


```

NN      NN      CCCCCCCC  PPPPPPPP  PPPPPPPP  RRRRRRRR  SSSSSSSS  AAAAAA  CCCCCCCC  TTTTTTTTTT
NN      NN      CCCCCCCC  PPPPPPPP  PPPPPPPP  RRRRRRRR  SSSSSSSS  AAAAAA  CCCCCCCC  TTTTTTTTTT
NN      NN      CC          PP          PP          RR          SS          AA          CC          TT
NN      NN      CC          PP          PP          RR          SS          AA          CC          TT
NNNN    NN      CC          PP          PP          RR          SS          AA          CC          TT
NNNN    NN      CC          PP          PP          RR          SS          AA          CC          TT
NN  NN  NN      CC          PPPPPPPP  PPPPPPPP  RRRRRRRR  SSSSSS  AA          CC          TT
NN  NN  NN      CC          PPPPPPPP  PPPPPPPP  RRRRRRRR  SSSSSS  AA          CC          TT
NN      NNNN    CC          PP          PP          RR  RR          SS          AAAAAAAAAA  CC          TT
NN      NNNN    CC          PP          PP          RR  RR          SS          AAAAAAAAAA  CC          TT
NN      NN      CC          PP          PP          RR  RR          SS          AA          CC          TT
NN      NN      CC          PP          PP          RR  RR          SS          AA          CC          TT
NN      NN      CCCCCCCC  PP          PP          RR          RR  SSSSSSSS  AA          CCCCCCCC  TT
NN      NN      CCCCCCCC  PP          PP          RR          RR  SSSSSSSS  AA          CCCCCCCC  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 %TITLE 'Parse Data and Action Routines'
2 0002 0 MODULE NCPPRSACT (IDENT = 'V04-000'
3 0003 0 ADDRESSING_MODE(EXTERNAL=GENERAL),
4 0004 0 ADDRESSING_MODE(NONEXTERNAL=GENERAL)) =
5 0005 1 BEGIN
6 0006 1
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: Network Control Program (NCP)
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 Data and Action routines for parsing
38 0038 1
39 0039 1 ENVIRONMENT: VAX/VMS Operating System
40 0040 1
41 0041 1 AUTHOR: Darrell Duffy , CREATION DATE: 28-August-79
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 V03-002 RPG0002 Bob Grosso 29-Jul-1982
46 0046 1 Add new routine to supply prompting help.
47 0047 1
48 0048 1 V03-001 RPG0001 Bob Grosso 30-Jun-1982
49 0049 1 Add routines to support channel lists.
50 0050 1
51 0051 1 V002 TMH0002 Tim Halvorsen 04-Nov-1981
52 0052 1 Remove duplicate definition of NCP$_NXT_STATE as
53 0053 1 both an external and a global. It should have been
54 0054 1 one or the other.
55 0055 1
56 0056 1 V001 TMH0001 Tim Halvorsen 18-Jun-1981
57 0057 1 Make all external references longword relative.

```

NCPPRSACT
V04-000

Parse Data and Action Routines

: 58

0058 1 !--

^{D 9}
15-Sep-1984 23:51:04
14-Sep-1984 12:48:15

YAX-11 Bliss-32 V4.0-742
[NCP.SRC]NCPPRSACT.B32;1

Page 2
(1)

NCI
VO

```

60 0059 1 %SBTTL 'Definitions'
61 0060 1
62 0061 1
63 0062 1
64 0063 1
65 0064 1
66 0065 1 FORWARD ROUTINE
67 0066 1 NCPSSIG_CMDERR; ! Signal a command error
68 0067 1
69 0068 1
70 0069 1
71 0070 1
72 0071 1
73 0072 1 LIBRARY 'LIBS:NCPLIBRY.L32';
74 0073 1 LIBRARY 'SYSSLIBRARY:STARLET.L32';
75 0074 1
76 0075 1
77 0076 1
78 0077 1
79 0078 1
80 0079 1 GLOBAL LITERAL
81 0080 1 ACTSC_RNGLSTMAX = 2 * MAX_RNGLST_PAIRS;
82 0081 1
83 0082 1 GLOBAL
84 0083 1 ACTSGA_RNGLST : VECTOR [ACTSC_RNGLSTMAX + 1, WORD],
85 0084 1 Channel list vector,
86 0085 1 ACTSGA_RNGLST [0] contains count
87 0086 1 NCP$NXT STATE : VECTOR [2], ! Next state table and keytable to use
88 0087 1 NCP$PRSCMD_DSC : VECTOR [2]; ! Descriptor of the parsed command
89 0088 1
90 0089 1
91 0090 1
92 0091 1
93 0092 1
94 0093 1 EXTERNAL LITERAL
95 0094 1 NCP$_AMBCMD, ! Error status for ambiguous command
96 0095 1 NCP$_INVCMD, ! Error status for invalid command
97 0096 1 NCP$_CMDCAN, ! Command canceled
98 0097 1 NCP$_CMDERR, ! I/O error
99 0098 1 NCP$_FIELDLM, ! Too many fields
100 0099 1 NCP$_NOTDONE, ! Prompt for non-terminal command
101 0100 1 NCP$_PRMRNG, ! Parameter range status code
102 0101 1 NCP$_PRMLEN, ! Parameter length status code
103 0102 1 NCP$_SYNTAX, ! Syntax error status
104 0103 1 LIBS_SYNTAXERR ! Syntax error status from LIBSTPARSE
105 0104 1
106 0105 1
107 0106 1 EXTERNAL ROUTINE
108 0107 1 LBR$OUTPUT_HELP, ! Prompting help
109 0108 1 LIB$GET_INPUT, ! for prompting for help
110 0109 1 LIB$PUT_OUTPUT, ! for printing help
111 0110 1 LIBSTPARSE; ! The table parser
112 0111 1
113 0112 1 EXTERNAL ROUTINE
114 0113 1 NCP$WRITE_LINE, ! Write a line to sys$output
115 0114 1 NCP$READ_CMD, ! Read a command with continuation
116 0115 1 NCP$CMD_TERM_Q ! Is input device a terminal?

```

```
: 117      0116 1      ;  
: 118      0117 1      ;  
: 119      0118 1  EXTERNAL  
: 120      0119 1      NCPS_CMDBUF_DSC : VECTOR,  
: 121      0120 1      ACTSGL_PMT_0  
: 122      0121 1      ;
```

```
! Command buffer descriptor  
! True for prompting active
```

```

124 0122 1 %SBTTL 'NCP$PARSE_CMD Parse Command'
125 0123 1 GLOBAL ROUTINE NCP$PARSE_CMD (INP_DSC, ST_TBL, KEY_TBL, RTN_DSC) = !
126 0124 1
127 0125 1 ++
128 0126 1 FUNCTIONAL DESCRIPTION:
129 0127 1     Calls LIB$TPARSE with the parse state table
130 0128 1
131 0129 1 FORMAL PARAMETERS:
132 0130 1
133 0131 1     INP_DSC      Address of descriptor of line
134 0132 1     ST_TBL      Address of state table to use
135 0133 1     KEY_TBL     Address of keyword table
136 0134 1     RTN_DSC     Address of descriptor to receive remainder
137 0135 1                of command line
138 0136 1
139 0137 1 IMPLICIT INPUTS:
140 0138 1
141 0139 1     NONE
142 0140 1
143 0141 1 IMPLICIT OUTPUTS:
144 0142 1
145 0143 1     NONE
146 0144 1
147 0145 1 ROUTINE VALUE:
148 0146 1 COMPLETION CODES:
149 0147 1
150 0148 1     Return status from LIB$TPARSE is signaled if syntax error
151 0149 1
152 0150 1 SIDE EFFECTS:
153 0151 1
154 C152 1     NONE
155 0153 1
156 0154 1 --
157 0155 1
158 0156 1 BEGIN
159 0157 1
160 0158 1 MAP
161 0159 1     INP_DSC : REF VECTOR [2],      ! Dsc of input line
162 0160 1     RTN_DSC : REF VECTOR [2],      ! Returned dsc of remainder
163 0161 1     :
164 0162 1
165 0163 1 LOCAL
166 0164 1     STATUS      ! Returned status
167 0165 1     :
168 0166 1     :
169 0167 1 OWN
170 0168 1     PARSE_STATE :      ! Parse state table to LIB$TPARSE
171 0169 1     :      BBLOCK [TPASK_LENGTH0]
172 0170 1     :
173 0171 1
174 0172 1 NCP$PRSCMD_DSC [0] = .INP_DSC [0]; ! Save the descriptor of the command
175 0173 1 NCP$PRSCMD_DSC [1] = .INP_DSC [1];
176 0174 1
177 0175 1 PARSE_STATE [TPASL_COUNT] =      ! Set count of arg block
178 0176 1     TPASK_COUNT0;
179 0177 1 PARSE_STATE [TPASL_OPTIONS] =    ! Set for minimum abbreviation
180 0178 1

```

```

181 0179 2 TPASM ABBREV;
182 0180 2 PARSE_STATE [TPASL_STRINGCNT] = ! Setup the initial command string
183 0181 2 INP_DSC [0];
184 0182 2 PARSE_STATE [TPASL_STRINGPTR] =
185 0183 2 .INP_DSC [1];
186 0184 2
187 0185 2 NCPS_NXT_STATE [0] = .ST_TBL; ! Setup the first round of state
188 0186 2 NCPS_NXT_STATE [1] = .KEY_TBL;
189 0187 2
190 0188 2 DO
191 0189 2 BEGIN
192 0190 2 LOCAL
193 0191 2 STATES, ! Temp for state table address
194 0192 2 KEYS ! Temp for keyword table address
195 0193 2
196 0194 2 STATES = .NCPS_NXT_STATE [0]; ! Set up this round
197 0195 2 KEYS = .NCPS_NXT_STATE [1];
198 0196 2
199 0197 2 NCPS_NXT_STATE = 0; ! Zero next round
200 0198 2
201 0199 2 STATUS = LIB$TPARSE (PARSE_STATE, ! Parse the string
202 0200 2 .STATES, .KEYS);
203 0201 2 END
204 0202 2 ! While no error and there is a next
205 0203 2 WHILE .STATUS AND (.NCPS_NXT_STATE [0] NEQ 0) ! round, keep going
206 0204 2 ;
207 0205 2
208 0206 2 IF NOT .STATUS ! Returned an error?
209 0207 2 THEN
210 0208 2 BEGIN ! Yes, then signal it somehow
211 0209 2 IF .STATUS EQLU LIB$SYNTAXERR ! Is it a vanilla syntax error?
212 0210 2 THEN ! Yes, then build the arguments
213 0211 2 NCP$SIG_CMDERR ! Signal a command syntax error
214 0212 2 ( (IF .PARSE_STATE [TPASL_AMBIG]
215 0213 2 THEN NCPS_AMBCMD ! Ambiguous keyword or
216 0214 2 ELSE NCPS_SYNTAX ! Syntax error
217 0215 2 ) OR ST$SR SEVERE, ! Severe error to stop
218 0216 2 .PARSE_STATE [TPASL_TOKENCNT],
219 0217 2 .PARSE_STATE [TPASL_TOKENPTR],
220 0218 2 .PARSE_STATE [TPASL_STRINGCNT],
221 0219 2 .PARSE_STATE [TPASL_STRINGPTR]
222 0220 2 )
223 0221 2 ELSE
224 0222 2 SIGNAL (.STATUS) ! Punt the signal of anything else
225 0223 2
226 0224 2 END
227 0225 2 ;
228 0226 2
229 0227 2 RTN_DSC [0] = .PARSE_STATE ! Return the remainder of the string
230 0228 2 [TPASL_STRINGCNT];
231 0229 2 RTN_DSC [1] = .PARSE_STATE
232 0230 2 [TPASL_STRINGPTR];
233 0231 2 RETURN .STATUS ! and the status of the call
234 0232 2
235 0233 2 END;

