RRRRRRRRRRR TTTTTTTTTTTTTTT PPPPPPPPPPP AAAAAAAAA DDDDDDDDDD
RRRRRRRRRRR TTTTTTTTTTTTTTT PPPPPPPPPPP AAAAAAAAA DDDDDDDDDD
RRRRRRRRRRR TTTTTTTTTTTTTTT PPPPPPPPPPP AAAAAAAAA DDDDDDDDDD
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
RRRRRRRRRRR TTT PPPPPPPPPPP AAA AAA DDD DDD
RRRRRRRRRRR TTT PPPPPPPPPPP AAA AAA DDD DDD
RRRRRRRRRRR TTT PPPPPPPPPPP AAA AAA DDD DDD
RRR RRR TTT PPP AAAAAAAAAAAAAAA DDD DDD
RRR RRR TTT PPP AAAAAAAAAAAAAAA DDD DDD
RRR RRR TTT PPP AAAAAAAAAAAAAAA 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 DDDDDDDDDDD
RRR RRR TTT PPP AAA AAA DDDDDDDDDDD
RRR RRR TTT PPP AAA AAA DDDDDDDDDDD

FILEID**RTDEF

```
RRRRRRRR    TTTTTTTTTT  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
RRRRRRRR    TTTTTTTTTT  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF
RR      RR      TT      DD      DD  EE          FF
RR      RR      TT      DD      DD  EE          FF
RR      RR      TT      DD      DD  EE          FF
RR      RR      TT      DD      DD  EE          FF
RRRRRRRR        TT      DD      DD  EEEEEEEE    FFFFFFFF
RRRRRRRR        TT      DD      DD  EEEEEEEE    FFFFFFFF
RR  RR          TT      DD      DD  EE          FF
RR  RR          TT      DD      DD  EE          FF
RR    RR        TT      DD      DD  EE          FF              ....
RR    RR        TT      DD      DD  EE          FF              ....
RR      RR      TT      DDDDDDDD    EEEEEEEEEE  FF              ....
RR      RR      TT      DDDDDDDD    EEEEEEEEEE  FF              ....

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   {       Version 'V04-000'
2   {
3   {*****************************************************************************
4   {*                                                                           *
5   {*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
6   {*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
7   {*  ALL RIGHTS RESERVED.                                                     *
8   {*                                                                           *
9   {*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
10  {*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
11  {*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
12  {*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
13  {*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
14  {*  TRANSFERRED.                                                             *
15  {*                                                                           *
16  {*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
17  {*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
18  {*  CORPORATION.                                                             *
19  {*                                                                           *
20  {*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
21  {*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
22  {*                                                                           *
23  {*                                                                           *
24  {*****************************************************************************
25  {
26  {       V03-004 JLV0351         Jake VanNoy             10-APR-1984
27  {               Add UNBIND constants.
28  {
29  {       V03-003 JLV0334         Jake VanNoy             28-FEB-1984
30  {               Add new constants.
31  {
32  {       V03-002 JLV0293         Jake VanNoy             28-JUL-1983
33  {               Added FLG$ symbols. Include $TSADEF. Add read
34  {               verify and upline broadcast symbols.
35  {
36  {       V03-001 MHB0091         Mark Bramhall            3-Mar-1983
37  {               Added constant MAXMSG.
