

RRRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPPP		AAAAAAAAAA		DDDDDDDDDDDD	
RRRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPPP		AAAAAAAAAA		DDDDDDDDDDDD	
RRRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPPP		AAAAAAAAAA		DDDDDDDDDDDD	
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRRRRRRRRRRR		TTT	PPPPPPPPPPPP		AAA	AAA	DDD	DDD
RRRRRRRRRRRR		TTT	PPPPPPPPPPPP		AAA	AAA	DDD	DDD
RRRRRRRRRRRR		TTT	PPPPPPPPPPPP		AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP		AAAAAAAAAAAAAAAA		DDD	DDD
RRR	RRR	TTT	PPP		AAAAAAAAAAAAAAAA		DDD	DDD
RRR	RRR	TTT	PPP		AAAAAAAAAAAAAAAA		DDD	DDD
RRR	RRR	TTT	PPP		AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP		AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP		AAA	AAA	DDD	DDD
RRR	RRR	TTT	PPP		AAA	AAA	DDDDDDDDDDDD	
RRR	RRR	TTT	PPP		AAA	AAA	DDDDDDDDDDDD	
RRR	RRR	TTT	PPP		AAA	AAA	DDDDDDDDDDDD	

VM
Ta

```

RRRRRRRRR      TTTTTTTTTT  VV      VV  MM      MM      SSSSSSSS  PPPPPPPP  EEEEEEEEEEE  CCCCCCCC
RRRRRRRRR      TTTTTTTTTT  VV      VV  MM      MM      SSSSSSSS  PPPPPPPP  EEEEEEEEEEE  CCCCCCCC
RR      RR      TT      TT      VV      VV  MM      MM      SS      SS      PP      PP  EE      EE  CC
RR      RR      TT      TT      VV      VV  MM      MM      SS      SS      PP      PP  EE      EE  CC
RR      RR      TT      TT      VV      VV  MM      MM      SS      SS      PP      PP  EE      EE  CC
RRRRRRRRR      TT      TT      VV      VV  MM      MM      SSSSSS  SSSSSS  PPPPPPPP  EEEEEEEEE  CC
RRRRRRRRR      TT      TT      VV      VV  MM      MM      SSSSSS  SSSSSS  PPPPPPPP  EEEEEEEEE  CC
RR  RR      TT      TT      VV      VV  MM      MM      SS      SS      PP      PP  EE      EE  CC
RR  RR      TT      TT      VV      VV  MM      MM      SS      SS      PP      PP  EE      EE  CC
RR      RR      TT      TT      VV      VV  MM      MM      SS      SS      PP      PP  EE      EE  CC
RR      RR      TT      TT      VV      VV  MM      MM      SS      SS      PP      PP  EE      EE  CC
RR      RR      TT      TT      VV      VV  MM      MM      SSSSSSSS  SSSSSSSS  PP      PP  EEEEEEEEEEE  CCCCCCCC
RR      RR      TT      TT      VV      VV  MM      MM      SSSSSSSS  SSSSSSSS  PP      PP  EEEEEEEEEEE  CCCCCCCC

```

```

LL      IIIIIII  SSSSSSSS
LL      IIIIIII  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
LLLLLLLLLLL  IIIIIII  SSSSSSSS
LLLLLLLLLLL  IIIIIII  SSSSSSSS