```


.TITLE NCPPRSACT Parse Data and Action Routines
.IDENT \V04-000\

.PSECT \$OWNS\$,NOEXE,2

00000 PARSE_STATE:
.BLKB 36

.PSECT \$GLOBALS\$,NOEXE,2

00000 ACT\$GA_RNGLST::
.BLKB 66

00042 .BLKB 2

00044 NCP\$NXT_STATE::
.BLKB 8

0004C NCP\$PRSCMD_DSC::
.BLKB 8

ACT\$C_RNGLSTMAX== 32

.EXTRN NCP\$AMBCMD, NCP\$INVCMD
.EXTRN NCP\$CMDCAN, NCP\$CMDERR
.EXTRN NCP\$FIELDLIM, NCP\$NOTDONE
.EXTRN NCP\$PRMRNG, NCP\$PRMLEN
.EXTRN NCP\$SYNTAX, LIB\$SYNTAXERR
.EXTRN LIB\$OUTPUT_HELP
.EXTRN LIB\$GET_INPUT, LIB\$PUT_OUTPUT
.EXTRN LIB\$PARSE, NCP\$WRITE_CINE
.EXTRN NCP\$READ_CMD, NCP\$CMD_TERM_Q
.EXTRN NCP\$CMDBUF_DSC
.EXTRN ACT\$GL_PMT_Q

.PSECT \$CODE\$,NOWRT,2

			001C	00000	.ENTRY	NCP\$PARSE_CMD, Save R2,R3,R4	: 0123
	54	00000000'	00	9E 00002	MOVAB	NCP\$NXT_STATE, R4	
	53	00000000'	00	9E 00009	MOVAB	PARSE_STATE+8, R3	
	50	04	AC	D0 00010	MOVL	INP_DSC, R0	: 0173
08	A4		60	7D 00014	MOVQ	(R0), NCP\$PRSCMD_DSC	
F8	A3		08	DC 00018	MOVL	#8, PARSE_STATE	: 0176
FC	A3		02	D0 0001C	MOVL	#2, PARSE_STATE+4	: 0178
	63		60	7D 00020	MOVQ	(R0), PARSE_STATE+8	: 0181
	64	08	AC	7D 00023	MOVQ	ST_TBL, NCP\$NXT_STATE	: 0185
	51		64	D0 00027	MOVL	NCP\$NXT_STATE, STATES	: 0194
	50	04	A4	D0 0002A	MOVL	NCP\$NXT_STATE+4, KEYS	: 0195
			64	D4 0002E	CLRL	NCP\$NXT_STATE	: 0197
			50	DD 00030	PUSHL	KEYS	: 0200
			51	DD 00032	PUSHL	STATES	
		F8	A3	9F 00034	PUSHAB	PARSE_STATE	: 0199
00000000G	00		03	FB 00037	CALLS	#3, LIB\$PARSE	
	52		50	D0 0003E	MOVL	R0, STATUS	
	07		52	E9 00041	BLBC	STATUS, 2\$: 0203
			64	D5 00044	TSTL	NCP\$NXT_STATE	
			DF	12 00046	BNEQ	1\$	
	3A		52	E8 00048	BLBS	STATUS, 6\$: 0206
00000000G	8F		52	D1 0004B	CMPL	STATUS, #LIB\$SYNTAXERR	: 0209
			28	12 00052	BNEQ	5\$	
	7E		63	7D 00054	MOVQ	PARSE_STATE+8, -(SP)	: 0218

	7E	08	A3	7D	00057	MOVQ	PARSE_STATE+16, -(SP)	:	0216
	09	FE	A3	E9	0005B	BLBC	PARSE_STATE+6, 3\$:	0212
	50	00000000G	8F	D0	0005F	MOVL	#NCP\$_AMBCMD, R0	:	
			07	11	00066	BRB	4\$:	
	50	00000000G	8F	D0	00068	3\$:	MOVL	#NCP\$ SYNTAX, R0	:
7E	50		04	C9	0006F	4\$:	BISL3	#4, R0, -(SP)	:
00000000V	00		05	FB	00073		CALLS	#5, NCP\$SIG.CMDERR	:
			09	11	0007A		BRB	6\$:
			52	DD	0007C	5\$:	PUSHL	STATUS	:
00000000G	00		01	FB	0007E		CALLS	#1, LIB\$SIGNAL	:
	50	10	AC	D0	00085	6\$:	MOVL	RTN DSC, R0	:
	60		63	7D	00089		MOVQ	PARSE_STATE+8, (R0)	:
	50		52	D0	0008C		MOVL	STATUS, R0	:
			04	0008F		RET		:	0233

; Routine Size: 144 bytes. Routine Base: \$CODE\$ + 0000

```

237 0234 1 %SBTTL 'NCPSSIG_CMDERR Signal a command syntax error'
238 0235 1 GLOBAL ROUTINE NCPSSIG_CMDERR (CODE, TKN_CNT, TKN_PTR, STR_CNT, STR_PTR) =
239 0236 1
240 0237 1
241 0238 1  --
242 0239 1  FUNCTIONAL DESCRIPTION:
243 0240 1      A command error is signalled for printing. The signal name is
244 0241 1      code. The remainder of the arguments give the user context for his
245 0242 1      error. The context is cleaned up and passed on for printing.
246 0243 1
247 0244 1      If prompting is not active, we signal stop to avoid further error
248 0245 1      messages being printed. If prompting is active, we signal so that
249 0246 1      the prompt will allow the user to correct his mistake.
250 0247 1
251 0248 1  FORMAL PARAMETERS:
252 0249 1
253 0250 1      CODE          Value of status code to signal
254 0251 1      TKN_CNT       Value of size of token in error
255 0252 1      TKN_PTR       Address of token in error
256 0253 1      STR_CNT      Value of size of remaining part of command
257 0254 1      STR_PTR      Address of remaining part of command
258 0255 1
259 0256 1  IMPLICIT INPUTS:
260 0257 1
261 0258 1      ACT$GL_PMT_Q   True for prompting active
262 0259 1      NCP$_PRSCMD_DSC Descriptor of parsed command
263 0260 1
264 0261 1  IMPLICIT OUTPUTS:
265 0262 1
266 0263 1      NONE
267 0264 1
268 0265 1  ROUTINE VALUE:
269 0266 1  COMPLETION CODES:
270 0267 1
271 0268 1      Error condition signalled
272 0269 1
273 0270 1  SIDE EFFECTS:
274 0271 1
275 0272 1      NONE
276 0273 1
277 0274 1  --
278 0275 2  BEGIN
279 0276 2
280 0277 2  LITERAL
281 0278 2      WDO_SIZ = 30          ! Window size for error text
282 0279 2      :
283 0280 2
284 0281 2      LOCAL
285 0282 2      BFR_CNT,          ! Before counter for error
286 0283 2      BFR_PTR,          ! Before pointer for error
287 0284 2      AFT_CNT
288 0285 2      :
289 0286 2
290 0287 3  IF (
291 0288 4      ( (.TKN_PTR + .TKN_CNT) GEQA .STR_PTR )
292 0289 3      AND
293 0290 4      ( (.TKN_PTR + .TKN_CNT) LSSA (.STR_PTR + .STR_CNT) )

```


		14	AC		51	D1	00008		CMPL	R1, STR_PTR		
					1C	1F	0000C		BLSSU	1\$		
	50	14	AC	10	AC	C1	0000E		ADDL3	STR_CNT, STR_PTR, R0	0290	
			50		51	D1	00014		CMPL	R1, R0		
					11	1E	00017		BGEQU	1\$		
	50	14	AC	10	AC	C1	00019		ADDL3	STR_PTR, STR_PTR, R0	0294	
10	AC		50		51	C3	0001F		SUBL3	R1, R0, STR_CNT	0295	
			14		51	D0	00024		MOVL	R1, STR_PTR	0296	
					08	11	00028		BRB	2\$		
			0C	AC	14	AC	D0	0002A	1\$:	MOVL	STR_PTR, TKN_PTR	0300
					08	AC	D4	0002F		CLRL	TKN_CNT	0301
			50	00000000	00	D0	00032	2\$:	MOVL	NCP\$ PRSCMD_DSC+4, R0	0306	
	51	0C	AC		50	C3	00039		SUBL3	R0, TKN_PTR, BFR_CNT		
			1E		51	D1	0003E		CMPL	BFR_CNT, #30	0307	
					0A	1B	00041		BLEQU	3\$		
	52	0C	AC		1E	C3	00043		SUBL3	#30, TKN_PTR, BFR_PTR	0311	
			51		1E	D0	00048		MOVL	#30, BFR_CNT	0312	
					03	11	0004B		BRB	4\$		
			52		50	D0	0004D	3\$:	MOVL	R0, BFR_PTR	0315	
			50	10	AC	D0	00050	4\$:	MOVL	STR_PTR, AFT_CNT	0318	
			1E		50	D1	00054		CMPL	AFT_CNT, #30	0319	
					03	1B	00057		BLEQU	5\$		
			50		1E	D0	00059		MOVL	#30, AFT_CNT	0322	
			18	00000000G	00	E9	0005C	5\$:	BLBC	ACT\$GL_PRT_0, 6\$	0324	
					14	AC	DD	00063		PUSHL	STR_PTR	0330
					50	DD	00066		PUSHL	AFT_CNT		
			7E		AC	7D	00068		MOVQ	TKN_CNT, -(SP)	0329	
					06	BB	0006C		PUSHR	#^MZR1,R2>	0328	
					06	DD	0006E		PUSHL	#6	0327	
					AC	DD	00070		PUSHL	CODE		
	00000000G		00		08	FB	00073		CALLS	#8, LIB\$SIGNAL		
					04	04	0007A		RET			
					14	AC	DD	0007B	6\$:	PUSHL	STR_PTR	0337
					50	DD	0007E		PUSHL	AFT_CNT		
			7E		AC	7D	00080		MOVQ	TKN_CNT, -(SP)	0336	
					06	BB	00084		PUSHR	#^MZR1,R2>	0335	
					06	DD	00086		PUSHL	#6	0334	
					AC	DD	00088		PUSHL	CODE		
	00000000G		00		08	FB	0008B		CALLS	#8, LIB\$STOP		
					04	04	00092		RET		0340	