38
39  MODULE  $RTPADDEF;
40
41  CONSTANT maxmsg EQUALS 1050;    { Maximum link data message size
42
43      AGGREGATE AST_BLOCK STRUCTURE PREFIX AST$;
44
45          STATE   LONGWORD;       /* AST ROUTINE (STATE)
46          IOSB    QUADWORD;       /* IOSB
47          OPCODE  WORD;           /* OPCODE (VMS RTT mode)
48          OFFSET  WORD;           /* OFFSET (CTERM)
49          BUFSIZ  WORD;           /* BUFFER SIZE (CTERM)
50          ODATA   LONGWORD;       /* OUTPUT DATA BUFFER (CTERM)
51          ITMLST  LONGWORD;       /* ADDRESS OF ITEM LIST FOR READ
52      END;
53
54      AGGREGATE CTERM_FLAGS STRUCTURE PREFIX FLG$;
55
56          CTERM           bitfield mask;  /* cterm protocol is running
57          CTRL_CY         bitfield mask;  /* flushing due to ^C or ^Y
```

```
58          CTRL_O          bitfield mask;  /* Control O state
59          LOGGING         bitfield mask;  /* Logging output
60          VAXHOST         bitfield mask;  /* HOST is a VAX
61          CTRLC           bitfield mask;  /* enable standard ^C
62      END;
63
64      /* FLAGS DEFINED IN "RTPAD$LOG"
65
66      AGGREGATE RTLOG_DBGFLAGS STRUCTURE PREFIX RTLOG$;
67
68          BANNER          bitfield mask;  /* protocol banner
69          TRACE           bitfield mask;  /* enable tracing to RTPAD$TRACE
70
71      END;
72  /*
73  /* Event Flags
74  /*
75      CONSTANT (
76          LINKEFN                         /* NET LINK
77              ) EQUALS 1 INCREMENT 1 PREFIX RT$ TAG C;
78
79  END_MODULE;
80
```



```
81  MODULE  $RTEDEF;
82
83      AGGREGATE RTE_BLOCK STRUCTURE PREFIX RTE$·
84
85      CONSTANT buflen EQUALS 80 TAG "c";  ( Maximum message size
86      flink       ADDRESS;                /* forward link
87      blink       ADDRESS;                /* backward link
88      size        WORD unsigned;          /* size of structure
89      spare1      WORD;                   /* spare byte
90      iosb        QUADWORD unsigned;      /* IOSB
91      buf         CHARACTER LENGTH 80;    /* must match buflen above
92      CONSTANT    length EQUALS . TAG "c";
93      END;
94
95
96  END_MODULE;
97
```



```
98  MODULE  $TSADEF;
99
100 CONSTANT (
101         bind,                           ( bind request
102         unbind,                         ( unbind request
103         rebind,                         ( rebind request
104         accept,                         ( bind accept
105         entermode,                      ( enter mode
106         exitmode,                       ( exit mode
107         confirm,                        ( confirm mode
108         nomode,                         ( no mode
109         data,                           ( data (cterm message)
110         mode                            ( mode message
111          ) EQUALS 1 INCREMENT 1 PREFIX "PRO$" TAG "C";
112
113 AGGREGATE oob STRUCTURE PREFIX oob TAG '"';
```

```
114
115         len_exclude LONGWORD TAG '";          /* Lengths are not used
116         exclude     LONGWORD TAG '";
117         len_include LONGWORD TAG '";
118         include     LONGWORD TAG '";
119         len_abort   LONGWORD TAG '";
120         abort       LONGWORD TAG '";
121         discard     LONGWORD TAG '";          /* discard output mask
122         echo        LONGWORD TAG '";          /* standard echo mask
123         CONSTANT    len EQUALS . TAG '";
124     END;
125
126     /*
127     /* Unbind reason codes
128     /*
129
130     CONSTANT (
131             badvers,                          { incompatible version
132             noport,                           { no portal available
133             user,                             { user requested unbind (logout)
134             disconnect,                       { disconnect (setmode hangup)
135             unused1,
136             unused2,
137             proterr                           { protocol error
138              ) EQUALS 1 INCREMENT 1 PREFIX "unbind$" TAG "c";
139
140     AGGREGATE cterm STRUCTURE PREFIX ctp$ ; /* cterm packet
141
142     /* Up to DATSIZE matches RTTDRIVER RBF header
143
144         flink       LONGWORD;                 /* forward link
145         blink       LONGWORD;                 /* backward link
146         size        WORD;                     /* size of structure
147         type        BYTE;                     /* DYN code (BUFIO)
148         spare1      BYTE;                     /* spare byte
149         msgdat      LONGWORD;                 /* message address
150         usrbfr      LONGWORD;                 /* user buffer
151         datsize     WORD;                     /* data size
152         irp         LONGWORD;                 /* address of associated IRP
153         jib         LONGWORD;                 /* address of associated JIB
154
```



```
155         spare2      LONGWORD;                 /* spare for RTPAD?
156         spare3      LONGWORD;                 /* spare for RTPAD?
157
158         #header = .;
159
160     /* start of protocol message
161
162         pro_msgtype BYTE;                     /* Protocol message type
163         pro_fill    BYTE;                     /* Protocol fill
164
165     /* start of cterm data packet
166
167         msgsize     WORD;                     /* length of first message
168         #header2 = .;
169
170         msgtype     BYTE;                     /* message type
171
172             CONSTANT (
173                 init,                         { Initiate              (H <---> S)
```

```
174             start_rd,                   { Start Read            (H  ---> S)
175             read_data,                  { Read Data             (H <---  S)
176             out_band,                   { Out-of-Band           (H <---  S)
177             unread,                     { Unread                (H  ---> S)
178             clr_input,                  { Clear Input           (H  ---> S)
179             write,                      { Write                 (H  ---> S)
180             write_com,                  { Write Completion      (H <---  S)
181             dis_state,                  { Discard State         (H <---  S)
182             read_char,                  { Read Characteristics  (H  ---> S)
183             char,                       { Characteristics       (H <---> S)
184             check_inp,                  { Check Input           (H  ---> S)
185             inp_count,                  { Input Count           (H <---  S)
186             inp_state,                  { Input State           (H <---  S)
187             vmsqio,                     { VMS specific QIO      (H <---> S)
188             vms_brdcst,                 { VMS spec broadcast    (H <---  S)
189             vms_readvfy )               { VMS spec read verify  (H  ---> S)
190         EQUALS T INCREMENT 1 TAG "c_mt";
191 /*
192 /* Remainder of block overlaid based on value of msgtype:
193 /*
194     msgfields UNION;
195
196
```



```
197 /*
198 /* init message structure (H <---> S)
199 /*
200     init STRUCTURE;
201         in_flags        BYTE;                   /* no flags defined
202         in_version      BYTE;                   /* protocol version number
203         in_eco          BYTE;                   /* ECO number for protocol
204         in_mod          BYTE;                   /* customer modification number
205         in_revision     CHARACTER LENGTH 8;     /* software revision number
206         in_parmtype     BYTE;                   /* purpose of the following value
207         in_parmval      BYTE;                   /* byte count
208
209         CONSTANT len    EQUALS . TAG "c_in";        /* length of structure
210         CONSTANT msglen EQUALS .-#header TAG "c_in";/* length of structure minus header
211         CONSTANT prolen EQUALS .-#header2 TAG "c_in";/* length of structure minus header
212
213     END init;
214
```



```
215 /*
216 /* start read and read verify structure (H ---> S)
217 /*
218     start_rd STRUCTURE;
219         sr_flags_overlay union fill;    /* Flags for unread
220           sr_flags character length 3 TAG "L";  /* 3 bytes of flags
221           sr_flag_bits structure fill;
222             sr_underflo BITFIELD LENGTH 2;      /* - underflow handling
223                 CONSTANT (
224                     ignore,                     { -- ignore underflow
225                     bel,                        { -- ring bell on underflow
226                     terminate )                 { -- terminate on underflow
227                 EQUALS 0 INCREMENT 1 TAG "m_sr";
228             sr_purge    BITFIELD MASK;          /* - purge type ahead
229             sr_format   BITFIELD MASK;          /* - formatting flag
230             sr_trmvert  BITFIELD MASK;          /* - terminate on vertical
```

```
231             sr_continue BITFIELD MASK;          /* - continuation read
232             #shift = ^;
233             sr_cvtlow   BITFIELD LENGTH 2;      /* - raise input
234                 CONSTANT (
235                     no_cvt,                     { -- use upper/lower characteristic
236                     none_only,                  { -- none this read only
237                     lowtoup )                   { -- Normal lower to upper
238                 EQUALS 0 INCREMENT 1 TAG "c_sr";
239                 CONSTANT no_cvt    EQUALS ctp$c_sr_no_cvt@#shift TAG "m_sr";
240                 CONSTANT none_only EQUALS ctp$c_sr_none_only@#shift TAG "m_sr";
241                 CONSTANT lowtoup   EQUALS ctp$c_sr_lowtoup@#shift TAG "m_sr";
242             #shift = ^;
243             sr_control  BITFIELD LENGTH 3;      /* - disable control
244                 CONSTANT (
245                     no_ctrl,                    { -- no control characters disabled
246                     u_and_r,                    { -- ^U and ^R disabled
247                     edit,                       { -- all edit control characters
248                     allbutx,                    { -- all but XON/XOFF
249                     all )                       { -- all
250                 EQUALS 0 INCREMENT 1 TAG "c_sr";
251                 CONSTANT no_ctrl EQUALS ctp$c_sr_no_ctrl@#shift TAG "m_sr";
252                 CONSTANT u_and_r EQUALS ctp$c_sr_u_and_r@#shift TAG "m_sr";
253                 CONSTANT edit    EQUALS ctp$c_sr_edit@#shift    TAG "m_sr";
254                 CONSTANT allbutx EQUALS ctp$c_sr_allbutx@#shift TAG "m_sr";
255                 CONSTANT all     EQUALS ctp$c_sr_all@#shift     TAG "m_sr";
256             sr_noecho   BITFIELD MASK;          7* - no echo read
257             sr_trmecho BITFIELD MASK;           /* - terminator echo
258             sr_timed    BITFIELD MASK;          /* - read timeout
259             #shift = ^;
260             sr_term_set BITFIELD LENGTH 2;      /* - termination set
261                 CONSTANT (
262                     prevterm,                   { -- use previous read terminators
263                     thisterm,                   { -- use this read terminators
264                     normterm )                  { -- use normal terminators
265                 EQUALS 0 INCREMENT 1 TAG "c_sr";
266                 CONSTANT prevterm EQUALS ctp$c_sr_prevterm@#shift TAG "m_sr";
267                 CONSTANT thisterm EQUALS ctp$c_sr_thisterm@#shift TAG "m_sr";
268                 CONSTANT normterm EQUALS ctp$c_sr_normterm@#shift TAG "m_sr";
269             sr_noescape  BITFIELD MASK;         /* - don't recognize escape
270             sr_escape    BITFIELD MASK;         /* - recognize escape
271
```



```
272             /* VMS specific bits follow
273
274             sr_noedit    BITFIELD MASK;         /* - disable editing
275             sr_norecall  BITFIELD MASK;         /* - disable recall
276
277           end sr_flag_bits;
278         end sr_flags_overlay;
279
280         sr_max_len      WORD;                   /* max length of read
281         sr_end_data     WORD;                   /* end of data in read buffer
282         sr_timeout      WORD;                   /* timeout value
283         sr_end_prmt     WORD;                   /* end of prompt
284         sr_str_disp     WORD;                   /* start of display
285         sr_lo_water     WORD;                   /* low water mark
286
287         TWO_READS structure;
288             TWO_READS_OVERLAY union fill;
289
290                 sr_term         CHARACTER LENGTH 1;     /* termination set (byte counted field)
```

```
291
292                 /* read data starting position (after term set)
293
294                 CONSTANT len    EQUALS . TAG "c_sr";        /* length of structure
295                 CONSTANT msglen EQUALS .-#header TAG "c_sr";/* length of structure minus header
296                 CONSTANT prolen EQUALS .-#header2 TAG "c_sr";/* length of structure minus header
297
298             READ_VERIFY structure ;
299
300                 sr2_altechsize  WORD UNSIGNED;          /* alt echo size
301                 sr2_picstrsize  WORD UNSIGNED;          /* picture string size
302                 sr2_editflags   WORD UNSIGNED;          /* flags
303                 sr2_fillchar    WORD UNSIGNED;          /* fill characters
304                 sr2_term        CHARACTER LENGTH 1;     /* terminator set
305
306                 CONSTANT len    EQUALS . TAG "c_sr2";         /* length of structure
307                 CONSTANT msglen EQUALS .-#header TAG "c_sr2";  /* length of structure minus header
308                 CONSTANT prolen EQUALS .-#header2 TAG "c_sr2"; /* length of structure minus header
309             end READ_VERIFY;
310             end TWO_READS_OVERLAY;
311         end TWO_READS;
312
313     END start_rd;
314
```



```
315 /*
316 /* read data structure (H <--- S)
317 /*
318     read_data STRUCTURE;
319         rd_flags_overlay union fill;    /* Flags for unread
320           rd_flags byte unsigned;
321           rd_flag_bits structure fill;
322             rd_com_code BITFIELD LENGTH 4;      /* - completion code
323                 CONSTANT (
324                     normal,                     { -- normal terminator
325                     valesc,                     { -- valid escape
326                     invesc,                     { -- invalid escape
327                     outband,                    { -- out of band