```


(1)	56
(2)	89
(3)	208
(4)	300

DECLARATIONS
CTERM_VMSQIO - Handle VMS Qio
CT_VMSQIO_DONE - complete VMS qio
XXR -

```
0000 1 .TITLE RTVMSPEC - RTPAD/CTERM VMS specific protocol
0000 2 .IDENT 'V04-000'
0000 3 :
0000 4 :*****
0000 5 :*
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :
0000 27 :
0000 28 :++
0000 29 :
0000 30 : FACILITY:
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 :
0000 35 : ENVIRONMENT:
0000 36 :
0000 37 :
0000 38 :--
0000 39 :
0000 40 : AUTHOR: Jake VanNoy, CREATION DATE: 5-Jan-83
0000 41 :
0000 42 : MODIFIED BY:
0000 43 :
0000 44 : V03-003 JLV0291 Jake VanNoy 28-JUL-1983
0000 45 : Changes to align this module with ECO's specified in
0000 46 : January for CTERM extensions.
0000 47 :
0000 48 : V03-002 JLV0238 Jake VanNoy 29-MAR-1983
0000 49 : Change name of module from CTVMSQ10.
0000 50 :
0000 51 : V03-001 MHB0090 Mark Bramhall 2-Mar-1983
0000 52 : Added correct .PSECT usage.
0000 53 :**
0000 54 :
0000 55 :
0000 56 : .SBTTL DECLARATIONS
0000 57 :
```



```
0000 58 ; INCLUDE FILES:
0000 59 ;
0000 60      $RTPADDEF
0000 61      $TSADEF
0000 62
0000 63 ;
0000 64 ; MACROS:
0000 65 ;
0000 66 ;
0000 67 ;
0000 68 ; EQUATED SYMBOLS:
0000 69 ;
0000 70 ;
0000 71 ;
0000 72 ; OWN STORAGE:
0000 73 ;
0000 74
00000000 75      .PSECT  _RTPAD, LONG      ; Read/Write data
0000 76
0000 77 QREGISTERS:
00000004 0000 78 QR1:      .BLKL  1
00000008 0004 79 QR2:      .BLKL  1
0000000C 0008 80 QR3:      .BLKL  1
00000010 000C 81 QR4:      .BLKL  1
00000014 0010 82 QR5:      .BLKL  1
00000018 0014 83 QR6:      .BLKL  1
0000001C 0018 84 QCHAN:     .BLKL  1
00000020 001C 85 QFUNC:     .BLKL  1
0000 0020 86
00000000 87      .PSECT  RTPAD, NOWRT      ; Code
```

```
0000 89      .SBTTL CTERM_VMSQIO - Handle VMS Qio
0000 90
0000 91      :++
0000 92
0000 93      : FUNCTIONAL DESCRIPTION:
0000 94
0000 95      :   Take VAX/VMS 'mode message' and format it into a QIO.
0000 96      :   *** This module may also handle RTPAD/TIMODExxx interactions.
0000 97
0000 98      : CALLING SEQUENCE:
0000 99      :   BSBW      CTERM_VMSQIO
0000 100
0000 101     : INPUT PARAMETERS:
0000 102     :   R0       - AST BLOCK
0000 103     :   R1-R6    - zero
0000 104     :   R7       - READCHAN
0000 105     :   R8       - uninitialized
0000 106     :   R11      - data block
0000 107
0000 108     : IMPLICIT INPUTS:
0000 109     :   WRITECHAN
0000 110
0000 111
0000 112     : OUTPUT PARAMETERS:
0000 113     :   R0       - preserved
0000 114     :   R1-R6    - P1-P6 of QIO
0000 115     :   R7       - channel number
0000 116     :   R8       - function code
0000 117     :   R11      - preserved
0000 118
0000 119     : IMPLICIT OUTPUTS:
0000 120     :   NONE
0000 121
0000 122     : COMPLETION CODES:
0000 123     :   NONE
0000 124
0000 125     : SIDE EFFECTS:
0000 126     :   NONE
0000 127
0000 128     :--
0000 129
0000 130     CTERM_VMSQIO::
0000 131
0000 132     CLRQ    W^QR1      : Initialize outputs
0000 133     CLRQ    W^QR3
0000 134     CLRQ    W^QR5
0000 135     MOVL    R7,W^QCHAN : Save
0000 136     CLRL    W^QFUNC
0000 137     CLRL    AST$$_ODATA(R0) : assume no buffer
0000 138
0000 139     MOVZWL   CTP$W_MSGSIZE(R11),R9 : Get length of message
0000 140     SUBL2    #<CTP$W_VMS_PLEN-CTP$B_MSGTYPE>,R9 : minus overhead
0000 141
0000 142
0000 143     CMPW     #CTP$C_MT_VMSQIO,- : Compare to VMS QIO
0000 144     CTP$B_MSGTYPE(R11)
0000 145     BEQL     20$
```