; Routine Size: 147 bytes, Routine Base: \$CODE\$ + 0090

```

345 0341 1 %SBTTL 'ACT$INV COMMAND Action routine for invalid command'
346 0342 1 GLOBAL ROUTINE ACT$INV COMMAND (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
347 0343 1 CHR, NOM, PARAM) =
348 0344 1
349 0345 1 +-
350 0346 1 FUNCTIONAL DESCRIPTION:
351 0347 1
352 0348 1 Action routine to signal an invalid command. Signal is either
353 0349 1 for invalid command or ambiguous command if the AMBIG bit is
354 0350 1 set in the parse options.
355 0351 1
356 0352 1 FORMAL PARAMETERS:
357 0353 1
358 0354 1 Parse state table
359 0355 1 OPT Value of the parse options
360 0356 1 STRCNT Size of the remainder of the command line
361 0357 1 STRPTR Address of the remainder of the command line
362 0358 1 TKNCNT Size of the token in error
363 0359 1 TKNPTR Address of the token in error
364 0360 1
365 0361 1 IMPLICIT INPUTS:
366 0362 1
367 0363 1 NONE
368 0364 1
369 0365 1 IMPLICIT OUTPUTS:
370 0366 1
371 0367 1 NONE
372 0368 1
373 0369 1 ROUTINE VALUE:
374 0370 1 COMPLETION CODES:
375 0371 1
376 0372 1 NONE
377 0373 1
378 0374 1 SIDE EFFECTS:
379 0375 1
380 0376 1 NONE
381 0377 1
382 0378 1 --
383 0379 1
384 0380 2 BEGIN
385 0381 2
386 0382 2 NCP$SIG_CMDERR ! Signal the error
387 0383 4 ( ( IF (.OPT AND TPASH_AMBIG) ! Use the appropriate code
388 0384 3 NEQ 0
389 0385 3 THEN NCP$_AMBCMD ! based on the ambiguous option bit
390 0386 3 ELSE NCP$_INVCMD
391 0387 2 ) OR ST$K SEVERE, ! Make severe to stop
392 0388 2 .TKNCNT, .TKNPTR, ! the token in error
393 0389 2 .STRCNT, .STRPTR ! the remainder of the command line
394 0390 2 )
395 0391 2
396 0392 1 END; ! End of routine

```

```

0000 00000
7E      08 AC 7D 00002
7E      10 AC 7D 00006
09      6C 30 E1 0000A
      50 00000000G 8F D0 0000E
      07 11 00015
      50 00000000G 8F D0 00017 1$:
7E      50      04 C9 0001E 2$:
      FF46 CF 05 FB 00022
      04 00027

```

```

.ENTRY ACT$INV_COMMAND, Save nothing
MOVQ STRCNT, -(SP)
MOVQ TKNCNT, -(SP)
BBC #48, OPT, 1$
MOVL #NCP$_AMB_CMD, R0
BRB 2$
MOVL #NCP$_INV_CMD, R0
BISL3 #4, R0, -(SP)
CALLS #5, NCP$SIG_CMDERR
RET

```

```

: 0342
: 0389
: 0388
: 0384
: 0383
:
: 0387
: 0392

```

; Routine Size: 40 bytes, Routine Base: \$CODE\$ + 0123

.....

```

: 398 0393 1 %SBTTL 'ACT$TMPSTR Save a temporary string'
: 399 0394 1 GLOBAL ROUTINE ACT$TMPSTR (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 400 0395 1 CHR, NUM, PARAM) =
: 401 0396 1
: 402 0397 1
: 403 0398 1 +-
: 404 0399 1 FUNCTIONAL DESCRIPTION:
: 405 0400 1 Action routine to save a temporary string in a descriptor
: 406 0401 1
: 407 0402 1 FORMAL PARAMETERS:
: 408 0403 1
: 409 0404 1 Parse state table
: 410 0405 1 TKNCNT Count of string
: 411 0406 1 TKNPTR Address of string
: 412 0407 1 PARAM Address of string descriptor to return
: 413 0408 1
: 414 0409 1 IMPLICIT INPUTS:
: 415 0410 1
: 416 0411 1 NONE
: 417 0412 1
: 418 0413 1 IMPLICIT OUTPUTS:
: 419 0414 1
: 420 0415 1 NONE
: 421 0416 1
: 422 0417 1 ROUTINE VALUE:
: 423 0418 1 COMPLETION CODES:
: 424 0419 1
: 425 0420 1 SUCCESS
: 426 0421 1
: 427 0422 1 SIDE EFFECTS:
: 428 0423 1
: 429 0424 1 NONE
: 430 0425 1
: 431 0426 1 --
: 432 0427 1
: 433 0428 2 BEGIN
: 434 0429 2
: 435 0430 2 MAP
: 436 0431 2 PARAM : REF VECTOR [2] ! Address of string descriptor
: 437 0432 2
: 438 0433 2 PARAM [0] = .TKNCNT; ! Fill in the string descriptor
: 439 0434 2 PARAM [1] = .TKNPTR;
: 440 0435 2 RETURN SUCCESS
: 441 0436 1 END;

```

```

          50      20      AC  D0 00002      .ENTRY ACT$TMPSTR, Save nothing      : 0394
          60      10      AC  7D 00006      MOVL  PARAM, R0      : 0433
          50      01      D0 0000A      MOVQ  TKNCNT, (R0)      :
          04      0000D      MOVL  #1, R0      : 0435
                                     RET      : 0436

```

; Routine Size: 14 bytes, Routine Base: \$CODES + 014B

NCPPRSACT
V04-000

Parse Data and Action Routines
ACT\$TMPSTR Save a temporary string

D 10
15-Sep-1984 23:51:04
14-Sep-1984 12:48:15

VAX-11 Bliss-32 V4.0-742
[NCP.SRC]NCPPRSACT.B32;1

Page 15
(6)

NCP
V04

.....

```

: 443 0437 1 %SBTTL 'ACT$BLNK SIG Blanks are significant'
: 444 0438 1 GLOBAL ROUTINE ACT$BLNK_SIG (OPTIONS) = !
: 445 0439 1
: 446 0440 1 +-
: 447 0441 1 FUNCTIONAL DESCRIPTION:
: 448 0442 1
: 449 0443 1     Set parse options so blanks (spaces, tabs) are significant
: 450 0444 1
: 451 0445 1 FORMAL PARAMETERS:
: 452 0446 1
: 453 0447 1     Parse state table
: 454 0448 1     OPT           Options longword
: 455 0449 1
: 456 0450 1 IMPLICIT INPUTS:
: 457 0451 1
: 458 0452 1     NONE
: 459 0453 1
: 460 0454 1 IMPLICIT OUTPUTS:
: 461 0455 1
: 462 0456 1     NONE
: 463 0457 1
: 464 0458 1 ROUTINE VALUE:
: 465 0459 1 COMPLETION CODES:
: 466 0460 1
: 467 0461 1     Success
: 468 0462 1
: 469 0463 1 SIDE EFFECTS:
: 470 0464 1
: 471 0465 1     NONE
: 472 0466 1
: 473 0467 1 --
: 474 0468 1
: 475 0469 2 BEGIN
: 476 0470 2
: 477 0471 2           ! Blanks are significant
: 478 0472 2     OPTIONS = .OPTIONS OR TPASM_BLANKS ;
: 479 0473 2     RETURN SUCCESS
: 480 0474 2
: 481 0475 1     END;

```

```

          04 AC          01 0000 0000          .ENTRY ACT$BLNK_SIG, Save nothing      : 0438
          50          01 88 00002          BISB2 #1, OPTIONS                          : 0472
          01 D0 00006          MOVL #1, R0                                       : 0473
          04 00009          RET                                                    : 0475

```

: Routine Size: 10 bytes, Routine Base: \$CODE\$ + 0159

```

: 483 0476 1 %SBTTL 'ACT$BLNK_NSIG Blanks are not significant'
: 484 0477 1 GLOBAL ROUTINE ACT$BLNK_NSIG (OPTIONS) =
: 485 0478 1
: 486 0479 1 :++
: 487 0480 1 FUNCTIONAL DESCRIPTION:
: 488 0481 1
: 489 0482 1 Parse options are set so blanks (spaces, tabs) are not significant
: 490 0483 1
: 491 0484 1 FORMAL PARAMETERS:
: 492 0485 1
: 493 0486 1 Parse state table
: 494 0487 1 OPT Options longword
: 495 0488 1
: 496 0489 1 IMPLICIT INPUTS:
: 497 0490 1
: 498 0491 1 NONE
: 499 0492 1
: 500 0493 1 IMPLICIT OUTPUTS:
: 501 0494 1
: 502 0495 1 NONE
: 503 0496 1
: 504 0497 1 ROUTINE VALUE:
: 505 0498 1 COMPLETION CODES:
: 506 0499 1
: 507 0500 1 Success
: 508 0501 1
: 509 0502 1 SIDE EFFECTS:
: 510 0503 1
: 511 0504 1 NONE
: 512 0505 1
: 513 0506 1 :--
: 514 0507 1
: 515 0508 2 BEGIN
: 516 0509 2
: 517 0510 2 OPTIONS = .OPTIONS AND (NOT TPASM_BLANKS) ; ! Blanks not significant
: 518 0511 2 RETURN SUCCESS
: 519 0512 2
: 520 0513 1 END;

