	000DE	CLRW	(PTR)+	1019
			83	94	000E0	8\$: CLRB	(PTR)+	1025
		00000000'	00	9F	000E2	PUSHAB	P.AAF	1030
			06	DD	000E8	9\$: PUSHL	#6	1029
00000000G	00		02	FB	000EA	CALLS	#2, MOM\$DEBUG_TXT	
	83	0C	AC	D0	000F1	MOVL	TRANSFER_ADDR, (PTR)+	1037
	50	04	AC	D0	000F5	MOVL	PLT MSG DSC, R0	1041
	51		6A	9E	000F9	MOVAB	MOM\$AB_MOP_XMIT_BUF, R1	
60	53		51	C3	000FC	SUBL3	R1, PTR, (R0)	
	04	A0	6A	9E	00100	MOVAB	MOM\$AB_MOP_XMIT_BUF, 4(R0)	1042
			04	00104		RET		1043

; Routine Size: 261 bytes, Routine Base: \$CODE\$ + 0549

: 1054 1044 1
: 1055 1045 1 END
: 1056 1046 1
: 1057 1047 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	168	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	148	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	1614	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[MOM.OBJ]MOMLIB.L32;1	194	49	25	21	00:00.1
-\$255\$DUA28:[SHRLIB]NMALIBRY.L32;1	887	14	1	47	00:00.2
-\$255\$DUA28:[SHRLIB]NET.L32;1	1279	22	1	63	00:00.3
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	3	0	581	00:03.2

COMMAND QUALIFIERS

:
: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:MOMSUBS/OBJ=OBJ\$:MOMSUBS MSRC\$:MOMSUBS/UPDATE=(ENH\$:MOMSUBS)
: Size: 1614 code + 316 data bytes
: Run Time: 00:34.9
: Elapsed Time: 01:12.0
: Lines/CPU Min: 1801
: Lexemes/CPU-Min: 15995
: Memory Used: 211 pages
: Compilation Complete