0000	CF	7C	0000	132	
0008	CF	7C	0004	133	
0010	CF	7C	0008	134	
C018	CF	57	D0	000C	135
	001C	CF	D4	C011	136
	12	A0	D4	0015	137
				0018	138
59	28	AB	3C	0018	139
	59	08	C2	001C	140
				001F	141
				001F	142
	0F	B1	001F	143	
2A	AB		0021	144	
	0D	13	0023	145	


```
00000000'8F DD 0025 146
00000000'GF 01 FB 0025 147 ; *** Alternate mode must be handled here ***
                                0025 148
                                0025 149 ; MINOR_ERROR
                                0025 150 PUSHL #shr$ valerr
                                0028 151 CALLS #1,G^LIB$SIGNAL ; error ***
                                0032 152 ; *** what to do here?
                                0032 153
                                0032 154 ;
                                0032 155 ; VMS mode qio
                                0032 156 ;
                                0032 157 20$:
30 AB 3C 0032 158 MOVZWL CTP$W_VMS_FUNC(R11),-
001C'CF 0035 159 W^QFUNC ; Function code
5A 32 AB 9E 0038 160 MOVAB CTP$W_VMS_PLEN(R11),R10 ; Get address of first paramter
59 5A C0 003C 161 ADDL2 R10,R9 ; end address = base + size
                                003F 162 ;
                                003F 163 ; Loop through paramter descriptors
                                003F 164
003F 165 PARAM_LOOP:
5A 59 D1 003F 166 CMPL R9,R10 ; address at or past end ?
48 15 0042 167 BLEQ 200$ ; Branch if run out of data
58 6A 3C 0044 168 MOVZWL VMS$W_PLEN(R10),R8 ; Get length
                                0047 169
02 E1 0047 170 BBC #CTP$V_VMS_BUFFER,-
1B 02 AA 0049 171 VMS$W_PFLAGS(R10),35$ ; branch if data is present
                                004C 172 ;
                                004C 173 ; This is a return buffer, must supply buffer
                                004C 174 ;
50 DD 004C 175 PUSHL R0 ; Save
FFAF' 30 004E 176 BSBW GETBUF ; Get buffer
52 50 D0 0051 177 MOVL R0,R2 ; Move address to R2
50 8ED0 0054 178 POPL R0 ; Restore
51 38 A2 9E 0057 179 MOVAB CTP$T_VMS_RDATA(R2),R1 ; Set address
12 A0 52 D0 005B 180 MOVL R2,AST$L_ODATA(R0) ; save address of buffer
32 AB 58 B0 005F 181 MOVW R8,CTP$W_VMS_PLEN(R11) ; Save requested length ***
58 D4 0063 182 clrl r8 ; *** null data...
OD 11 0065 183 BRB 40$ ; Skip
                                0067 184 35$:
51 06 AA 9E 0067 185 MOVAB VMS$T_PDATA(R10),R1 ; Assume passed by reference
00 E0 006B 186 BBS #CTP$V_VMS_REF,- ; Branch if passed by refererence
04 02 AA 006D 187 VMS$W_PFLAGS(R10),40$
51 06 AA D0 0070 188 MOVL VMS$T_PDATA(R10),R1 ; Set data
                                0074 189 40$:
                                0074 190 ; *** assumes qio parameter, should use flag #1 to validate
                                0074 191
52 04 AA 3C 0074 192 MOVZWL VMS$W_PCODE(R10),R2 ; Fetch paramter number (PCODE ???)
53 0000'CF 9E 0078 193 MOVAB W^QREGISTERS,R3 ; base address
FC A342 51 D0 007D 194 MOVL R1,-4(R3)[R2] ; set value
                                0082 195
5A 06 AA 9E 0082 196 MOVAB VMS$T_PDATA(R10),R10 ; add offset to R10
5A 58 C0 0086 197 ADDL2 R8,R10 ; Add size of data field
FFB3 31 0089 198 BRW PARAM_LOOP ; Loop
                                008C 199
                                008C 200 200$: ; return to do qio
                                008C 201
51 0000'CF 7D 008C 202 MOVQ W^QR1,R1 ; Set outputs
```