```

```

          04 AC          01 0000 0000          .ENTRY ACT$BLNK_NSIG, Save nothing          : 0477
          50          01 8A 0002          BICB2 #1, OPTIONS          : 0510
          50          01 D0 0006          MOVL #1, R0          : 0511
          50          04 0009          RET          : 0513

```

: Routine Size: 10 bytes, Routine Base: \$CODE\$ + 0163

```

522 0514 1 %SBTTL 'ACT$ZAPTMPDSC Zero a temporary descriptor'
523 0515 1 GLOBAL ROUTINE ACT$ZAPTMPDSC (OPT, STRCNT, STRPTR, TKNCTR,TKNPTR,
524 0516 1 CHR, NUM, PARAM) = !
525 0517 1
526 0518 1 ++
527 0519 1 FUNCTIONAL DESCRIPTION:
528 0520 1
529 0521 1 Zero a list of descriptors for temporary strings
530 0522 1
531 0523 1 FORMAL PARAMETERS:
532 0524 1
533 0525 1 Parse state table
534 0526 1 PARAM Address of PLIT for addresses of descriptor
535 0527 1
536 0528 1 IMPLICIT INPUTS:
537 0529 1
538 0530 1 NONE
539 0531 1
540 0532 1 IMPLICIT OUTPUTS:
541 0533 1
542 0534 1 NONE
543 0535 1
544 0536 1 ROUTINE VALUE:
545 0537 1 COMPLETION CODES:
546 0538 1
547 0539 1 SUCCESS
548 0540 1
549 0541 1 SIDE EFFECTS:
550 0542 1
551 0543 1 NONE
552 0544 1
553 0545 1 --
554 0546 1
555 0547 1 BEGIN
556 0548 1
557 0549 1 MAP
558 0550 1 PARAM : REF VECTOR [10] ! Expect a rather long list
559 0551 1 :
560 0552 1
561 0553 1 INCRU IDX FROM 0 ! Scan the PLIT for addresses
562 0554 1 TO .PARAM [-1] - 1 ! PLIT count
563 0555 1 DO
564 0556 1 .PARAM [.IDX] = 0 ! Zap the count for the descriptor
565 0557 1
566 0558 1 RETURN SUCCESS
567 0559 1
568 0560 1 END:

```

```

53 FC A2 20 AC D0 00002
01 C3 00006
50 D4 0000B
08 11 0000D

```

```

.ENTRY ACT$ZAPTMPDSC, Save R2,R3
MOVL PARAM, R2
SUBL3 #1, -4(R2), R3
CLRL IDX
BRB 2$

```

```

: 0515
: 0554
: 0553
:

```



```

570 0561 1 %SBTTL 'ACT$PRMPT Action routine to prompt'
571 0562 1 GLOBAL ROUTINE ACT$PRMPT (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
572 0563 1     CHR, NUM, PARAM) =
573 0564 1
574 0565 1  !++
575 0566 1  FUNCTIONAL DESCRIPTION:
576 0567 1
577 0568 1     Prompt for the remaining portion of a command if input is
578 0569 1     interactive.
579 0570 1
580 0571 1  FORMAL PARAMETERS:
581 0572 1
582 0573 1     Parse state table
583 0574 1     PARAM      Address of descriptor of prompt string
584 0575 1
585 0576 1  IMPLICIT INPUTS:
586 0577 1
587 0578 1     NCP$CMD_BUF_DSC Descriptor of command line buffer
588 0579 1
589 0580 1  IMPLICIT OUTPUTS:
590 0581 1
591 0582 1     NCP$_PRSCMD_DSC Descriptor of new command line portion
592 0583 1
593 0584 1  ROUTINE VALUE:
594 0585 1  COMPLETION CODES:
595 0586 1
596 0587 1     Success or error signaled
597 0588 1
598 0589 1  SIDE EFFECTS:
599 0590 1
600 0591 1     NONE
601 0592 1
602 0593 1  --
603 0594 1
604 0595 2  BEGIN
605 0596 2
606 0597 2  MAP
607 0598 2     PARAM : REF VECTOR      ! Descriptor of prompt string
608 0599 2     ;
609 0600 2
610 0601 2  LOCAL
611 0602 2     STATUS
612 0603 2     ;
613 0604 2
614 0605 2  IF NOT NCP$CMD_TERM_Q ( )      ! If we are not on a terminal,
615 0606 2  THEN
616 0607 2     SIGNAL_STOP (NCP$_NOTDONE) ! Dump off with a special error
617 0608 2     ;
618 0609 2
619 0610 2  STATUS = NCP$READ_CMD          ! Read another command portion
620 0611 2     (NCP$CMDBUF_DSC,
621 0612 2     .PARAM,
622 0613 2     NCP$_PRSCMD_DSC
623 0614 2     )
624 0615 2     ;
625 0616 2  IF NOT .STATUS                ! Any error dumps us off
626 0617 2  THEN

```

```

: 627      0618      3      BEGIN
: 628      0619      3      IF .STATUS EQL RMSS EOF
: 629      0620      3      THEN SIGNAL_STOP (NCP$ _CMDCAN) ! Command canceled if EOF
: 630      0621      3      ! Report any other error too
: 631      0622      3      ELSE SIGNAL_STOP (NCP$ _CMDERR, 0, .STATUS)
: 632      0623      3      END
: 633      0624      3      ;
: 634      0625      3      STRPTR = .NCP$ _PRSCMD _DSC [1];      ! Set parse state to the portion
: 635      0626      3      STRCNT = .NCP$ _PRSCMD _DSC [0];
: 636      0627      3      RETURN SUCCESS      ! And continue with parse
: 637      0628      3
: 638      0629      3
: 639      0630      3
: 640      0631      3      END;

```

		000C 00000	.ENIRY	ACT\$PRMPT, Save R2,R3	: 0562		
	53	00000000'	00	9E 00002	MOVAB	NCP\$ PRSCMD _DSC, R3	
	52	00000000G	00	9E 00009	MOVAB	LIB\$STOP, R2	
00000000G	00		00	FB 00010	CALLS	#0, NCP\$CMD_TERM_q	: 0605
	09		50	E8 00017	BLBS	RO, 1\$	
	62	00000000G	8F	DD 0001A	PUSHL	#NCP\$ NOTDONE	: 0607
			01	FB 00020	CALLS	#1, LIB\$STOP	
			53	DD 00023	PUSHL	R3	: 0611
		20	AC	DD 00025	PUSHL	PARAM	: 0612
		00000000G	00	9F 00028	PUSHAB	NCP\$ CMDBUF _DSC	: 0611
00000000G	00		03	FB 0002E	CALLS	#3, NCP\$READ_CMD	
	21		50	E8 00035	BLBS	STATUS, 3\$: 0616
0001827A	8F		50	D1 00038	CMP	STATUS, #98938	: 0619
			0B	12 0003F	BNEQ	2\$	
		00000000G	8F	DD 00041	PUSHL	#NCP\$ CMDCAN	: 0620
	62		01	FB 00047	CALLS	#1, LIB\$STOP	
			0D	11 0004A	BRB	3\$	
			50	DD 0004C	PUSHL	STATUS	: 0622
			7E	D4 0004E	CLRL	-(SP)	
		00000000G	8F	DD 00050	PUSHL	#NCP\$ CMDERR	
	62		03	FB 00056	CALLS	#3, LIB\$STOP	
08	AC		63	7D 00059	MOVQ	NCP\$ PRSCMD _DSC, STRCNT	: 0627
	50		01	D0 0005D	MOVL	#1, RO	: 0629
			04	00060	RET		: 0631

; Routine Size: 97 bytes, Routine Base: \$CODE\$ + 018D

```

642 0632 1 %SBTTL 'ACT$NUM_RNG Check numeric ranges'
643 0633 1 GLOBAL ROUTINE ACT$NUM_RNG (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
644 0634 1     CHR, NOM, PARAM) =
645 0635 1
646 0636 1
647 0637 1  +-+
648 0638 1  FUNCTIONAL DESCRIPTION:
649 0639 1      Action routine to check numeric range of numeric parameter
650 0640 1
651 0641 1  FORMAL PARAMETERS:
652 0642 1
653 0643 1      Parse state table
654 0644 1      NUM      Value of the numeric token
655 0645 1      PARAM     Address of range as two long words, low first
656 0646 1
657 0647 1  IMPLICIT INPUTS:
658 0648 1
659 0649 1      NONE
660 0650 1
661 0651 1  IMPLICIT OUTPUTS:
662 0652 1
663 0653 1      NONE
664 0654 1
665 0655 1  ROUTINE VALUE:
666 0656 1  COMPLETION CODES:
667 0657 1
668 0658 1      Success or error signal
669 0659 1
670 0660 1  SIDE EFFECTS:
671 0661 1
672 0662 1      NONE
673 0663 1
674 0664 1  --
675 0665 1
676 0666 2  BEGIN
677 0667 2
678 0668 2  MAP
679 0669 2  PARAM : REF VECTOR [2]      ! Address of UPLIT (low, high)
680 0670 2  ;
681 0671 2
682 0672 2  IF .NUM GEQU .PARAM [0] AND    ! If inbounds return success
683 0673 2  .NUM LEQU .PARAM [1]
684 0674 2  THEN
685 0675 2  RETURN SUCCESS
686 0676 2  ELSE
687 0677 2  BEGIN
688 0678 2  NCP$SIG CMDERR                ! Signal parameter out of range
689 0679 2  (NCP$ PRMRNG,
690 0680 2  .TKNCNT, .TKNPTR,
691 0681 2  .STRCNT, .STRPTR
692 0682 2  )
693 0683 2  ;
694 0684 2  RETURN FAILURE                ! Fail transition in parse table
695 0685 2  END;
696 0686 1  END;

```


			0000	00000		.ENTRY	ACT\$NUM_RNG, Save nothing	:	0633
	50	20	AC	D0	00002	MOVL	PARAM, R0	:	0672
	60	1C	AC	D1	00006	CMPL	NUM, (R0)	:	
			0B	1F	0000A	BLSSU	1\$:	
04	A0	1C	AC	D1	0000C	CMPL	NUM, 4(R0)	:	0673
			04	1A	00011	BGTRU	1\$:	
	50		01	D0	00013	MOVL	#1, R0	:	0677
				04	00016	RET		:	
	7E	08	AC	7D	00017	MOVQ	STRCNT, -(SP)	:	0681
	7E	10	AC	7D	0001B	MOVQ	TKNCNT, -(SP)	:	0680
		00000000G	8F	DD	0001F	PUSHL	#NCP\$ PRMRNG	:	0679
FE78	CF		05	FB	00025	CALLS	#5, NCP\$SIG_CMDERR	:	
			50	D4	0002A	CLRL	R0	:	0684
				04	0002C	RET		:	0686

; Routine Size: 45 bytes, Routine Base: \$CODE\$ + 01EE

```
698 0687 1 %SBTTL 'ACT$NUM RNGSAV Check numeric ranges and store value in vector'  
699 0688 1 GLOBAL ROUTINE ACT$NUM RNGSAV (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,  
700 0689 1 CHR, NOM, PARAM) = !  
701 0690 1  
702 0691 1 +-  
703 0692 1 FUNCTIONAL DESCRIPTION:  
704 0693 1  
705 0694 1 Action routine to check numeric range of numeric parameter  
706 0695 1 and store value in vector.  
707 0696 1  
708 0697 1 FORMAL PARAMETERS:  
709 0698 1  
710 0699 1 Parse state table  
711 0700 1 NUM Value of the numeric token  
712 0701 1 PARAM Address of range as two long words, low first  
713 0702 1  
714 0703 1 IMPLICIT INPUTS:  
715 0704 1  
716 0705 1 NONE  
717 0706 1  
718 0707 1 IMPLICIT OUTPUTS:  
719 0708 1  
720 0709 1 NONE  
721 0710 1  
722 0711 1 ROUTINE VALUE:  
723 0712 1 COMPLETION CODES:  
724 0713 1  
725 0714 1 Success or error signal  
726 0715 1  
727 0716 1 SIDE EFFECTS:  
728 0717 1  
729 0718 1 NONE  
730 0719 1  
731 0720 1 --  
732 0721 1  
733 0722 2 BEGIN  
734 0723 2  
735 0724 2 MAP  
736 0725 2 PARAM : REF VECTOR [2] ! Address of UPLIT (low, high)  
737 0726 2 ;  
738 0727 2  
739 0728 2 IF .NUM GEQU .PARAM [0] AND ! If inbounds then store  
740 0729 2 .NUM LEQU .PARAM [1]  
741 0730 2 THEN  
742 0731 3 BEGIN ! Store the value  
743 0732 3 IF .ACT$GA_RNGLST [0] LSS ACT$C_RNGLSTMAX  
744 0733 3 THEN  
745 0734 4 BEGIN  
746 0735 4 ACT$GA_RNGLST [0] = .ACT$GA_RNGLST [0] + 1;  
747 0736 4 ACT$GA_RNGLST [.ACT$GA_RNGLST [0]] = .NUM;  
748 0737 4 END  
749 0738 3 ELSE  
750 0739 4 BEGIN  
751 0740 4  
752 0741 4 ! If the vector is full then complain  
753 0742 4  
754 0743 4 NCP$SIG_CMDERR ! Signal too many ranges
```

```

: 755      0744 4      (NCP$ FIELDLIM,
: 756      0745 4      .TKNCNT, .TKNPTR,
: 757      0746 4      .STRCNT, .STRPTR
: 758      0747 4      );
: 759      0748 4      RETURN FAILURE;      ! Fail transition in parse table
: 760      0749 3      END;
: 761      0750 3      ELSE
: 762      0751 3      BEGIN
: 763      0752 3      NCP$SIG CMDERR      ! Signal parameter out of range
: 764      0753 3      (NCP$ PRMRNG,
: 765      0754 3      .TKNCNT, .TKNPTR,
: 766      0755 3      .STRCNT, .STRPTR
: 767      0756 3      );
: 768      0757 3      ;
: 769      0758 3      RETURN FAILURE;      ! Fail transition in parse table
: 770      0759 3      END;
: 771      0760 3
: 772      0761 2
: 773      0762 2      RETURN SUCCESS;
: 774      0763 1      END;

```

```

: 0004 00000 .ENTRY ACT$NUM_RNGSAV, Save R2      : 0688
: 52 00000000' 00 9E 00002 MOVAB ACT$GA_RNGLST, R2
: 50 20 AC D0 00009 MOVL PARAM, R0      : 0728
: 60 1C AC D1 0000D CMPL NUM, (R0)
: 28 1F 00011 BLSSU 2$
: 04 A0 1C AC D1 00013 CMPL NUM, 4(R0)      : 0729
: 21 1A 00018 BGTRU 2$
: 20 62 B1 0001A CMPW ACT$GA_RNGLST, #32      : 0732
: 0C 1E 0001D BGEQU 1$
: 62 B6 0001F INCW ACT$GA_RNGLST      : 0735
: 50 62 3C 00021 MOVZWL ACT$GA_RNGLST, R0      : 0736
: 6240 1C AC B0 00024 MOVW NUM, ACT$GA_RNGLST[R0]
: 25 11 00029 BRB 4$      : 0732
: 7E 08 AC 7D 0002B 1$: MOVQ STRCNT, -(SP)      : 0746
: 7E 10 AC 7D 0002F 1$: MOVQ TKNCNT, -(SP)      : 0745
: 00000000G 8F DD 00033 PUSHL #NCP$_FIELDLIM      : 0744
: 0E 11 00039 BRB 3$
: 7E 08 AC 7D 0003B 2$: MOVQ STRCNT, -(SP)      : 0756
: 7E 10 AC 7D 0003F 2$: MOVQ TKNCNT, -(SP)      : 0755
: 00000000G 8F DD 00043 PUSHL #NCP$ PRMRNG      : 0754
: FE27 CF 05 FB 00049 3$: CALLS #5, NCP$SIG_CMDERR
: 04 11 0004E BRB 5$      : 0759
: 50 01 D0 00050 4$: MOVL #1, R0      : 0762
: 04 00053 RET
: 50 D4 00054 5$: CLRL R0
: 04 00056 RET      : 0763

```

; Routine Size: 87 bytes, Routine Base: \$CODE\$ + 021B

```

: 776 0764 1 %SBTTL 'ACT$NUM_SAV Store value in vector'
: 777 0765 1 GLOBAL ROUTINE ACT$NUM_SAV (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 778 0766 1           CHR, NOM) =
: 779 0767 1
: 780 0768 1 ++
: 781 0769 1 FUNCTIONAL DESCRIPTION:
: 782 0770 1
: 783 0771 1     Action routine to store value in vector.
: 784 0772 1
: 785 0773 1 FORMAL PARAMETERS:
: 786 0774 1
: 787 0775 1     Parse state table
: 788 0776 1     NUM           Value of the numeric token
: 789 0777 1
: 790 0778 1 IMPLICIT INPUTS:
: 791 0779 1
: 792 0780 1     NONE
: 793 0781 1
: 794 0782 1 IMPLICIT OUTPUTS:
: 795 0783 1
: 796 0784 1     NONE
: 797 0785 1
: 798 0786 1 ROUTINE VALUE:
: 799 0787 1 COMPLETION CODES:
: 800 0788 1
: 801 0789 1     Success or error signal
: 802 0790 1
: 803 0791 1 SIDE EFFECTS:
: 804 0792 1
: 805 0793 1     NONE
: 806 0794 1
: 807 0795 1 --
: 808 0796 1
: 809 0797 2 BEGIN
: 810 0798 2 IF .ACT$GA_RNGLST [0] LSS ACT$C_RNGLSTMAX
: 811 0799 2 THEN
: 812 0800 3     BEGIN
: 813 0801 3         ! Store the value
: 814 0802 3         ACT$GA_RNGLST [0] = .ACT$GA_RNGLST [0] + 1;
: 815 0803 3         ACT$GA_RNGLST [.ACT$GA_RNGLST [0] ] = .NUM;
: 816 0804 3     END
: 817 0805 3 ELSE
: 818 0806 3     BEGIN
: 819 0807 3         ! If the vector is full then complain
: 820 0808 3         NCP$SIG CMDERR
: 821 0809 3         ! Signal too many ranges
: 822 0810 3         (NCP$ FIELDLIM,
: 823 0811 3         .TKNCNT, .TKNPTR,
: 824 0812 3         .STRCNT, .STRPTR
: 825 0813 3         );
: 826 0814 3     RETURN FAILURE;
: 827 0815 3     ! Fail transition in parse table
: 828 0816 3     END;
: 829 0817 2 RETURN SUCCESS;
: 830 0818 1 END;

```

; R


```

: 832 0819 1 %SBTTL 'ACT$STR_LEN Check string length'
: 833 0820 1 GLOBAL ROUTINE ACT$STR_LEN (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 834 0821 1 CHR, NOM, PARAM) =
: 835 0822 1
: 836 0823 1
: 837 0824 1 ++
: 838 0825 1 FUNCTIONAL DESCRIPTION:
: 839 0826 1 Action routine to check length of a string token.
: 840 0827 1 Length is checked for strings with trailing spaces and
: 841 0828 1 quoted strings. Trailing spaces and tabs are removed and
: 842 0829 1 quoted strings which must begin with a " are counted with
: 843 0830 1 the quotes parsed correctly.
: 844 0831 1
: 845 0832 1 FORMAL PARAMETERS:
: 846 0833 1
: 847 0834 1 Parse state table
: 848 0835 1 TKNCNT Length of the string token
: 849 0836 1 PARAM Value of the maximum length of token
: 850 0837 1
: 851 0838 1 IMPLICIT INPUTS:
: 852 0839 1
: 853 0840 1 NONE
: 854 0841 1
: 855 0842 1 IMPLICIT OUTPUTS:
: 856 0843 1
: 857 0844 1 NONE
: 858 0845 1
: 859 0846 1 ROUTINE VALUE:
: 860 0847 1 COMPLETION CODES:
: 861 0848 1
: 862 0849 1 Success or error signal
: 863 0850 1
: 864 0851 1 SIDE EFFECTS:
: 865 0852 1
: 866 0853 1 NONE
: 867 0854 1
: 868 0855 1 --
: 869 0856 1
: 870 0857 2 BEGIN
: 871 0858 2
: 872 0859 2 LOCAL
: 873 0860 2 PTR, ! Point into token
: 874 0861 2 PEND, ! End of token
: 875 0862 2 CTR ! Size of string
: 876 0863 2 ;
: 877 0864 2
: 878 0865 2 IF CHRCHAR (.TKNPTR) EQL '"' ! Quoted string?
: 879 0866 2 THEN
: 880 0867 3 BEGIN
: 881 0868 3 CTR = 0; ! Setup counters and pointers
: 882 0869 3 PTR = .TKNPTR + 1;
: 883 0870 3 PEND = .TKNPTR + .TKNCNT;
: 884 0871 3 WHILE .PTR LSS .PEND ! Scan string
: 885 0872 3 DO
: 886 0873 4 BEGIN
: 887 0874 4 IF CHRCHAR_A (PTR) EQL '"' ! Quote inside?
: 888 0875 4 THEN

```

```

889 0876 5 BEGIN
890 0877 5 IF CH$RCHAR (.PTR) EQL '"'
891 0878 5 AND
892 0879 5 .PTR LSS .PEND
893 0880 5 THEN PTR = .PTR + 1 ! Count one quote for two
894 0881 5 ELSE EXITLOOP ! Single quote ends it
895 0882 5 END
896 0883 5 ;
897 0884 5 CTR = .CTR + 1
898 0885 5 END
899 0886 5 END
900 0887 5 ELSE
901 0888 5 BEGIN
902 0889 5 PTR = .TKNPTR + .TKNCNT - 1; ! Strip trailing spaces from token
903 0890 5 WHILE PTR GTRU .TKNPTR
904 0891 5 AND
905 0892 5 (
906 0893 5 CH$RCHAR (.PTR) EQL ' ' ! Space
907 0894 5 OR
908 0895 5 CH$RCHAR (.PTR) EQL 9 ! Tab
909 0896 5 )
910 0897 5 DO
911 0898 5 PTR = .PTR - 1
912 0899 5 ;
913 0900 5 CTR = (.PTR + 1) - .TKNPTR ! Compute real size of token
914 0901 5 END
915 0902 5 ;
916 0903 5 ;
917 0904 5 IF .CTR LEQU .PARAM ! Check size of token
918 0905 5 THEN
919 0906 5 RETURN SUCCESS
920 0907 5 ELSE
921 0908 5 BEGIN
922 0909 5 NCP$SIG CMDERR ! Signal to print error message
923 0910 5 (NCP$ PRMLEN,
924 0911 5 .TKNCNT, .TKNPTR,
925 0912 5 .STRCNT, .STRPTR
926 0913 5 )
927 0914 5 ;
928 0915 5 RETURN FAILURE ! Fail transition in parse table
929 0916 5 END
930 0917 5
931 0918 1 END:

```

			000C	00000	.ENTRY	ACT\$STR_LEN, Save R2,R3	: 0820
	SE		04	C2 00002	SUBL2	#4, SP	: 0865
	52	14	AC	00 00005	MOVL	TKNPTR, R2	: 0870
51	52	10	AC	C1 00009	ADDL3	TKNCNT, R2, R1	: 0865
	22		62	91 0000E	CMPB	(R2), #34	: 0868
			27	12 00011	BNEQ	3\$: 0869
			50	D4 00013	CLRL	CTR	: 0871
	6E	01	A2	9E 00015	MOVAB	1(R2), PTR	: 0871
	51		6E	D1 00019 1\$:	CMPL	PTR, PEND	: 0871

			40	18	0C01C	BGEQ	7\$		
	53	00	BE	9A	0001E	MOVZBL	@PTR, R3		0874
			6E	D6	00022	INCL	PTR		
	22		53	91	00024	CMPB	R3, #34		
			0D	12	00027	BNEQ	2\$		
	22	00	BE	91	00029	CMPB	@PTR, #34		0877
			2F	12	0002D	BNEQ	7\$		
	51		6E	D1	0002F	CMPL	PTR, PEND		0879
			2A	18	00032	BGEQ	7\$		
			6E	D6	00034	INCL	PTR		0880
			50	D6	00036	INCL	CTR		0884
			DF	11	00038	BRB	1\$		
	6E	FF	A1	9E	0003A	MOVAB	-1(R1), PTR		0889
	51		6E	9E	0003E	MOVAB	PTR, R1		0890
	51		52	D1	00041	CMPL	R2, R1		
			10	1E	00044	BGEQU	6\$		
	20	00	BE	91	00046	CMPB	@PTR, #32		0893
			06	13	0004A	BEQL	5\$		
	09	00	BE	91	0004C	CMPB	@PTR, #9		0895
			04	12	00050	BNEQ	6\$		
			6E	D7	00052	DECL	PTR		0898
			E8	11	00054	BRB	4\$		
51	6E		52	C3	00056	SUBL3	R2, PTR, R1		0900
	50	01	A1	9E	0005A	MOVAB	1(R1), CTR		
20	AC		50	D1	0005E	CMPL	CTR, PARAM		0904
			04	1A	00062	BGTRU	8\$		
	50		01	D0	00064	MOVL	#1, R0		0908
				04	00067	RET			
	7E	08	AC	7D	00068	MOVQ	STRCNT, -(SP)		0912
			52	DD	0006C	PUSHL	R2		0911
		10	AC	DD	0006E	PUSHL	TKCNT		
		00000000G	8F	DD	00071	PUSHL	#NCP\$ PRMLEN		0910
FD6C	CF		05	FB	00077	CALLS	#5, NCP\$SIG_CMDERR		
			50	D4	0007C	CLRL	R0		0915
				04	0007E	RET			0918

; Routine Size: 127 bytes, Routine Base: \$CODE\$ + 02AB


```

933 0919 1 ZSBTTL 'ACT$NXT_STATE Set next state table for parse'
934 0920 1 GLOBAL ROUTINE ACT$NXT_STATE (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
935 0921 1 CHR, NOM, PARAM) = !
936 0922 1
937 0923 1 !+
938 0924 1 FUNCTIONAL DESCRIPTION:
939 0925 1
940 0926 1 Setup pointers to skip to another state table to finish parse
941 0927 1
942 0928 1 FORMAL PARAMETERS:
943 0929 1
944 0930 1 Parse state table
945 0931 1 PARAM Address of address pair - state_table, key_table
946 0932 1
947 0933 1 IMPLICIT INPUTS:
948 0934 1
949 0935 1 NONE
950 0936 1
951 0937 1 IMPLICIT OUTPUTS:
952 0938 1
953 0939 1 NONE
954 0940 1
955 0941 1 ROUTINE VALUE:
956 0942 1 COMPLETION CODES:
957 0943 1
958 0944 1 Success
959 0945 1
960 0946 1 SIDE EFFECTS:
961 0947 1
962 0948 1 NONE
963 0949 1
964 0950 1 --
965 0951 1
966 0952 2 BEGIN
967 0953 2
968 0954 2 MAP
969 0955 2 PARAM : REF VECTOR
970 0956 2 ;
971 0957 2
972 0958 2 NCP$_NXT_STATE [0] = .PARAM [0]; ! State table address
973 0959 2 NCP$_NXT_STATE [1] = .PARAM [1]; ! Key table address
974 0960 2 RETURN SUCCESS
975 0961 2
976 0962 1 END;

```

```

00000000' 50      20      AC  D0 00002      .ENTRY ACT$NXT_STATE, Save nothing      : 0920
00000000' 00      60      7D 00006      MOVL  PARAM, R0                          : 0958
00000000' 50      01      D0 0000D      MOVL  (R0), NCP$_NXT_STATE                : 0960
00000000' 04      04      00010      MOVL  #1, R0                              : 0962
00000000' 04      04      00010      RET

```

; Routine Size: 17 bytes, Routine Base: \$CODE\$ + 0327

NCPPRSACT
V04-000

Parse Data and Action Routines
ACT\$NXT_STATE Set next state table for parse

H 11
15-Sep-1984 23:51:04
14-Sep-1984 12:48:15

VAX-11 Bliss-32 V4.0-742
[NCP.SRC]NCPPRSACT.B32;1

Page 32
(15)

NCP
V04

: R

```

: 978 0963 1 %SBTTL 'ACT$WRI_STR Write a string to the output device'
: 979 0964 1 GLOBAL ROUTINE ACT$WRI_STR (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 980 0965 1 CHR, NOM, PARAM) =
: 981 0966 1
: 982 0967 1 !++
: 983 0968 1 ! FUNCTIONAL DESCRIPTION:
: 984 0969 1 !
: 985 0970 1 ! Write a string to SYSS$OUTPUT
: 986 0971 1 !
: 987 0972 1 ! FORMAL PARAMETERS:
: 988 0973 1 !
: 989 0974 1 ! Parse state table
: 990 0975 1 ! PARAM Address of descriptor of the string
: 991 0976 1 !
: 992 0977 1 ! IMPLICIT INPUTS:
: 993 0978 1 !
: 994 0979 1 ! NONE
: 995 0980 1 !
: 996 0981 1 ! IMPLICIT OUTPUTS:
: 997 0982 1 !
: 998 0983 1 ! NONE
: 999 0984 1 !
: 1000 0985 1 ! ROUTINE VALUE:
: 1001 0986 1 ! COMPLETION CODES:
: 1002 0987 1 !
: 1003 0988 1 ! Success
: 1004 0989 1 !
: 1005 0990 1 ! SIDE EFFECTS:
: 1006 0991 1 !
: 1007 0992 1 ! NONE
: 1008 0993 1 !
: 1009 0994 1 ! --
: 1010 0995 1 !
: 1011 0996 2 ! BEGIN
: 1012 0997 2 !
: 1013 0998 2 !
: 1014 0999 2 ! NCP$WRITE_LINE (.PARAM); ! Write the line
: 1015 1000 2 ! RETURN SUCCESS
: 1016 1001 2 !
: 1017 1002 1 ! END;

```

```

                0000 0000      .ENTRY ACT$WRI_STR, Save nothing      : 0964
                20 AC DD 00002    PUSHL PARAM                      : 0999
00000000G 00 01 FB 00005    CALLS #1, NCP$WRITE_LINE          : 1000
                50 01 D0 0000C    MOVL #1, R0                       : 1002
                04 0000F    RET

```

; Routine Size: 16 bytes. Routine Base: \$CODE\$ + 0338

SRJJC

```

: 1019 1003 1 %SBTTL 'ACT$$SIGNAL Signal and error from parse'
: 1020 1004 1 GLOBAL ROUTINE ACT$$SIGNAL (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 1021 1005 1     CHR, NUM, PARAM) =
: 1022 1006 1
: 1023 1007 1 !**
: 1024 1008 1 | FUNCTIONAL DESCRIPTION:
: 1025 1009 1 |
: 1026 1010 1 |     Signal and error from parse and return to parse
: 1027 1011 1 |
: 1028 1012 1 | FORMAL PARAMETERS:
: 1029 1013 1 |
: 1030 1014 1 |     Parse state table
: 1031 1015 1 |     PARAM     Value of status code to signal
: 1032 1016 1 |
: 1033 1017 1 | IMPLICIT INPUTS:
: 1034 1018 1 |
: 1035 1019 1 |     NONE
: 1036 1020 1 |
: 1037 1021 1 | IMPLICIT OUTPUTS:
: 1038 1022 1 |
: 1039 1023 1 |     NONE
: 1040 1024 1 |
: 1041 1025 1 | ROUTINE VALUE:
: 1042 1026 1 | COMPLETION CODES:
: 1043 1027 1 |
: 1044 1028 1 |     Success
: 1045 1029 1 |
: 1046 1030 1 | SIDE EFFECTS:
: 1047 1031 1 |
: 1048 1032 1 |     NONE
: 1049 1033 1 |
: 1050 1034 1 | --
: 1051 1035 1 |
: 1052 1036 2 | BEGIN
: 1053 1037 2 |
: 1054 1038 2 |
: 1055 1039 2 | SIGNAL (.PARAM);           ! Signal the condition to print the message
: 1056 1040 2 |
: 1057 1041 2 | RETURN SUCCESS
: 1058 1042 2 |
: 1059 1043 1 | END;

```

```

                                0000 0000      .ENTRY ACT$$SIGNAL, Save nothing      : 1004
                                20  AC  DD 00002    PUSHL  PARAM                          : 1039
                                00000000G 00 01  FB 00005    CALLS  #1, LIB$$SIGNAL
                                50          01  DO 0000C    MOVL   #1, R0                          : 1041
                                04 0000F    RET                               : 1043

```

; Routine Size: 16 bytes, Routine Base: \$CODE\$ + 0348

```

: 1061      1044  1 %SBTTL 'ACT$PMT_ON Enable prompting'
: 1062      1045  1 GLOBAL ROUTINE ACT$PMT_ON =
: 1063      1046  1
: 1064      1047  1 !++
: 1065      1048  1 FUNCTIONAL DESCRIPTION:
: 1066      1049  1
: 1067      1050  1     Action routine to enable prompting for command prompting
: 1068      1051  1
: 1069      1052  1 FORMAL PARAMETERS:
: 1070      1053  1
: 1071      1054  1     NONE
: 1072      1055  1
: 1073      1056  1 IMPLICIT INPUTS:
: 1074      1057  1
: 1075      1058  1     NONE
: 1076      1059  1
: 1077      1060  1 IMPLICIT OUTPUTS:
: 1078      1061  1
: 1079      1062  1     ACT$GL_PMT_Q
: 1080      1063  1
: 1081      1064  1 ROUTINE VALUE:
: 1082      1065  1 COMPLETION CODES:
: 1083      1066  1
: 1084      1067  1     Success
: 1085      1068  1
: 1086      1069  1 SIDE EFFECTS:
: 1087      1070  1
: 1088      1071  1     NONE
: 1089      1072  1
: 1090      1073  1 !--
: 1091      1074  1
: 1092      1075  2 BEGIN
: 1093      1076  2
: 1094      1077  2
: 1095      1078  2 GLOBAL
: 1096      1079  2     ACT$GL_PMT_Q           ! True for prompting enabled
: 1097      1080  2     ;
: 1098      1081  2
: 1099      1082  2     ACT$GL_PMT_Q = TRUE;       ! Enable prompting
: 1100      1083  2
: 1101      1084  2     RETURN SUCCESS           ! Continue the parse
: 1102      1085  2
: 1103      1086  1     END;

```

.PSECT \$GLOBALS,NOEXE,2

00054 ACT\$GL_PMT_Q::
.B[KB 4

.PSECT \$CODES,NOWRT,2

```

00000000' 00          0000 0000      .ENTRY ACT$PMT_ON, Save nothing      : 1045
01 00 00002          01 00 00002      MOVL #1, ACT$GL_PMT_Q                : 1082

```



```

: 1105      1087 1 %SBTTL 'ACT$PMT_OFF Disable prompting'
: 1106      1088 1 GLOBAL ROUTINE ACT$PMT_OFF = !
: 1107      1089 1
: 1108      1090 1 !++
: 1109      1091 1 FUNCTIONAL DESCRIPTION:
: 1110      1092 1
: 1111      1093 1     Action routine to disable command prompting
: 1112      1094 1
: 1113      1095 1 FORMAL PARAMETERS:
: 1114      1096 1
: 1115      1097 1     NONE
: 1116      1098 1
: 1117      1099 1 IMPLICIT INPUTS:
: 1118      1100 1
: 1119      1101 1     NONE
: 1120      1102 1
: 1121      1103 1 IMPLICIT OUTPUTS:
: 1122      1104 1
: 1123      1105 1     ACT$GL_PMT_Q
: 1124      1106 1
: 1125      1107 1 ROUTINE VALUE:
: 1126      1108 1 COMPLETION CODES:
: 1127      1109 1
: 1128      1110 1     Success
: 1129      1111 1
: 1130      1112 1 SIDE EFFECTS:
: 1131      1113 1
: 1132      1114 1     NONE
: 1133      1115 1
: 1134      1116 1 !--
: 1135      1117 1
: 1136      1118 2 BEGIN
: 1137      1119 2
: 1138      1120 2 ACT$GL_PMT_Q = FALSE;           ! Disable prompting
: 1139      1121 2
: 1140      1122 2 RETURN SUCCESS                 ! Continue the parse
: 1141      1123 2
: 1142      1124 1 END;

```

```

          0000 0000      .ENTRY ACT$PMT_OFF, Save nothing      : 1088
          50 0000000G 00 D4 00002      CLRL ACT$GL_PMT_Q      : 1120
          01 D0 00008      MOVL #1, R0      : 1122
          04 0000B      RET      : 1124

```

: Routine Size: 12 bytes, Routine Base: \$CODE\$ + 0365

```

: 1144 1125 1 XSBTTL 'ACT$PMT_Q Control command prompting'
: 1145 1126 1 GLOBAL ROUTINE ACT$PMT_Q (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 1146 1127 1     CHR, NOM, PARAM) =
: 1147 1128 1
: 1148 1129 1 ++
: 1149 1130 1 FUNCTIONAL DESCRIPTION:
: 1150 1131 1
: 1151 1132 1     Action routine to control command prompting
: 1152 1133 1     These three routines work with the COMMAND_PROMPT
: 1153 1134 1     macro to control command prompting. Keywords or
: 1154 1135 1     an entity is prompted if the state is entered with
: 1155 1136 1     the TPAS_EOS condition. ACT$PMT_ON and OFF are used
: 1156 1137 1     to set prompting on or off. If prompting is on and
: 1157 1138 1     any other transition fails, this action routine is called
: 1158 1139 1     to signal an error and set the EOS condition so ACT$PRMPT
: 1159 1140 1     will obtain the next string to try. The parse loops in the
: 1160 1141 1     state until an acceptable response is given or EOF causes
: 1161 1142 1     return to the initial command level.
: 1162 1143 1
: 1163 1144 1 FORMAL PARAMETERS:
: 1164 1145 1
: 1165 1146 1     Parse state table
: 1166 1147 1     PARAM          Value of status code to signal if non-zero
: 1167 1148 1
: 1168 1149 1 IMPLICIT INPUTS:
: 1169 1150 1
: 1170 1151 1     ACT$GL_PMT_Q
: 1171 1152 1
: 1172 1153 1 IMPLICIT OUTPUTS:
: 1173 1154 1
: 1174 1155 1     NONE
: 1175 1156 1
: 1176 1157 1 ROUTINE VALUE:
: 1177 1158 1 COMPLETION CODES:
: 1178 1159 1
: 1179 1160 1     Success of prompting, failure if not prompting
: 1180 1161 1
: 1181 1162 1 SIDE EFFECTS:
: 1182 1163 1
: 1183 1164 1     NONE
: 1184 1165 1
: 1185 1166 1 --
: 1186 1167 1
: 1187 1168 2 BEGIN
: 1188 1169 2
: 1189 1170 2 IF .ACT$GL_PMT_Q          ! If prompting
: 1190 1171 2 THEN
: 1191 1172 2     BEGIN
: 1192 1173 2     IF .PARAM NEQ 0      ! If condition to signal
: 1193 1174 2     THEN SIGNAL (.PARAM) ! Signal the condition
: 1194 1175 2     .
: 1195 1176 2     STRCNT = 0;         ! Set for EOS parse to occur
: 1196 1177 2     RETURN SUCCESS      ! Continue parse
: 1197 1178 2     END
: 1198 1179 2
: 1199 1180 2 ELSE
: 1200 1181 2     RETURN FAILURE      ! Cause failure in state

```


: 1201
: 1202
1182 2
1183 1 END;

		0000	00000		.ENTRY	ACT\$PMT_Q, Save nothing	:	1126
16	00000000G	00	E9 00002		BLBC	ACT\$GL_PMT_Q, 2\$:	1170
		20	AC D5 00009		TSTL	PARAM	:	1173
			0A 13 0000C		BEQL	1\$:	
		20	AC DD 0000E		PUSHL	PARAM	:	1174
00000000G	00	01	FB 00011		CALLS	#1, LIB\$SIGNAL	:	
		08	AC D4 0001B	1\$:	CLRL	STRCNT	:	1176
	50	01	D0 0001B		MOVL	#1, R0	:	1181
			04 0001E		RET		:	
		50	D4 0001F	2\$:	CLRL	R0	:	
			04 00021		RET		:	1183

: Routine Size: 34 bytes, Routine Base: \$CODE\$ + 0371

```

: 1204 1184 1 %SBTTL 'ACT$EXECQ Test if Component is Executor'
: 1205 1185 1 GLOBAL ROUTINE ACT$EXECQ (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 1206 1186 1     CHR, NUM, PDB) =
: 1207 1187 1
: 1208 1188 1 !++
: 1209 1189 1 !FUNCTIONAL DESCRIPTION:
: 1210 1190 1
: 1211 1191 1     Action routine to test if the current node component is the
: 1212 1192 1     executor node. Called from the ncpstanod module to select
: 1213 1193 1     the correct parameters for prompting.
: 1214 1194 1     The executor is coded as an address of zero, that is three
: 1215 1195 1     data bytes of zero.
: 1216 1196 1
: 1217 1197 1 !FORMAL PARAMETERS:
: 1218 1198 1
: 1219 1199 1     Parse state table
: 1220 1200 1     PDB          Address of PDB data block for the component.
: 1221 1201 1
: 1222 1202 1 !IMPLICIT INPUTS:
: 1223 1203 1
: 1224 1204 1     NONE
: 1225 1205 1
: 1226 1206 1 !IMPLICIT OUTPUTS:
: 1227 1207 1
: 1228 1208 1     NONE
: 1229 1209 1
: 1230 1210 1 !ROUTINE VALUE:
: 1231 1211 1 !COMPLETION CODES:
: 1232 1212 1
: 1233 1213 1     Success if component is executor node, failure if not
: 1234 1214 1
: 1235 1215 1 !SIDE EFFECTS:
: 1236 1216 1
: 1237 1217 1     NONE
: 1238 1218 1
: 1239 1219 1 !--
: 1240 1220 1
: 1241 1221 2     BEGIN
: 1242 1222 2
: 1243 1223 2     MAP
: 1244 1224 2     PDB : REF BBLOCK [PDB$C_SIZE] ! Pointer to the component PDB
: 1245 1225 2     ;
: 1246 1226 2
: 1247 1227 2     IF .PDB [1, 0, 24, 0] EQL 0 ! Look at three bytes of the data
: 1248 1228 2     THEN
: 1249 1229 2     RETURN SUCCESS ! It is the executor
: 1250 1230 2
: 1251 1231 2     ELSE
: 1252 1232 2     RETURN FAILURE ! Cause failure in state
: 1253 1233 2
: 1254 1234 1     END;

```

0000 00000

.ENTRY ACT\$EXECQ, Save nothing

; 1185

: R


```

: 1256 1235 1 %SBTTL 'ACT$PMTDONEQ Terminate Prompts?'
: 1257 1236 1 GLOBAL ROUTINE ACT$PMTDONEQ (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
: 1258 1237 1 CHR, NUM, PDB) =
: 1259 1238 1
: 1260 1239 1 :++
: 1261 1240 1 :FUNCTIONAL DESCRIPTION:
: 1262 1241 1
: 1263 1242 1 This routine checks the parsed token to see if it is "_DONE".
: 1264 1243 1 If so the routine returns success and the parse tables exit the
: 1265 1244 1 prompt sequence. Otherwise the routine returns false and the
: 1266 1245 1 remainder of the prompts are processed.
: 1267 1246 1
: 1268 1247 1 :FORMAL PARAMETERS:
: 1269 1248 1
: 1270 1249 1 Parse state
: 1271 1250 1 TKNCNT Descriptor of token just parsed
: 1272 1251 1 TKNPTR
: 1273 1252 1
: 1274 1253 1 :IMPLICIT INPUTS:
: 1275 1254 1
: 1276 1255 1 NONE
: 1277 1256 1
: 1278 1257 1 :IMPLICIT OUTPUTS:
: 1279 1258 1
: 1280 1259 1 NONE
: 1281 1260 1
: 1282 1261 1 :ROUTINE VALUE:
: 1283 1262 1 :COMPLETION CODES:
: 1284 1263 1
: 1285 1264 1 Success or failure
: 1286 1265 1
: 1287 1266 1 :SIDE EFFECTS:
: 1288 1267 1
: 1289 1268 1 NONE
: 1290 1269 1
: 1291 1270 1 :--
: 1292 1271 1 BEGIN
: 1293 1272 2
: 1294 1273 2
: 1295 1274 2 IF CH$EQL (.TKNCNT, .TKNPTR, 5, UPLIT BYTE ('_DONE'), 0)
: 1296 1275 2 THEN
: 1297 1276 2 RETURN SUCCESS
: 1298 1277 2 ELSE
: 1299 1278 2 RETURN FAILURE
: 1300 1279 2
: 1301 1280 1 END;

```

.PSECT \$PLITS,NOWRT,NOEXE,2

45 4E 4F 44 5F 0000 P.AAA: .ASCII _DONE\ ;

.PSECT \$CODE\$,NOWRT,2

NCPPRSACT
V04-000

Parse Data and Action Routines
ACT\$PMTDONEQ Terminate Prompts?

F 12
15-Sep-1984 23:51:04
14-Sep-1984 12:48:15

VAX-11 Bliss-32 V4.0-742
[NCP.SRC]NCPPRSACT.B32;1

Page 43
(22)

NCP
V04

05	00	14	BC	10	00000000'	000C 00000	AC 2D 00002
						00	00009
		50				04 12 0000E	01 D0 00010
						04 00013	50 D4 00014 1\$:
						04 00016	

.ENTRY	ACT\$PMTDONEQ, Save R2,R3
CMPC5	TKNCNT, @TKNPTR, #0, #5, P.AAA
BNEQ	1\$
MOVL	#1, R0
RET	
CLRL	R0
RET	

: 1236
: 1274
:
:
: 1278
:
: 1280

; Routine Size: 23 bytes, Routine Base: \$CODE\$ + 03A8

: R

```

1303 1281 1
1304 1282 1 %SBTTL 'ACT$HELP Provide prompting help'
1305 1283 1 GLOBAL ROUTINE ACT$HELP (OPT, STRCNT, STRPTR, TKNCNT, TKNPTR,
1306 1284 1 CHR, NUM, PARAM) = !
1307 1285 1
1308 1286 1
1309 1287 1 **
1310 1288 1 FUNCTIONAL DESCRIPTION:
1311 1289 1 Action routine to provide prompting help
1312 1290 1
1313 1291 1 FORMAL PARAMETERS:
1314 1292 1 Parse state table
1315 1293 1 STRCNT Size of the remainder of the command line
1316 1294 1 STRPTR Address of the remainder of the command line
1317 1295 1
1318 1296 1
1319 1297 1 IMPLICIT INPUTS:
1320 1298 1
1321 1299 1 NONE
1322 1300 1
1323 1301 1 IMPLICIT OUTPUTS:
1324 1302 1
1325 1303 1 NONE
1326 1304 1
1327 1305 1 ROUTINE VALUE:
1328 1306 1 COMPLETION CODES:
1329 1307 1
1330 1308 1 SUCCESS
1331 1309 1
1332 1310 1 SIDE EFFECTS:
1333 1311 1
1334 1312 1 NONE
1335 1313 1
1336 1314 1 --
1337 1315 1
1338 1316 2 BEGIN
1339 1317 2 LOCAL
1340 1318 2 HLP_DESC : BBLOCK [DSC$C_S_BLN],
1341 1319 2 STATUS;
1342 1320 2
1343 1321 2 CH$FILL (0, DSC$C_S_BLN, HLP_DESC); ! zero descriptor
1344 1322 2 HLP_DESC [DSC$W_LENGTH] = .STRCNT;
1345 1323 2 HLP_DESC [DSC$A_POINTER] = .STRPTR;
1346 1324 2
1347 1325 2
1348 1326 2 !
1349 1327 2 ! Request help be printed by lib$put_output to sys$output,
1350 1328 2 ! from library SYSSHELP:NCPHELP.HLB. Query for additional help
1351 1329 2 ! to sys$input using lib$get_input.
1352 1330 2 STATUS = LBR$OUTPUT_HELP (LIB$PUT_OUTPUT, 0, HLP_DESC,
1353 1331 2 $DESCRIPTOR('NCPHELP'), 0, LIB$get_input);
1354 1332 2
1355 1333 2 IF NOT .STATUS THEN SIGNAL (.STATUS);
1356 1334 2
1357 1335 2 RETURN SUCCESS
1358 1336 1 END;

```

```

.PSECT $SPLITS,NOWRT,NOEXE,2
50 4C 45 48 50 43 4E 00005 P.AAC: .ASCII \NCPHELP\
00000007 0000C P.AAB: .LONG 7
00000000' 00010 .ADDRESS P.AAC

```

```

.PSECT $CODE$,NOWRT,2
08 00 5E 08 003C 00000 .ENTRY ACT$HELP, Save R2,R3,R4,R5
6E 00 C2 00002 .SUBL2 #8, SP
04 6E 08 AC B0 0000B MOVW STRCNT, HLP_DESC
AE 0C AC D0 0000F MOVL STRPTR, HLP_DESC+4
00000000G 00 9F 00014 PUSHAB LIB$GET_INPUT
00000000' 7E D4 0001A CLRL -(SP)
0C AE 9F 0001C PUSHAB P.AAB
00000000G 7E D4 00022 PUSHAB HLP_DESC
00 9F 00025 CLRL -(SP)
00000000G 00 9F 00027 PUSHAB LIB$PUT_OUTPUT
00 06 FB 0002D CALLS #6, LIB$OUTPUT_HELP
09 50 E8 00034 BLBS STATUS, 1$
00000000G 00 50 DD 00037 PUSHL STATUS
00 01 FB 00039 CALLS #1, LIB$SIGNAL
50 01 D0 00040 1$: MOVL #1, R0
04 00043 RET

```

: Routine Size: 68 bytes, Routine Base: \$CODE\$ + 03BF

: 1360
: 1361
1337 1 END
1338 0 ELUDOM

!End of module

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	88	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	36	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	1027	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
ABS	0	NOVEC, NOWRT, NORD, NOEXE, NOSHR, LCL, ABS, CON, NOPIC, ALIGN(0)
\$SPLITS	20	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Symbols -----		Pages Mapped	Processing Time
	Total	Loaded		
\$_\$255\$DUA28:[NCP.OBJ]NCPLIBRY.L32;1	373	7	52	00:00.1
\$_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	18	581	00:01.4

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:NCPPRSACT/OBJ=OBJ\$:NCPPRSACT MSRCS:NCPPRSACT/UPDATE=(ENHS:NCPPRSACT)

: Size: 1027 code + 144 data bytes
: Run Time: 00:21.0
: Elapsed Time: 01:23.1
: Lines/CPU Min: 3819
: Lexemes/CPU-Min: 11009
: Memory Used: 87 pages
: Compilation Complete

The image displays a grid of 144 terminal windows, arranged in 12 rows and 12 columns. Each window contains text-based data, likely system logs or diagnostic information. The text is small and dense, but several windows are clearly legible and contain the following text:

- Top Center:** NCP... LIS
- Bottom Left:** NCP... LIS
- Bottom Center:** NCP... LIS
- Bottom Right:** NCP... LIS

Other windows show various alphanumeric strings, some resembling error codes or system identifiers, such as 'VMS', 'LIS', and 'NCP...'. The overall appearance is that of a multi-processor system's diagnostic output.