328                     inpfull,                    { -- input buffer full
329                     timeout,                    { -- read timed out
330                     unread,                     { -- unread request received
331                     underflo,                   { -- underflow
332                     abstoken,                   { -- absentee token
333                     vert_cng,                   { -- veritcal postition change
334                     lineBrk,                    { -- line break
335                     framerr,                    { -- frame error
336                     parity,                     { -- parity error
337                     overun )                    { -- receiver over-run
338                 EQUALS 0 INCREMENT 1 TAG "m_rd";
339             rd_mor_data BITFIELD MASK;          /* - more data in typeahead
340           end rd_flag_bits;
341         end rd_flags_overlay;
342         rd_lo_water     WORD;                   /* low water
343         rd_vert_cng     BYTE;                   /* vertical change since read started
344         rd_curs_pos     BYTE;                   /* cursor position from EOL
345         rd_term_pos     WORD;                   /* position of terminator
346         rd_data         CHARACTER LENGTH 0;     /* start of read data
347
348         CONSTANT len    EQUALS . TAG "c_rd";        /* length of structure
349         CONSTANT msglen EQUALS .-#header TAG "c_rd"; /* length of structure minus header
350         CONSTANT prolen EQUALS .-#header2 TAG "c_rd"; /* length of structure minus header
```

```
351
352         END read_data;
353
                                                    15-SEP-1984 23:07:19.39     SDL V2.0                                Page   10
                                                    15-SEP-1984 22:49:32        _$255$DUA28:[RTPAD.SRC]RTDEF.SDL;1
354     /*
355     /* out of band structure (H <--- S)
356     /*
357         out_band STRUCTURE;
358
359             ob_flags_overlay union fill;    /* Flags for unread
360               ob_flags byte unsigned;
361               ob_flag_bits structure fill;
362                 ob_linebrk  BITFIELD MASK;          /* - line break occured
363               end ob_flag_bits;
364             end ob_flags_overlay;
365
366             ob_char         BYTE;                   /* one byte of data
367
368             CONSTANT len    EQUALS . TAG "c_ob";        /* length of structure
369             CONSTANT msglen EQUALS .-#header TAG "c_ob"; /* length of structure minus header
370             CONSTANT prolen EQUALS .-#header2 TAG "c_ob"; /* length of structure minus header
371
372         END out_band;
373
                                                    15-SEP-1984 23:07:19.39     SDL V2.0                                Page   11
                                                    15-SEP-1984 22:49:32        _$255$DUA28:[RTPAD.SRC]RTDEF.SDL;1
374     /*
375     /*  unread structure (H ---> S)
376     /*
377         unread STRUCTURE;
378
379             ur_flags_overlay union fill;    /* Flags for unread
380               ur_flags byte unsigned;
381               ur_flag_bits structure fill;
382                 ur_cond     BITFIELD MASK;          /* - unread conditional
383               end ur_flag_bits;
384             end ur_flags_overlay;
385
386             CONSTANT len    EQUALS . TAG "c_ur";        /* length of structure
387             CONSTANT msglen EQUALS .-#header TAG "c_ur"; /* length of structure minus header
388             CONSTANT prolen EQUALS .-#header2 TAG "c_ur"; /* length of structure minus header
389
390         END unread;
391
                                                        15-SEP-1984 23:07:19.39     SDL V2.0                            Page   12
                                                        15-SEP-1984 22:49:32        _$255$DUA28:[RTPAD.SRC]RTDEF.SDL;1
392     /*
393     /* clear input structure (H ---> S)
394     /*
395         clr_input STRUCTURE;
396             ci_flags        BYTE;                   /* no flags defined
397
398             CONSTANT len    EQUALS . TAG "c_ci";        /* length of structure
399             CONSTANT msglen EQUALS .-#header TAG "c_ci"; /* length of structure minus header
400             CONSTANT prolen EQUALS .-#header2 TAG "c_ci"; /* length of structure minus header
401
402         END clr_input;
403
                                                        15-SEP-1984 23:07:19.39     SDL V2.0                            Page   13
```

```
404   /*
405   /* write structure (H ---> S)
406   /*
407       write STRUCTURE;
408           wr_flags_overlay union fill;    /* Flags for write
409             wr_flags word unsigned;
410             wr_flag_bits structure fill;
411               wr_lock     BITFIELD LENGTH 2;      /* - locking
412                   CONSTANT (
413                       noaction,                   { -- no locking action
414                       before,                     { -- lock before, leave locked
415                       befaft,                     { -- lock before, unlock after
416                       befaftre )                  { -- lock before, unlock after, redisplay
417                   EQUALS 0 INCREMENT 1 TAG "m_wr";
418               wr_newline  BITFIELD MASK;          /* - VMS specific, newline modifier
419               wr_discard  BITFIELD MASK;          /* - cancel ^O
420               wr_begin    BITFIELD MASK;          /* - beginning of write
421               wr_end      BITFIELD MASK;          /* - end of write
422               #shift = ^;
423               wr_prefix   BITFIELD LENGTH 2;      /* - prefix code
424                   CONSTANT (
425                       no_fix,                     { -- no prefix
426                       newlinecnt,                 { -- new line count
427                       char )                      { -- character prefix
428                   EQUALS 0 INCREMENT 1 TAG "c_wr";
429                   CONSTANT no_fix  EQUALS ctp$c_wr_no_fix@#shift  TAG "m_wr";
430                   CONSTANT newlinecnt EQUALS ctp$c_wr_newlinecnt@#shift TAG "m_wr";
431                   CONSTANT char    EQUALS ctp$c_wr_char@#shift    TAG "m_wr";
432               wr_postfix  BITFIELD LENGTH 2;      /* - postfix code
433               wr_status   BITFIELD MASK;          /* - return status
434               wr_transparent BITFIELD MASK;       /* - write passall
435             END wr_flag_bits;
436           END wr_flags_overlay;
437           wr_prefix       BYTE;                   /* prefix value
438           wr_postfix      BYTE;                   /* postfix value
439           wr_data         CHARACTER LENGTH 0;     /* start of data
440
441           CONSTANT len    EQUALS . TAG "c_wr";        /* length of structure
442           CONSTANT msglen EQUALS .-#header TAG "c_wr"; /* length of structure minus header
443           CONSTANT prolen EQUALS .-#header2 TAG "c_wr"; /* length of structure minus header
444
445       END write;
446
```



```
447   /*
448   /* write completion structure (H <--- S)
449   /*
450       write_com STRUCTURE;
451           wc_flags_overlay union fill;    /* Flags for unread
452             wc_flags byte unsigned;
453             wc_flag_bits structure fill;
454               wc_discard  BITFIELD MASK;          /* - discard state
455             end wc_flag_bits;
456           end wc_flags_overlay;
457
458           wc_horpos       WORD;                   /* horizontal position
459           wc_verpos       WORD;                   /* vertical position
460
461           CONSTANT len    EQUALS . TAG "c_wc";        /* length of structure
```

```
462         CONSTANT msglen EQUALS .-#header TAG "c_wc"; /* length of structure minus header
463         CONSTANT prolen EQUALS .-#header2 TAG "c_wc"; /* length of structure minus header
464
465     END write_com;
466
```



```
467 /*
468 /* discard state structure (H <--- S)
469 /*
470     dis_state STRUCTURE;
471         ds_flags_overlay union fill;    /* Flags for unread
472           ds_flags_byte unsigned;
473           ds_flag_bits structure fill;
474             ds_discard  BITFIELD MASK;          /* - discard state
475           end ur_flag_bits;
476         end ur_flags_overlay;
477
478         CONSTANT len    EQUALS . TAG "c_ds";        /* length of structure
479         CONSTANT msglen EQUALS .-#header TAG "c_ds"; /* length of structure minus header
480         CONSTANT prolen EQUALS .-#header2 TAG "c_ds"; /* length of structure minus header
481
482     END dis_state;
483
```



```
484 /*
485 /* read characteristics structure (H ---> S)
486 /*
487     read_char STRUCTURE;
488         rc_flags        BYTE;                   /* no flags defined
489         rc_selector     WORD;                   /* selector start position
490
491         CONSTANT len    EQUALS . TAG "c_rc";        /* length of structure
492         CONSTANT msglen EQUALS .-#header TAG "c_rc"; /* length of structure minus header
493         CONSTANT prolen EQUALS .-#header2 TAG "c_rc"; /* length of structure minus header
494
495     END read_char;
496
```



```
497 /*
498 /* characteristics structure (H <---> S)
499 /*
500     char STRUCTURE;
501         ch_flags        BYTE;                   /* no flags defined
502         ch_param        WORD;                   /* start of characteristics
503         ch_value        CHARACTER LENGTH 0;     /* value
504
505         CONSTANT len    EQUALS . TAG "c_ch";        /* length of structure
506         CONSTANT msglen EQUALS .-#header TAG "c_ch"; /* length of structure minus header
507         CONSTANT prolen EQUALS .-#header2 TAG "c_ch"; /* length of structure minus header
508
509         /* characteristics selector types
510
511         CONSTANT (
512                 physical,
513                 logical,
514                 cterm
515                 ) EQUALS 0 INCREMENT 1 PREFIX "CH$" TAG C;
```

```
516
517         /* characteristics selectors, type = physical
518
519         CONSTANT (
520                 in_speed,
521                 out_speed,
522                 char_size,
523                 parity_enable,
524                 parity_type,
525                 modem_present,
526                 autobaud,
527                 manage_guar,
528                 swchar1,
529                 swchar2,
530                 eightbit,
531                 manage_ena
532                 ) EQUALS 1 INCREMENT 1 PREFIX "CH$" TAG "C_PH";
533
534         /* characteristics selectors, type = logical
535
536         CONSTANT (
537                 mode_writing,
538                 term_bits,
539                 term_type,
540                 output_flow,
541                 page_stop,
542                 flow_char_pass,
543                 input_flow,
544                 loss_notif,
545                 line_width,
546                 page_length,
547                 stop_length,
548                 cr_fill,
549                 lf_fill,
550                 wrap,
551                 hor_tab,
552                 vert_tab,
553                 form_feed
```



```
554                 ) EQUALS 1 INCREMENT 1 PREFIX "CH$" TAG "C_LG";
555
556
557         /* characteristics selectors, type = cterm
558
559         CONSTANT (
560                 ignore_input,
561                 char_att,               /* see tty_attributes, etc. below
562                 ctrlo_pass,
563                 raise_input,
564                 normal_echo,
565                 input_esc,
566                 output_esc,
567                 input_count,
568                 auto_prompt,
569                 error_processing
570                 ) EQUALS 1 INCREMENT 1 PREFIX "CH$" TAG "C_CT";
571
572         CONSTANT (
573                 even,
574                 odd,
575                 clear,          /* no support for set or clear on VMS
```

```
576             set
577             ) EQUALS 1 INCREMENT 1 PREFIX "CH$" TAG C_PARITY;
578
579         tty_attributes STRUCTURE;
580             ch_known    BITFIELD MASK;
581             ch_scope    BITFIELD MASK;
582         end tty_attributes;
583
584         oob_handling STRUCTURE;
585             ch_oo       bitfield mask length 2; /* out of band handling
586             ch_i        bitfield mask;          /* include character
587             ch_d        bitfield mask;          /* discard output
588             ch_ee       bitfield mask length 2; /* echo control
589             ch_f        bitfield mask;          /* special enable
590         end oob_handling;
591
592         oob_data STRUCTURE;
593             ch_char     BYTE;           /* data character
594             ch_mask     BYTE;           /* mask for attributes
595             ch_attr     BYTE;           /* attributes
596         end oob_data;
597
598         CONSTANT (                      { - out of band flags
599             cancel,                     { -- cancel previous    0
600             iclear,                     { -- immediate clear    1
601             dclear,                     { -- deferred clear     2
602             hello )                     { -- hello              3
603         EQUALS 0 INCREMENT 1 TAG "c_ch";
604
605         CONSTANT (                      { echo control
606             echonone,                   { -- don't echo
607             echoself,                   { -- echo character as self
608             echostandard,               { -- standard echo
609             echoboth )                  { -- echo both
610         EQUALS 0 INCREMENT 1 TAG "c_ch";
611
```



```
612         END char;
613
```



```
614     /*
615     /* check input structure (H ---> S)
616     /*
617         check_inp STRUCTURE;
618             ck_flags        BYTE;                   /* no flags defined
619
620             CONSTANT len    EQUALS . TAG "c_ck";        /* length of structure
621             CONSTANT msglen EQUALS .-#header TAG "c_ck"; /* length of structure minus header
622             CONSTANT prolen EQUALS .-#header2 TAG "c_ck"; /* length of structure minus header
623
624         END check_inp;
625
```



```
626     /*
627     /* input count structure (H <--- S)
628     /*
629         inp_count STRUCTURE;
```

```
630         ic_flags        BYTE;                   /* no flags defined
631         ic_count        WORD;                   /* input count
632
633         CONSTANT len    EQUALS . TAG "c_ic";        /* length of structure
634         CONSTANT msglen EQUALS .-#header TAG "c_ic"; /* length of structure minus header
635         CONSTANT prolen EQUALS .-#header2 TAG "c_ic"; /* length of structure minus header
636
637     END inp_count;
638
```



```
639 /*
640 /* input state structure (H <--- S)
641 /*
642     inp_state STRUCTURE;
643         is_flags_overlay union fill;    /* Flags for unread
644           is_flags_byte unsigned;
645           is_flag_bits structure fill;
646             is_nonzero  BITFIELD MASK;          /* - count change to non-zero
647           end is_flag_bits;
648         end is_flags_overlay;
649
650         CONSTANT len    EQUALS . TAG "c_is";        /* length of structure
651         CONSTANT msglen EQUALS .-#header TAG "c_is"; /* length of structure minus header
652         CONSTANT prolen EQUALS .-#header2 TAG "c_is"; /* length of structure minus header
653
654     END inp_state;
655
656 /*
657 /* VMS QIO REQUEST (H ---> S)
658 /*
659     vmsqio STRUCTURE;
660         vms_flags_overlay union fill;   /* Flags for unread
661           vms_flags_byte unsigned;
662           vms_flag_bits structure fill;
663             vms_useiosb BITFIELD MASK;          /* use iosb to determine length
664             vms_readlen BITFIELD MASK;          /* - read-type iosb buffer length
665           end vms_flag_bits;
666         end vms_flags_overlay;
667
668         vms_reqid       LONGWORD;               /* qio request id
669
670         vmsfields UNION;
671
672             VMSREQ STRUCTURE;                   /* request
673                 vms_func        WORD;               /* qio function code
674                 vms_plen        WORD;               /* these four are repeated...
675                 vms_pcode       WORD;               /*   for each parameter
676                 vms_pflags      STRUCTURE TAG W;    /*
677                     vms_ref     BITFIELD MASK;      /* - pass by reference
678                     vms_item    BITFIELD MASK;      /* - item list or Pn
679                     vms_buffer  BITFIELD MASK;      /* - this is return buffer
680                     vms_fill2   BITFIELD LENGTH 16-^;   /* - fill to 1 word
681                     END vms_pflags;
682                 vms_pdata       CHARACTER LENGTH 0;     /* value
683
684                 CONSTANT len    EQUALS . TAG "c_vms";       /* length of structure
685                 CONSTANT msglen EQUALS .-#header TAG "c_vms"; /* length of structure minus header
686                 CONSTANT prolen EQUALS .-#header2 TAG "c_vms"; /* length of structure minus header
687             END VMSREQ;
688
689             VMSRESP STRUCTURE;
```

```
690                 vms_iosb        QUADWORD;                   /* iosb
691                 vms_rdata       CHARACTER LENGTH 0;         /* start of data
692             END VMSRESP;
693
694         END vmsfields;
695
```



```
696     END vmsqio;
697
```



```
698 /*
699 /* upline broadcast (H <--- S)
700 /*
701     broadcast STRUCTURE;
702         br_flags        WORD UNSIGNED;          /* no flags defined
703         br_msgcode      WORD UNSIGNED;          /* mailbox message code
704         br_unitnum      WORD UNSIGNED;          /* unit number
705         br_devname      CHARACTER LENGTH 16;    /* device name
706         br_msglen       WORD UNSIGNED;          /* length of text
707         br_msgtxt       CHARACTER LENGTH 0;     /* start of text
708
709         CONSTANT len    EQUALS . TAG "c_br";        /* length of structure
710         CONSTANT msglen EQUALS .-#header TAG "c_br"; /* length of structure minus header
711         CONSTANT prolen EQUALS .-#header2 TAG "c_br"; /* length of structure minus header
712
713     END broadcast;
714
715 END msgfields; /* end of protocol messages
716
717 END cterm;
718
719
```



```
720 AGGREGATE VMSQIO STRUCTURE PREFIX vms$;
721
722         plen    WORD;                   /* these four are repeated...
723         pflags  WORD;                   /*   ...
724         pcode   WORD;                   /*   ... for each parameter
725         pdata   CHARACTER LENGTH 0;     /* value
726
727 END; /* VMSQIO
728
729 END_MODULE;
```

RTMSG
LIS

RSXRT
LIS

RTDEF
LIS

RTDTE
LIS

RTPAD
LIS

RTLOGIO
LIS