53	0008'CF	7D	0091	203	MOVQ	W^QR3,R3	:
55	0010'CF	7D	0096	204	MOVQ	W^QR5,R5	:
57	0018'CF	7D	009B	205	MOVQ	W^QCHAN,R7	:
		05	00A0	206	RSB		:


```
00A1 208 .SBTTL CT_VMSQIO_DONE - complete VMS qio
00A1 209
00A1 210 :++
00A1 211 :
00A1 212 : FUNCTIONAL DESCRIPTION:
00A1 213 :
00A1 214 :
00A1 215 : CALLING SEQUENCE:
00A1 216 : NONE
00A1 217 :
00A1 218 : INPUT PARAMETERS:
00A1 219 : NONE
00A1 220 :
00A1 221 : IMPLICIT INPUTS:
00A1 222 : NONE
00A1 223 :
00A1 224 : OUTPUT PARAMETERS:
00A1 225 : NONE
00A1 226 :
00A1 227 : IMPLICIT OUTPUTS:
00A1 228 : NONE
00A1 229 :
00A1 230 : COMPLETION CODES:
00A1 231 : NONE
00A1 232 :
00A1 233 : SIDE EFFECTS:
00A1 234 : NONE
00A1 235 :
00A1 236 :--
00A1 237
00A1 238 CT_VMSQIO_DONE::
00A1 239
00A1 240 :
00A1 241 : *** should validate that this is a VAX QIO mode message
00A1 242 :
00000000'EF D7 00A1 243 DECL WRITEQIO ;
2C AB D5 00A7 244 TSTL CTP$$_VMS_REQID(R11) ; response expected?
03 12 00AA 245 BNEQ 110$ ; branch if yes
004A 31 00AC 246 BRW VMSQIO_DONE_EXIT ; done
53 12 A0 D0 00AF 247 110$: MOVL AST$_ODATA(R0),R3 ; Buffer address
0B 12 00B3 248 BNEQ 120$ ; Branch if there is one (data returned)
00B5 249
00B5 250 ;
00B5 251 ; There is no data to return, only status (i.e. IOSB)
00B5 252 ;
50 DD 00B5 253 PUSHL R0 ; Save R0
FF46' 30 00B7 254 BSBW GETBUF ; Get buffer to write to net with
53 50 D0 00BA 255 MOVL R0,R3 ; Set address
50 8ED0 00BD 256 POPL R0 ; Restore R0
00C0 257
00C0 258 ; Initialize header of message to be written and copy IOSB
00C0 259 ;
00C0 260 120$:
09 9B 00C0 261 MOVZBW #PROSC DATA,-
26 A3 00C2 262 CTP$$_PRO_MSGTYPE(R3) ; mode message, clear flags
0F 9B 00C4 263 MOVZBW #CTP$$_MT_VMSQIO,-
2A A3 00C6 264 CTP$$_MSGTYPE(R3) ; qio request message
```

```
2C AB D0 00C8 265      MOVL      CTP$VMS_REQID(R11),-
2C A3      00CB 266      CTP$VMS_REQID(R3)      ; Set request ID
04 A0 7D 00CD 267      MOVQ      AST$Q_IOSB(R0),-
30 A3      00D0 268      CTP$Q_VMS_IOSB(R3)      ; set iosb
      00D2 269      ;
      00D2 270      ; set up for writing to net, calculate length
      00D2 271      ;
51 0E 9A 00D2 272      MOVZBL    #<CTP$T_VMS_RDATA-CTP$B_MSGTYPE>,R1
      00D5 273      ; Size of message up to data
      12 A0 D5 00D5 274      TSTL      AST$Q_ODATA(R0)      ; Was data returned from QIO?
      1B 13 00D8 275      BEQL      140$      ; Branch if not, IOSB only
      00 E0 00DA 276      BBS      #CTP$V_VMS_USE_IOSB,-
09 2B AB 00DC 277      CTP$B_VMS_FLAGS(R11),125$ ; Branch if should use IOSB for len
7E 32 AB 3C 00DF 278      MOVZWL    CTP$W_VMS_PLEN(R11),-(SP) ; fetch length requested
51 8E C0 00E3 279      ADDL2      (SP)+,R1      ; add to overhead
      0D 11 00E6 280      BRB      140$
      00E8 281      ;
      00E8 282      ; There are two types of IOSB's to understand to calculate length
      00E8 283      ; of the data returned. The host driver has set the READLEN flag
      00E8 284      ; if the format is (IOSB+2)+(IOSB+6), otherwise, length is
      00E8 285      ; contained in (IOSB+2), as in a WRITE QIO.
      00E8 286      ;
      00E8 287 125$:
      01 E1 00E8 288      BBC      #CTP$V_VMS_READLEN,-
04 2B AB 00EA 289      CTP$B_VMS_FLAGS(R11),130$ ; Branch if right
51 0A A0 A0 00ED 290      ADDW2     AST$Q_IOSB+6(R0),R1      ; len is (IOSB+2)+(IOSB+6)
      00F1 291 130$:
51 06 A0 A0 00F1 292      ADDW2     AST$Q_IOSB+2(R0),R1      ; add in (IOSB+2)
      00F5 293 140$:
52 26 A3 9E 00F5 294      MOVAB     CTP$B_PRO_MSGTYPE(R3),R2 ; Address of message
      00F9 295
      00F9 296 VMSQIO_DONE_EXIT:
      05 00F9 297      RSB
      00FA 298
```



```
00FA 300      .SBTTL  XXX -
00FA 301
00FA 302 :++
00FA 303 :
00FA 304 : FUNCTIONAL DESCRIPTION:
00FA 305 :
00FA 306 :
00FA 307 : CALLING SEQUENCE:
00FA 308 :     NONE
00FA 309 :
00FA 310 : INPUT PARAMETERS:
00FA 311 :     NONE
00FA 312 :
00FA 313 : IMPLICIT INPUTS:
00FA 314 :     NONE
00FA 315 :
00FA 316 : OUTPUT PARAMETERS:
00FA 317 :     NONE
00FA 318 :
00FA 319 : IMPLICIT OUTPUTS:
00FA 320 :     NONE
00FA 321 :
00FA 322 : COMPLETION CODES:
00FA 323 :     NONE
00FA 324 :
00FA 325 : SIDE EFFECTS:
00FA 326 :     NONE
00FA 327 :
00FA 328 :--
00FA 329 :
00FA 330 :
00FA 331 : .ENTRY  XXX,^M<>
00FA 332 :
00FA 333 :
00FA 334      .END
```

RTVMSPEC
Symbol table

- RTPAD/CTERM VMS specific protocol ^{F 4}

16-SEP-1984 02:13:55 VAX/VMS Macro V04-00
5-SEP-1984 03:15:57 [RTPAD.SRC]RTVMSPEC.MAR;1

Page 9
(4)

AST\$L_ODATA	= 00000012		
AST\$Q_IOSB	= 00000004		
CTERM_VMSQIO	= 00000000	RG	03
CTPSB_MSGTYPE	= 0000002A		
CTPSB_PRO_MSGTYPE	= 00000026		
CTPSB_VMS_FLAGS	= 0000002B		
CTPSC_MT_VMSQIO	= 0000000F		
CTPSL_VMS_REQID	= 0000002C		
CTPSQ_VMS_IOSB	= 00000030		
CTPST_VMS_RDATA	= 00000038		
CTPSV_VMS_BUFFER	= 00000002		
CTPSV_VMS_READLEN	= 00000001		
CTPSV_VMS_REF	= 00000000		
CTPSV_VMS_USEIOSB	= 00000000		
CTPSW_MSGSIZE	= 00000028		
CTPSW_VMS_FUNC	= 00000030		
CTPSW_VMS_PLEN	= 00000032		
CT_VMSQIO_DONE	= 000000A1	RG	03
GETBUF	*****	X	03
LIB\$SIGNAL	*****	X	03
PARAM_LOOP	= 0000003F	R	03
PROSC_DATA	= 00000009		
QCHAN	= 00000018	R	02
QFUNC	= 0000001C	R	02
QR1	= 00000000	R	02
QR2	= 00000004	R	02
QR3	= 00000008	R	02
QR4	= 0000000C	R	02
QR5	= 00000010	R	02
QR6	= 00000014	R	02
QREGISTERS	= 00000000	R	02
SHR\$ VALERR	*****	X	03
VM\$ST_PDATA	= 00000006		
VM\$SW_PCODE	= 00000004		
VM\$SW_PFLAGS	= 00000002		
VM\$SW_PLEN	= 00000000		
VMSQIO_DONE_EXIT	= 000000F9	R	03
WRITEQIO	*****	X	03

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes														
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
\$ABSS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE				
RTPAD	00000020 (32.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG				
RTPAD	000000FA (250.)	03 (3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE				

VMS
V04

0

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	33	00:00:00.04	00:00:01.93
Command processing	127	00:00:00.45	00:00:03.66
Pass 1	219	00:00:03.10	00:00:13.66
Symbol table sort	0	00:00:00.42	00:00:01.69
Pass 2	67	00:00:00.74	00:00:02.36
Symbol table output	6	00:00:00.04	00:00:00.04
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	456	00:00:04.81	00:00:23.37

The working set limit was 1200 pages.

25500 bytes (50 pages) of virtual memory were used to buffer the intermediate code.

There were 30 pages of symbol table space allocated to hold 476 non-local and 9 local symbols.

334 source lines were read in Pass 1, producing 15 object records in Pass 2.

9 pages of virtual memory were used to define 8 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[RTPAD.OBJ]RTPAD.MLB;1	2
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	3
TOTALS (all libraries)	5

522 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:RTVMSPEC/OBJ=OBJ\$:RTVMSPEC MSRC\$:RTVMSPEC/UPDATE=(ENH\$:RTVMSPEC)+EXECML\$/LIB+LIB\$:RTPAD/LIB

0335

AH-BT13A-SE
 VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY