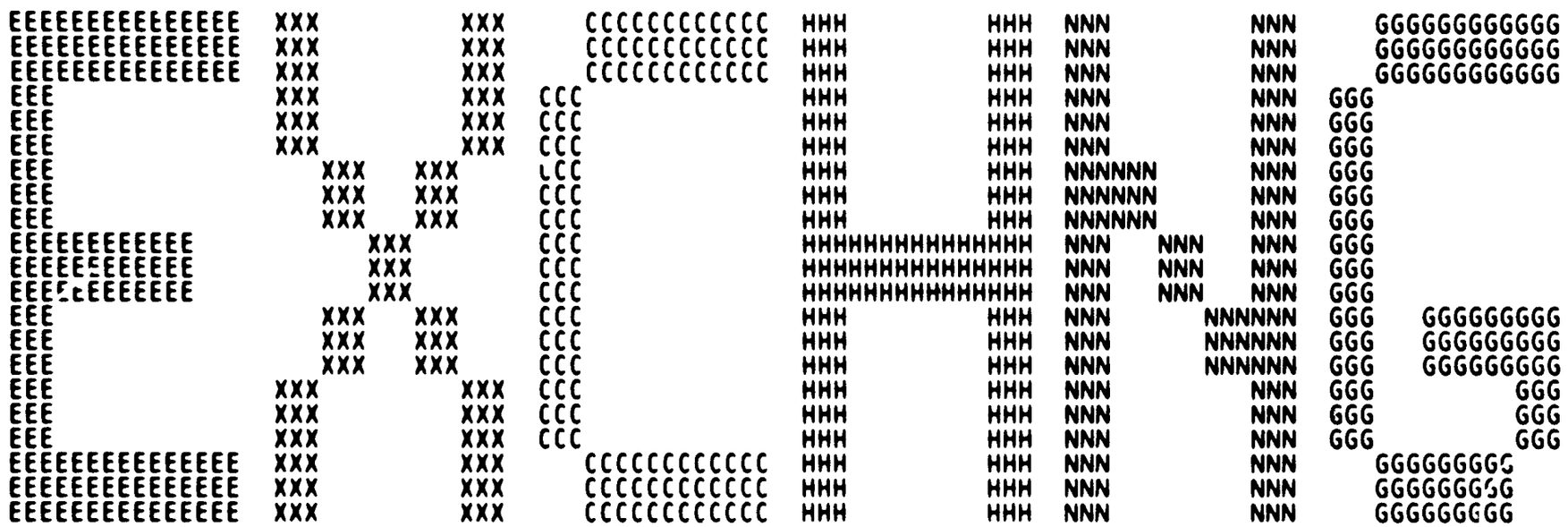
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNN NNN GGG
EEEEEEEEEEEE XXX CCC HHHHHHHHHHHHHHH NNN NNN NNN GGG
EEEEEEEEEEEE XXX CCC HHHHHHHHHHHHHHH NNN NNN NNN GGG
EEEEEEEEEEEE XXX CCC HHHHHHHHHHHHHHH NNN NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG

FILEID**EXCDEFS

```
EEEEEEEEEE  XX      XX    CCCCCCCC  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    SSSSSSSS
EEEEEEEEEE  XX      XX    CCCCCCCC  DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    SSSSSSSS
EE          XX      XX  CC          DD      DD  EE          FF          SS
EE          XX      XX  CC          DD      DD  EE          FF          SS
EE            XX  XX    CC          DD      DD  EE          FF          SS
EE            XX  XX    CC          DD      DD  EE          FF          SS
EEEEEEEE        XX      CC          DD      DD  EEEEEEEE    FFFFFFFF      SSSSSS
EEEEEEEE        XX      CC          DD      DD  EEEEEEEE    FFFFFFFF      SSSSSS
EE            XX  XX    CC          DD      DD  EE          FF                SS
EE            XX  XX    CC          DD      DD  EE          FF                SS
EE          XX      XX  CC          DD      DD  EE          FF                SS      ....
EE          XX      XX  CC          DD      DD  EE          FF                SS      ....
EEEEEEEEEE  XX      XX    CCCCCCCC  DDDDDDDD    EEEEEEEEEE  FF          SSSSSSSS      ....
EEEEEEEEEE  XX      XX    CCCCCCCC  DDDDDDDD    EEEEEEEEEE  FF          SSSSSSSS      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
1   {****************************************************************************
2   {*                                                                          *
3   {*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
4   {*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
5   {*  ALL RIGHTS RESERVED.                                                    *
6   {*                                                                          *
7   {*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
8   {*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
9   {*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
10  {*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
11  {*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY      *
12  {*  TRANSFERRED.                                                            *
13  {*                                                                          *
14  {*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE    *
15  {*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT        *
16  {*  CORPORATION.                                                            *
17  {*                                                                          *
18  {*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS     *
19  {*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
20  {*                                                                          *
21  {*                                                                          *
22  {****************************************************************************
23
24  MODULE $exchdef;
25  /*
26  /* Version:     'V04-000'
27  /*
28
29  /*++
30  /*
31  /* FACILITY:    EXCHANGE - Foreign volume interchange utility
32  /*
33  /* ABSTRACT:    Data structure definitions for EXCHANGE.
34  /*
35  /* ENVIRONMENT: VAX/VMS operating system, unprivileged user-mode utility
36  /*
37  /* AUTHOR:      CW Hobbs
38  /*
39  /* DATE:        12-July-1982    Last Edit:
40  /*
41  /* MODIFIED BY:
42  /*
43  /*
44  /*--
45
```



```
46  /++
47  /*
48  /* Local constants which define the sizes of adjustable structures in $EXCHDEF
49  /*
50  /*-
51
52  #io_buf_blocks = 12;                    /* must be 5 or more, at least 2 blocks
53                                          /*  larger than rec_buf_size, and LEQU 127
54  #rec_buf_size = 512;
55
56
```

```
57  /**
58  /*
59  /* DMPDSV - Flag options for EXCH$DBG_DUMP_DOSV debugging routine
60  /*
61  /*-
62  AGGRFGATE dmpdsv STRUCTURE PREFIX dmpdsv$;
63      dmpdsv STRUCTURE BYTE UNSIGNED;
64          volb            BITFIELD MASK;          /* print volb and dosv addresses
65          status          BITFIELD MASK;          /* print status flags
66          position        BITFIELD MASK;          /* print current position
67          entries         BITFIELD MASK;          /* print the directory entries
68          END dmpdsv;
69  END dmpdsv;
70
71
72  /**
73  /*
74  /* DOSNXT - Flag options for EXCH$DOS11_NEXT_ENTRY
75  /*
76  /*-
77  AGGREGATE dosnxt STRUCTURE PREFIX dosnxt$;
78      dosnxt STRUCTURE BYTE UNSIGNED;
79          rewind          BITFIELD MASK;          /* "rewind" to the start of the tape
80          count_blocks    BITFIELD MASK;          /* make sure that the block count is valid
81          END dosnxt;
82  END dosnxt;
83
84
85  /**
86  /*
87  /* EXCHBLK - Constant definitions for block type codes
88  /*
89  /*-
90  CONSTANT
91          (
92          copy,                                   /* work area for COPY and TYPE verbs
93          dire,                                   /* DIRECTORY verb work area
94          dos11,                                  /* volume specifics for DOS-11
95          dos11ctx,                               /* DOS-11 file context block
96          excg,                                   /* global data structure
97          filb,                                   /* file context block
98          init,                                   /* INIT verb work area
99          moun,                                   /* MOUNT verb work area
100         namb,                                   /* name block
101         rmsb,                                   /* exchblk$k_rmsb - code for file block
102         rt11,                                   /* volume specifics for RT-11

                                                        15-SEP-1984 23:01:31.30     SDL V2.0                        Page    3
                                                        15-SEP-1984 22:44:23        _$255$DUA28:[EXCHNG.SRC]EXCDEFS.SDL;1
103         rt11ctx,                                /* RT-11 file context block
104         volb,                                   /* volume block
105         rtnam                                   /* RT-11 name routine work area
106         )
107         EQUALS 255 INCREMENT -1 PREFIX exchblk$;
108
109
110 /**
111 /*
112 /* FINDBIN - Return status codes for PDP_FIND_BINARY_RECORD
113 /*
114 /*-
115 CONSTANT
116         (
```

```
117             success,                                /* found a record
118             eob,                                    /* end of buffer, no record found
119             chksum,                                 /* checksum failed to match
120             too_big,                                /* record larger than output buffer
121             bad_fmt                                 /* badly formed record
122             )
123             EQUALS 0 INCREMENT 1 PREFIX findbin$;
124
125     CONSTANT lobound EQUALS findbin$k_success PREFIX findbin$;
126     CONSTANT hibound EQUALS findbin$k_bad_fmt PREFIX findbin$;
127
128
129     /**
130     /*
131     /* FINDSTM - Return status codes for PDP_FIND_STREAM_RECORD
132     /*
133     /*-
134     CONSTANT
135             (
136             success,                                /* found a record
137             ctrlz_eof,                              /* ^Z at beginning of record
138             eob,                                    /* end of buffer, no record found
139             no_term,                                /* end of buffer, partial record found
140             bad_fmt                                 /* record longer than output buffer
141             )
142             EQUALS 0 INCREMENT 1 PREFIX findstm$;
143
144     CONSTANT lobound EQUALS findstm$k_success PREFIX findstm$;
145     CONSTANT hibound EQUALS findstm$k_bad_fmt PREFIX findstm$;
146
147
148     /**
149     /*
150     /* PRSOPT - Flag options for EXCH$CMD_PARSE_FILESPEC
151     /*
152     /*-
153     AGGREGATE prsopt STRUCTURE PREFIX prsopt$;
154         prsopt STRUCTURE BYTE UNSIGNED;
155             virtual_device  BITFIELD MASK;          /* parse device name as a virtual device
156             no_get_value    BITFIELD MASK;          /* skip CLI$GET_VALUE call, use parameter name as value
157             END prsopt;
158     END prsopt;
159
160
```



```
161     /**
162     /*
163     /* RTNXT - Flag options for EXCH$RTACP_NEXT_ENTRY
164     /*
165     /*-
166     AGGREGATE rtnxt STRUCTURE PREFIX rtnxt$;
167         rtnxt STRUCTURE BYTE UNSIGNED;
168             permanent       BITFIELD MASK;          /* look for permanent entries
169             empty           BITFIELD MASK;          /* look for empty entries
170             tentative       BITFIELD MASK;          /* look for tentative entries
171             unknown         BITFIELD MASK;          /* look for invalid entries
172             skip_check      BITFIELD MASK;          /* skip the check for moved entry
173             skip_expand     BITFIELD MASK;          /* skip expanding the radix-50 name to ascii
174             END rtnxt;
175     END rtnxt;
176
```

```
177   /*+
178   /*
179   /* $COPY - copy block
180   /*
181   /* Contains the work area for the COPY and TYPE verbs
182   /*
183   /*-
184
185   AGGREGATE copy STRUCTURE PREFIX copy$;
186   /*
187   /* Dynamic character string descriptors for name components.  These strings are reclaimed at the start of each copy
188   /*
189       default_filename    QUADWORD UNSIGNED;      /* desc for sticky name
190   /*
191       size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_copy)
192       type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_copy)
193       std_fill            BYTE FILL;              /* unused
194   /*
195       input_filename      QUADWORD UNSIGNED;      /* desc for current item from input parameter list
196       output_filename     QUADWORD UNSIGNED;      /* desc for output parameter
197   /*
198       input_sticky_name   QUADWORD UNSIGNED;      /* name to make input files sticky
199   /*
200   {\  q_boot              QUADWORD UNSIGNED;      /* qualifier /BOOT=string
201   {\  q_fdl               QUADWORD UNSIGNED;      /* qualifier /FDL=string
202   /*
203       CONSTANT start_zero EQUALS .;               /* here to end is cleared for each copy command
204   /*
205   /* Integer valued command qualifiers
206   /*
207       q_allocation        LONGWORD UNSIGNED;      /* qualifier /ALLOCATION=n                    (optional)
208       q_extension         LONGWORD UNSIGNED;      /* qualifier /EXTENSION=n                     (optional)
209       q_start_block       LONGWORD UNSIGNED;      /* qualifier /START_BLOCK=n
210   /*
211   /* Flags for boolean DCL command qualifiers
212   /*
213       qual_flags STRUCTURE LONGWORD UNSIGNED;     /* the whole flags longword
214           q_best_try_contiguous   BITFIELD MASK;
215   {\      q_confirm               BITFIELD MASK;
216           q_contiguous            BITFIELD MASK;
217           q_delete                BITFIELD MASK;
218           q_log                   BITFIELD MASK;
219           q_nolog_explicit        BITFIELD MASK;  /* we had an explicit /NOLOG
220           q_protect               BITFIELD MASK;  /* value for protect bit, not useful unless explicit also set
221           q_protect_explicit      BITFIELD MASK;  /* an explicit /PROTECT or /NOPROTECT was seen
222           q_replace               BITFIELD MASK;
223           q_rewind                BITFIELD MASK;
224           q_system                BITFIELD MASK;
225           q_truncate              BITFIELD MASK;
226           END qual_flags;
227   /*
228   /* Local copy command flags
229   /*
230       copy_flags STRUCTURE LONGWORD UNSIGNED;     /* the whole flags longword
231           multiple_files  BITFIELD MASK;          /* the input potentially describes multiple files
232           type_command    BITFIELD MASK;          /* the command is TYPE rather than COPY
233           reopen_input    BITFIELD MASK;          /* reopen previous input file (due to RT-11 bad file)
```

```
234         reopen_in_progress BITFIELD MASK;       /* current file has been reopened
235         END copy_flags;
236     /*
237     /* Other misc data items
238     /*
239         max_rec             LONGWORD UNSIGNED;  /* current maximum output length
240     /*
241     /* Work area for the current input file
242     /*
243         inp_filb            ADDRESS;            /* pointer to a filb for the input file
244         inp_namb            ADDRESS;            /* pointer to a namb describing the input file
245     /*
246     /* Work area for the current output file
247     /*
248         out_filb            ADDRESS;            /* pointer to a filb for the output file
249         out_namb            ADDRESS;            /* pointer to a namb describing the output file
250     /*
251         CONSTANT end_zero EQUALS .;             /* no need to clear the buffers (or create pages not to be used)
252         CONSTANT "length" EQUALS .;             /* length of data structure
253
254     END copy;
255
```



```
256     /*
257     /* Generic file context block.  This is the front part of both the RT11CTX and DOS11CTX blocks, and is used
258     /* by the PDP record routines shared by both RT-11 and DOS-11 formats
259     /*
260
261     AGGREGATE ctx STRUCTURE PREFIX ctx$;
262     /*
263     /* Links to the queues of in-use and available  blocks, plus standard type and size fields
264     /*
265         header STRUCTURE QUADWORD UNSIGNED;      /* queue header
266             flink   ADDRESS;                    /* forward link
267             blink   ADDRESS;                    /* reverse link
268             END link;
269     /*
270         size                WORD UNSIGNED;      /* standard size of the block   (symbol EXCHBLK$S_dos11ctx)
271         type                BYTE UNSIGNED;      /* type code field              (symbol EXCHBLK$K_dos11ctx)
272         std_fill            BYTE FILL;          /* unused
273     /*
274         alloc               ADDRESS;            /* link to list of allocated blocks
275     /*
276     /* Addresses of associated control structures
277     /*
278         assoc_filb          ADDRESS;            /* address of the filb
279         assoc_volb          ADDRESS;            /* address of the volb
280     /*
281     /* I/O buffer for this record stream
282     /*
283         buffer              ADDRESS;            /* Pointer to the buffer
284     /*
285         CONSTANT buffer_blocks EQUALS #io_buf_blocks;       /* size of the buffer in blocks
286         CONSTANT buffer_length EQUALS #io_buf_blocks*512;   /* size of the buffer in bytes
287     /*
288         CONSTANT start_zero EQUALS .;           /* here to end is cleared for each copy command
289     /*
290     /* Data to describe a record stream in the file.
291     /*
292         cur_block           LONGWORD UNSIGNED;  /* pbn of current block
293         eof_block           LONGWORD UNSIGNED;  /* pbn of last block in file
```

```
294         cur_byte            LONGWORD UNSIGNED;      /* offset into the block
295     /*
296     /* Flags for an open file.  This longword will be zero when file is not open
297     /*
298         flags STRUCTURE LONGWORD UNSIGNED;          /* the whole flags longword
299             stream_active   BITFIELD MASK;          /* block currently describes an open file
300             output_file     BITFIELD MASK;          /* block currently describes an output file
301             flush           BITFIELD MASK;          /* tell advance routine to flush last block
302             pointer         BITFIELD MASK;          /* tape position is at this file
303             END flags;
304     /*
305     /* Pointers to the I/O buffer for this record stream
306     /*
307         buf_base_block      LONGWORD UNSIGNED;      /* Pbn of first block in buffer
308         buf_high_block      LONGWORD UNSIGNED;      /* Pbn of last block in buffer
309         high_block_written  LONGWORD UNSIGNED;      /* Pbn of highest block to the file
310     /*
311         CONSTANT "length" EQUALS .;                 /* length of data structure
312
```



```
313     END ctx;
314
```



```
315     /**
316     /*
317     /* $DIRE - Directory block
318     /*
319     /* Contains the work area for the DIRECTORY verb
320     /*
321     /*-
322
323     AGGREGATE dire STRUCTURE PREFIX dire$;
324     /*
325         filename            QUADWORD UNSIGNED;      /* file name parameter to command
326     /*
327         size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_dire)
328         type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_dire)
329         std_fill            BYTE FILL;              /* unused
330     /*
331     /* Dynamic character string descriptors for name components
332     /*
333         output_file         QUADWORD UNSIGNED;      /* desc for /OUTPUT file name
334         default_name        QUADWORD UNSIGNED;      /* desc for sticky name
335     /*
336         CONSTANT start_zero EQUALS .;               /* here to end is cleared for each directory command
337     /*
338     /* Some pointers to RMS structures
339     /*
340         outrmsb             ADDRESS;                /* A $RMSB structure for the /OUTPUT file block
341         outfab              ADDRESS;                /* Ref to fab for output
342         outrab              ADDRESS;                /* Ref to rab for output
343     /*
344     /* Integer valued command qualifiers
345     /*
346         q_columns           LONGWORD UNSIGNED;      /* qualifier /COLUMNS=n (always present)
347     /*
348     /* Command qualifier flags
349     /*
350         qual_flags STRUCTURE LONGWORD UNSIGNED;     /* the whole flags longword
```

```
351             q_all           BITFIELD MASK;          /* qualifier /ALL
352     {\      q_badblocks     BITFIELD MASK;          /* qualifier /BADBLOCKS
353             q_blocks        BITFIELD MASK;          /* qualifier /BLOCKS
354             q_brief         BITFIELD MASK;          /* qualifier /BRIEF
355             q_date          BITFIELD MASK;          /* qualifier /DATE
356             q_deleted       BITFIELD MASK;          /* qualifier /DELETED
357             q_free          BITFIELD MASK;          /* qualifier /FREE
358             q_full          BITFIELD MASK;          /* qualifier /FULL
359             q_octal         BITFIELD MASK;          /* qualifier /OCTAL
360             q_output        BITFIELD MASK;          /* qualifier /OUTPUT=<filespec>
361             q_owner         BITFIELD MASK;          /* qualifier /OWNER
362             q_printer       BITFIELD MASK;          /* qualifier /PRINTER
363             q_size          BITFIELD MASK;          /* qualifier /SIZE
364             q_summary       BITFIELD MASK;          /* qualifier /SUMMARY
365             END qual_flags;
366     /*
367     /* Local directory command flags
368     /*
369         dire_flags STRUCTURE LONGWORD;
370             item_to_print   BITFIELD MASK;  /* an item is waiting to be printed
371             items_printed   BITFIELD MASK;  /* one or more items have been printed
```



```
372             END dire_flags;
373     /*
374     /* Other misc data items
375     /*
376         item_width          LONGWORD UNSIGNED;      /* width of a single directory listing item
377         list_width          LONGWORD UNSIGNED;      /* maximum listing width
378         cur_width           LONGWORD UNSIGNED;      /* current listing width
379         cur_column          LONGWORD UNSIGNED;      /* current column in listing buffer
380         list_buffer         CHARACTER LENGTH 512;   /* buffer where listing lines are assembled
381         CONSTANT item_spacing EQUALS 3;
382     /*
383         CONSTANT "length" EQUALS .;                 /* length of data structure
384
385     END dire;
386
```



```
387     /++
388     /*
389     /* $DOS11 - volume specific structure for an DOS-11 volume
390     /*
391     /* Contains all the DOS-11 specific control structures for a volume.  This structure would
392     /* normally hang from VOLB$A_VFMT_SPECIFIC.
393     /*
394     /*-
395
396     AGGREGATE dos11 STRUCTURE PREFIX dos11$;
397         dummy               QUADWORD FILL;          /* filler to put the size and type fields in the right spot
398     /*
399         size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_rmsb)
400         type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_rmsb)
401         std_fill            BYTE FILL;              /* unused
402     /*
403         CONSTANT start_zero EQUALS .;
404     /*
405     /* Variables to describe the state of the DOS-11 device go here
406     /*
407         status STRUCTURE WORD UNSIGNED;             /* complete status word
```

```
408         position_valid  BITFIELD MASK;          /* value in dos11$l_current_file is valid
409         beg_of_tape     BITFIELD MASK;          /* last operation rewound the magtape
410         tape_mark       BITFIELD MASK;          /* magtape read found a tape mark
411         end_of_tape     BITFIELD MASK;          /* magtape read found two consecutive tape marks
412         directory       BITFIELD MASK;          /* magtape "directory" is complete and valid
413         current_marked  BITFIELD MASK;          /* file underneath the heads marked with "->"
414         error_rewind    BITFIELD MASK;          /* tape has been rewound due to error
415         END status;
416 /*
417 /* Remember which file on the tape was last read.  The file at BOT is file #0.  The current_file
418 /* number in conjunction with the position bits marks a location on the tape.
419 /*
420     current_file        LONGWORD UNSIGNED;
421 /*
422 /* Directory entries are linked from here.  This is really just a queue of the labels
423 /*
424     entry_header STRUCTURE QUADWORD UNSIGNED;   /* queue header
425         entry_flink     ADDRESS;                /* forward link
426         entry_blink     ADDRESS;                /* reverse link
427         END entry_header;
428 /*
429 /* The address of the current label for wildcard operations.  Note that this will not necessarily correspond
430 /* to the current_file position.
431 /*
432     current_entry       ADDRESS;                /* address of the current entry in the queue
433 /*
434 /* Various bits and pieces for IO operations on tapes
435 /*
436     iosb STRUCTURE QUADWORD UNSIGNED;
437         iosb_status     WORD UNSIGNED;          /* returned status
438         counts UNION;
439             iosb_skipcnt        WORD;           /* number of files or records skipped (signed)
440             iosb_bytecnt        WORD UNSIGNED;  /* size of transfer
441             END counts;
442         iosb_char       LONGWORD UNSIGNED;      /* characteristics buffer
443         END iosb;
```



```
444 /*
445 /* We need a scratch buffer to read in our tape labels
446 /*
447     label_buf   CHARACTER LENGTH 14;            /* anything over 14 is invalid
448 /*
449 /* The routine EXCH$IO_DOS11_COUNT_BLOCKS stores the block count in the next field
450 /*
451     block_count WORD UNSIGNED;
452 /*
453     CONSTANT end_zero EQUALS .;
454     CONSTANT "length" EQUALS .;                 /* length of data structure
455
456 END dos11;
457
```



```
458 /*
459 /* DOS-11 "directory" entry
460 /*
461
462 AGGREGATE dos11ent STRUCTURE PREFIX dos11ent$;
463 /*
464 /* Links to the rest of the entries for a tape
```

```
465   /*
466       header STRUCTURE QUADWORD UNSIGNED;          /* queue header
467           flink   ADDRESS;                         /* forward link
468           blink   ADDRESS;                         /* reverse link
469           END link;
470   /*
471   /* The entry info will be copied into the dos11ctx entry, put a marker
472   /* so that we can automatically compare the lengths
473   /*
474       CONSTANT ''start_entry'' EQUALS .;
475   /*
476   /* Next seven words map directly onto the 14-byte mag tape label
477   /*
478           filename_1      LONGWORD UNSIGNED;       /* 6 Radix-50 characters
479           filetype        WORD UNSIGNED;           /* 3 Radix-50 characters
480           uic STRUCTURE WORD UNSIGNED;             /* User ID code
481               member      BYTE UNSIGNED;           /* Member number
482               group       BYTE UNSIGNED;           /* Group number
483               END uic;
484           protection      WORD UNSIGNED;           /* Protection code
485           date            WORD UNSIGNED;           /* Creation date field
486           filename_2      WORD UNSIGNED;           /* 3 Radix-50 characters
487   /*
488   /* Rest of fields are computed
489   /*
490           blocks          WORD UNSIGNED;           /* Total length of file
491           file_number     WORD UNSIGNED;           /* Position of file on tape, first file is #0
492   /*
493           entry_status STRUCTURE WORD UNSIGNED;
494               blocks_valid        BITFIELD MASK;   /* The ''BLOCKS'' field is correct
495               END entry_status;
496   /*
497       CONSTANT ''end_entry'' EQUALS .;
498   /*
499       CONSTANT ''length'' EQUALS .;                /* length of data structure
500
501   END dos11ent;
502
```



```
503   /*
504   /* DOS-11 file context block  This is a combination of the FAB, RAB and NAM block
505   /* for an DOS-11 random access file
506   /*
507
508   AGGREGATE dos11ctx STRUCTURE PREFIX dos11ctx$;
509   /*
510   /* Links to the queues of in-use and available  blocks, plus standard type and size fields
511   /*
512       header STRUCTURE QUADWORD UNSIGNED;          /* queue header
513           flink   ADDRESS;                         /* forward link
514           blink   ADDRESS;                         /* reverse link
515           END link;
516   /*
517       size                WORD UNSIGNED;         /* standard size of the block   (symbol EXCHBLK$S_dos11ctx)
518       type                BYTE UNSIGNED;         /* type code field              (symbol EXCHBLK$K_dos11ctx)
519       std_fill            BYTE FILL;             /* unused
520   /*
521       alloc               ADDRESS;               /* link to list of allocated blocks
522   /*
523   /* Addresses of associated control structures
524   /*
```

```
525         assoc_filb          ADDRESS;                /* address of the filb
526         assoc_volb          ADDRESS;                /* address of the volb
527     /*
528     /* I/O buffer for this record stream
529     /*
530         buffer              ADDRESS;                /* Pointer to the buffer
531     /*
532         CONSTANT start_zero EQUALS .;               /* here to end is cleare      each copy command
533     /*
534     /* Data to describe a record stream in the file.
535     /*
536         cur_block           LONGWORD UNSIGNED;      /* pbn of current block
537         eof_block           LONGWORD UNSIGNED;      /* pbn of last block in file
538         cur_byte            LONGWORD UNSIGNED;      /* offset into the block
539     /*
540     /* Flags for an open file.  This longword will be zero when file is not open
541     /*
542         flags STRUCTURE LONGWORD UNSIGNED;          /* the whole flags longword
543             stream_active   BITFIELD MASK;          /* block currently describes an open file
544             output_file     BITFIELD MASK;          /* block currently describes an output file
545             flush           BITFIELD MASK;          /* tell advance routine to flush last block
546             pointer         BITFIELD MASK;          /* tape position is at this file
547             END flags;
548     /*
549     /* Pointers to the I/O buffer for this record stream
550     /*
551         buf_base_block      LONGWORD UNSIGNED;      /* Pbn of first block in buffer
552         buf_high_block      LONGWORD UNSIGNED;      /* Pbn of last block in buffer
553         high_block_written  LONGWORD UNSIGNED;      /* Pbn of highest block to the file
554     /*
555     /* The next part is a duplicate of the directory entry
556     /*
557         current_entry       ADDRESS;                /* the address of this entry in the "directory" queue
558         entry_union UNION;
559             "entry"         CHARACTER LENGTH dos11ent$k_end_entry - dos11ent$k_start_entry; /* The overall entry
```



```
560             entry_fields STRUCTURE;
561     /*
562     /* Next seven words map directly onto the 14-byte mag tape label
563     /*
564                 filename_1          LONGWORD UNSIGNED;      /* 6 Radix-50 characters
565                 filetype            WORD UNSIGNED;          /* 3 Radix-50 characters
566                 uic STRUCTURE WORD UNSIGNED;                /* User ID code
567                     member          BYTE UNSIGNED;          /* Member number
568                     group           BYTE UNSIGNED;          /* Group number
569                     END uic;
570                 protection          WORD UNSIGNED;          /* Protection code
571                 date                WORD UNSIGNED;          /* Creation date field
572                 filename_2          WORD UNSIGNED;          /* 3 Radix-50 characters
573     /*
574     /* Rest of fields are computed
575     /*
576                 blocks              WORD UNSIGNED;          /* Total length of file
577                 file_number         WORD UNSIGNED;          /* Position of file on tape, first file is #0
578     /*
579                 entry_status STRUCTURE WORD UNSIGNED;
580                     blocks_valid    BITFIELD MASK;          /* The "BLOCKS" field is correct
581                     END entry_status;
582                 END entry_fields;
583             END entry_union;
584     /*
```

```
585     /* Expanded name fields
586     /*
587         exp_fullname_len    LONGWORD UNSIGNED;      /* Length of the concatenated name
588         exp_name_len        LONGWORD UNSIGNED;      /* Length of the name field
589         exp_type_len        LONGWORD UNSIGNED;      /* Length of the file extension (type)
590         exp_directory       CHARACTER LENGTH 10;    /* UIC format directory "[000,000]"
591         exp_fullname        CHARACTER LENGTH 13;    /* Name in normal form, no embedded blanks
592         exp_name            CHARACTER LENGTH 9;     /* Blank padded name
593         exp_type            CHARACTER LENGTH 3;     /* Blank padded extension
594         exp_date            CHARACTER LENGTH 11;    /* VMS format ascii date
595     /*
596         CONSTANT end_zero EQUALS .;
597         CONSTANT "length" EQUALS .;                 /* length of data structure
598
599     END dos11ctx;
600
```



```
601     /**
602     /*
603     /* $EXCG - EXCHANGE Global data
604     /*
605     /* Contains all the global data necessary for the EXCHANGE facility
606     /*
607     /*-
608
609     AGGREGATE excg STRUCTURE PREFIX excg$;
610     /*
611     /* Flags for command line qualifiers and local stat
612     /*
613         flags STRUCTURE WORD UNSIGNED;              /* whole flags word
614             control_c       BITFIELD MASK;          /* Control/C AST has been received
615             q_cache         BITFIELD MASK;          /* EXCHANGE /CACHE is in effect for RT-11
616             q_message       BITFIELD MASK;          /* EXCHANGE /MESSAGE is in effect
617             foreign_command BITFIELD MASK;          /* exchange is running in single-command mode
618             exiting         BITFIELD MASK;          /* exit handler is active
619             END flags;
620     /*
621     /* Channel for input terminal, used for setting control/c asts.  0 if input is not a terminal device
622     /*
623         tt_channel          WORD UNSIGNED;
624     /*
625         copy_work           ADDRESS;                /* pointer to the work area for COPY and TYPE
626     /*
627         size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_EXCG)
628         type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_EXCG)
629         std_fill            BYTE FILL;              /* unused
630     /*
631     /* Work areas for the individual command verbs (1 of these at top for alignment)
632     /*
633         dire_work           ADDRESS;                /* pointer to the work area for DIRECTORY
634         init_work           ADDRESS;                /* pointer to the work area for INIT
635         moun_work           ADDRESS;                /* pointer to the work area for MOUNT
636         rtnam_work          ADDRESS;                /* pointer to the work area for DELETE and RENAME
637     /*
638     /* Process identification items
639     /*
640         uic_member          WORD UNSIGNED;          /* binary member number for uic directories
641         uic_group           WORD UNSIGNED;          /* binary group number for uic directories
642         username            CHARACTER LENGTH 12;    /* process' username
643     /*
644     /* Exit handler for per-command activity, exh_routine is 0 if no handler is declared
```

```
645   /*
646       exit_block STRUCTURE;
647           exh_flink       ADDRESS;                /* system-maintained forward link
648           exh_routine     ADDRESS;                /* address of exit handling routine
649           exh_arg_count   LONGWORD UNSIGNED;      /* number of arguments for routine
650           exh_status      ADDRESS;                /* address to store status code, reason for exit
651           exh_volb        ADDRESS;                /* exit handler specific argument (volb address)
652           exh_condvalu    LONGWORD UNSIGNED;      /* put the status code here
653           END exit_block;
654   /*
655   /* Exit handler for global /CACHE actions
656   /*
657       cachexit_block STRUCTURE;
```



```
658           cachexh_flink   ADDRESS;                /* system-maintained forward link
659           cachexh_routine ADDRESS;                /* address of cachexit handling routine
660           cachexh_arg_count LONGWORD UNSIGNED;    /* number of arguments for routine
661           cachexh_status  ADDRESS;                /* address to store status code, reason for cachexit
662           cachexh_condvalu LONGWORD UNSIGNED;     /* put the status code here
663           END cachexit_block;
664   /*
665   /* Queue headers for managed resources
666   /*
667       dos11ctx_alloc                  ADDRESS;        /* Head of list of all allocated $dos11ctxs
668       dos11ctx_use STRUCTURE QUADWORD UNSIGNED;       /* Head of queue of $dos11ctxs in-use queue
669           dos11ctx_use_flink      ADDRESS;            /* Forward link
670           dos11ctx_use_blink      ADDRESS;            /* Backward link
671           END dos11ctx_use;
672       dos11ctx_avl STRUCTURE QUADWORD UNSIGNED;       /* Head of queue of all available $dos11ctxs
673           dos11ctx_avl_flink      ADDRESS;            /* Forward link
674           dos11ctx_avl_blink      ADDRESS;            /* Backward link
675           END dos11ctx_avl;
676   /*
677       filb_alloc              ADDRESS;                /* Head of list of all allocated $filBs
678       filb_use STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of file blocks in use queue
679           filb_use_flink  ADDRESS;                    /* Forward link
680           filb_use_blink  ADDRESS;                    /* Backward link
681           END filb_use;
682       filb_avl STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of all available $filBs
683           filb_avl_flink  ADDRESS;                    /* Forward link
684           filb_avl_blink  ADDRESS;                    /* Backward link
685           END filb_avl;
686   /*
687       namb_alloc              ADDRESS;                /* Head of list of all allocated $NAMBs
688       namb_use STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of name blocks in use queue
689           namb_use_flink  ADDRESS;                    /* Forward link
690           namb_use_blink  ADDRESS;                    /* Backward link
691           END namb_use;
692       namb_avl STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of all available $NAMBs
693           namb_avl_flink  ADDRESS;                    /* Forward link
694           namb_avl_blink  ADDRESS;                    /* Backward link
695           END namb_avl;
696   /*
697       rmsb_alloc              ADDRESS;                /* Head of list of all allocated $RMSBs
698       rmsb_use STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of $RMSBs in-use queue
699           rmsb_use_flink  ADDRESS;                    /* Forward link
700           rmsb_use_blink  ADDRESS;                    /* Backward link
701           END rmsb_use;
702       rmsb_avl STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of all available $RMSBs
703           rmsb_avl_flink  ADDRESS;                    /* Forward link
704           rmsb_avl_blink  ADDRESS;                    /* Backward link
```

```
705         END rmsb_avl;
706     /*
707         rt11ctx_alloc                   ADDRESS;        /* Head of list of all allocated $rt11ctxs
708         rt11ctx_use STRUCTURE QUADWORD UNSIGNED;        /* Head of queue of $rt11ctxs in-use queue
709             rt11ctx_use_flink           ADDRESS;        /* Forward link
710             rt11ctx_use_blink           ADDRESS;        /* Backward link
711             END rt11ctx_use;
712         rt11ctx_avl STRUCTURE QUADWORD UNSIGNED;        /* Head of queue of all available $rt11ctxs
713             rt11ctx_avl_flink           ADDRESS;        /* Forward link
714             rt11ctx_avl_blink           ADDRESS;        /* Backward link
715             END rt11ctx_avl;
```



```
716     /*
717         volb_alloc              ADDRESS;                /* Head of list of all allocated $VOLBs
718         volb_use STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of volume blocks in use queue
719             volb_use_flink      ADDRESS;                /* Forward link
720             volb_use_blink      ADDRESS;                /* Backward link
721             END volb_use;
722         volb_avl STRUCTURE QUADWORD UNSIGNED;           /* Head of queue of all available $VOLBs
723             volb_avl_flink      ADDRESS;                /* Forward link
724             volb_avl_blink      ADDRESS;                /* Backward link
725             END volb_avl;
726     /*
727     /* RMS data structures for command output, these point to memory at the end of the block
728     /*
729         sysout_fab              ADDRESS;                /* pointer to RMS File access block
730         sysout_rab              ADDRESS;                /* pointer to RMS Record access block
731         sysout_nam              ADDRESS;                /* pointer to RMS Name block
732         sysout_ebuf             ADDRESS;                /* pointer to RMS expanded name buffer
733         sysout_rbuf             ADDRESS;                /* pointer to RMS result name buffer
734     /*
735     /* Buffer for formatting
736     /*
737         fao_buffer              CHARACTER LENGTH 258;   /* used by util_fao_buffer
738     /*
739     /* Other items which would be in here if SDL supported external literals.  This space is allocated with the
740     /* rest of the block.  Certain pointers above point to these buffers.  This area is part of the block, it
741     /* is only due to restrictions in SDL that these are not actually defined here.
742     /*
743         /*  buf_sysout_fab  CHARACTER LENGTH #fab$k_bln;    /* RMS File access block
744         /*  buf_sysout_rab  CHARACTER LENGTH #rab$k_bln;    /* RMS Record access block
745         /*  buf_sysout_nam  CHARACTER LENGTH #nam$k_bln;    /* RMS Name block
746         /*  buf_sysout_ebuf CHARACTER LENGTH #nam$c_maxrss; /* RMS expanded name buffer
747         /*  buf_sysout_rbuf CHARACTER LENGTH #nam$c_maxrss; /* RMS result name buffer
748     /*
749         CONSTANT "length" EQUALS .;             /* length of data structure
750
751     END excg;
752
```



```
753     /**
754     /*
755     /* $FILB - File Block
756     /*
757     /* Contains data to describe a particular active file
758     /*
759     /*-
760
761     AGGREGATE filb STRUCTURE PREFIX filb$;
```

```
762
763     /*
764     /* Links to the queues of in-use and available  blocks, plus standard type and size fields
765     /*
766         header STRUCTURE QUADWORD UNSIGNED;          /* queue header
767             flink   ADDRESS;                        /* forward link
768             blink   ADDRESS;                        /* reverse link
769             END link;
770         size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_NAMB)
771         type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_NAMB)
772         std_fill            BYTE FILL;              /* unused
773     /*
774         alloc               ADDRESS;                /* link to list of allocated blocks
775     /*
776     /* The following name string has several uses depending on the routine.  For example, it is used to hold the
777     /*  result name from LIB$FIND_FILE routine for EXCH$FIL11_OPEN_FILE.
778     /*
779         name_string         QUADWORD UNSIGNED;
780     /*
781     /* The rest of the block will be reset to nulls each time the block is allocated
782     /*
783         CONSTANT start_zero EQUALS .;
784     /*
785     /*
786         assoc_namb          ADDRESS;                /* namb for this file
787         assoc_volb          ADDRESS;                /* address of mounted volb for this name, 0 for Files-11
788     /*
789         context             ADDRESS;                /* pointer to volume-specific structure defining file context
790                                                     /*  DOS11CTX structure for DOS-11 tape files
791                                                     /*  RT11CTX structure for RT-11 disk files
792                                                     /*  RMSB for Files-11 files
793         fil11_wcc           ADDRESS;                /* wildcard context for LIB$FIND_FILE routine for Files-11 files
794     /*
795     /* The record format for this file (the volume format byte is in the assoc_namb)
796     /*
797         rec_format          BYTE UNSIGNED;          /* record format code, a constant filb$k_rfmt_xxx
798             CONSTANT (
799                 rfmt_invalid,                       /* value 0, format not set to known value
800                 rfmt_binary,                        /* formatted binary records
801                 rfmt_fixed,                         /* fixed length records
802                 rfmt_stream                         /* stream format records
803                  ) EQUALS 0 INCREMENT 1;
804     /*
805         CONSTANT rfmt_lobound EQUALS 0;                     /* low bound for case statement
806         CONSTANT rfmt_hibound EQUALS filb$k_rfmt_stream;     /* high bound
807     /*
808         transfer_mode       BYTE UNSIGNED;          /* transfer mode code, a constant filb$k_xfrm_xxx
809             CONSTANT (
```



```
810                 xfrm_automatic,                     /* decide block or record automatically, the default, value is zero
811                 xfrm_block,                         /* transfer block by block
812                 xfrm_record                         /* transfer record by record
813                  ) EQUALS 0 INCREMENT 1;
814     /*
815     /* The carriage control for this file
816     /*
817         car_control         BYTE UNSIGNED;          /* carriage control code, a constant filb$k_cctl_xxx
818             CONSTANT (
819                 cctl_cr,                            /* carriage return, the default, value is zero
820                 cctl_fortran,                       /* FORTRAN files
821                 cctl_none                           /* no carriage control
```

```
822             ) EQUALS 0 INCREMENT 1;
823     /*
824         CONSTANT cctl_lobound EQUALS 0;                     /* low bound for case statement
825         CONSTANT cctl_hibound EQUALS filb$k_cctl_none;      /* high bound
826     /*
827     /* Record attribute and other misc flags
828     /*
829         flags STRUCTURE WORD UNSIGNED;                      /* the whole flags word
830             rfmt_explicit           BITFIELD MASK;          /* record format specified
831             cctl_explicit           BITFIELD MASK;          /* carriage control specified
832             file_erased             BITFIELD MASK;          /* file has been erased (e.g. due to write errors)
833             files_found             BITFIELD MASK;          /* one or more files have been found using this filb
834             files_created           BITFIELD MASK;          /* one or more files have been created using this filb
835             explicit_version        BITFIELD MASK;          /* file name concained an explicit version
836             delete_previous         BITFIELD MASK;          /* delete previous version of RT11 file during close
837             name_change             BITFIELD MASK;          /* the output name was changed from the input
838             q_best_try_contiguous   BITFIELD MASK;          /* there is a /BEST on the input file
839             q_contiguous            BITFIELD MASK;          /* there is a /CONT on the input file
840             q_truncate              BITFIELD MASK;          /* there is a /TRUN on the input file
841             END flags;
842     /*
843         q_allocation        LONGWORD UNSIGNED;              /* a /ALLOC was on the input file
844         q_extension         LONGWORD UNSIGNED;              /* a /EXTEN was on the input file
845         fixed_len           LONGWORD UNSIGNED;              /* length of FIXED records
846         pad_char            BYTE UNSIGNED;                  /* pad character, default 0
847     /*
848         result_name_len     LONGWORD UNSIGNED;              /* length of the result name
849     /*
850         block_count         LONGWORD UNSIGNED;              /* size of an existing file
851     /*
852     /* Length of record, record address
853     /*
854         record_len          LONGWORD UNSIGNED;              /* length of current record
855         record              ADDRESS;                        /* address of current record
856     /*
857     /* Addresses of action routines for this file
858     /*
859         close_routine       ADDRESS;                        /* address of routine to close the file
860         delete_routine      ADDRESS;                        /* address of routine to delete the file
861         get_routine         ADDRESS;                        /* routine to get records, 0 for output-only files
862         put_routine         ADDRESS;                        /* routine to put records, 0 for input-only files
863     /*
864         CONSTANT end_zero EQUALS .;
865     /*
866     /* No need to zero all the buffers
867     /*
```



```
868         result_name         CHARACTER LENGTH 256;   /* buffer for the result name
869     /*
870     /* and actual record for when we need move mode
871     /*
872         record_buffer       CHARACTER LENGTH #rec_buf_size;
873         guard_byte          BYTE FILL;              /* eob logic much easier if we can write 1 byte past end of buffer
874
875         CONSTANT "length" EQUALS .;                 /* length of data structure
876
877     END filb;
878
```

```
879     /**
880     /*
881     /* $INIT - Init block
882     /*
883     /* Contains the work area for the INIT verb
884     /*
885     /*-
886
887     AGGREGATE init STRUCTURE PREFIX init$;
888     /*
889         namb                ADDRESS;                /* pointer to our name block
890         volb                ADDRESS;                /* pointer to our volume block
891     /*
892         size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_INIT)
893         type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_INIT)
894         std_fill            BYTE FILL;              /* unused
895     /*
896     /* Character string descriptors for name components
897     /*
898         device              QUADWORD UNSIGNED;      /* desc for device name, or filename for INIT /CREATE
899         volumeid            QUADWORD UNSIGNED;      /* desc for volume id (for RT-11)
900     /*
901     /* Command qualifier variables and flags
902     /*
903         q_allocation        LONGWORD UNSIGNED;      /* size of virtual device to create /ALLOC=n
904         q_extra_words       LONGWORD UNSIGNED;      /* number of extra words for directory entries  /EXTRA=n
905         q_segments          LONGWORD UNSIGNED;      /* number of directory segments /SEGMENTS=n
906         flags STRUCTURE LONGWORD UNSIGNED;          /* the whole flags longword
907             q_create                BITFIELD MASK;  /* /CREATE
908             q_message               BITFIELD MASK;  /* /MESSAGE
909     {\      q_badblocks             BITFIELD MASK;  /* /BADBLOCKS
910     {\      q_badblocks_retain      BITFIELD MASK;  /* /BADBLOCKS=RETAIN
911     {\      q_replace               BITFIELD MASK;  /* /REPLACE
912     {\      q_replace_retain        BITFIELD MASK;  /* /REPLACE=RETAIN
913             END flags;
914     /*
915         CONSTANT "length" EQUALS .;                 /* length of data structure
916
917     END init;
918
                                                    15-SEP-1984 23:01:31.30     SDL V2.0                            Page   23
                                                    15-SEP-1984 22:44:23        _$255$DUA28:[EXCHNG.SRC]EXCDEFS.SDL;1

919     /**
920     /*
921     /* $MOUN - Mount block
922     /*
923     /* Contains the work area for the MOUNT verb
924     /*
925     /*-
926
927     AGGREGATE moun STRUCTURE PREFIX moun$;
928     /*
929         namb                ADDRESS;                /* pointer to our name block
930         volb                ADDRESS;                /* pointer to our volume block
931     /*
932         size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_MOUN)
933         type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_MOUN)
934         std_fill            BYTE FILL;              /* unused
935     /*
936     /* Character string descriptors for name components
937     /*
938         device              QUADWORD UNSIGNED;      /* desc for device name
```

```
939         filename            QUADWORD UNSIGNED;      /* desc for virtual device input file name
940     /*
941     /* Command qualifier flags
942     /*
943         q_write             LONGWORD UNSIGNED;      /* write protection on the device (cli$_present tested)
944         flags STRUCTURE LONGWORD UNSIGNED;  /* the whole flags longword
945             q_read_check    BITFIELD MASK;          /* /DATA_CHECK=READ
946             q_write_check   BITFIELD MASK;          /* /DATA_CHECK=WRITE
947             q_foreign       BITFIELD MASK;          /* /FOREIGN device mount
948             q_virtual       BITFIELD MASK;          /* /VIRTUAL virtual device mount
949             q_message       BITFIELD MASK;          /* /MESSAGE was given
950             END flags;
951     /*
952         CONSTANT "length" EQUALS .;                 /* length of data structure
953
954     END moun;
955
```



```
956     /*+
957     /*
958     /* $NAMB - File Name Block
959     /*
960     /* Contains a parsed file specification
961     /*
962     /*-
963
964     AGGREGATE namb STRUCTURE PREFIX namb$;
965
966     /*
967     /* Links to the queues of in-use and available  blocks, plus standard type and size fields
968     /*
969         header STRUCTURE QUADWORD UNSIGNED;          /* queue header
970             flink   ADDRESS;                        /* forward link
971             blink   ADDRESS;                        /* reverse link
972             END link;
973         size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_NAMB)
974         type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_NAMB)
975         std_fill            BYTE FILL;              /* unused
976     /*
977         alloc               ADDRESS;                /* link to list of allocated blocks
978     /*
979     /* Dynamic character string descriptors for name strings
980     /*
981         input       QUADWORD UNSIGNED;              /* desc for input name
982         fullname    QUADWORD UNSIGNED;              /* desc for most complete (exp or res)
983         expanded    QUADWORD UNSIGNED;              /* desc for expanded name
984         result      QUADWORD UNSIGNED;              /* desc for result name
985         device_dvi  QUADWORD UNSIGNED;              /* desc for canonical device
986     /*
987     /* Descriptors of individual components of name.  These might be fixed descriptors pointing into one of the
988     /* above strings (usual case), or they might be dynamic copies of the components (the namb was cloned).
989     /*
990         node        QUADWORD UNSIGNED;              /* desc for node name
991         device      QUADWORD UNSIGNED;              /* desc for device name
992         directory   QUADWORD UNSIGNED;              /* desc for directory name
993         name        QUADWORD UNSIGNED;              /* desc for name name
994         type        QUADWORD UNSIGNED;              /* desc for type name
995         version     QUADWORD UNSIGNED;              /* desc for version name
996
997     /*
998     /* The rest of the block will be reset to nulls each time the block is allocated.  The routine EXCH$CMD_NAMB_CLONE
```

```
 999    /* copies the rest of the block to a new namb when it clones the namb.
1000    /*
1001        CONSTANT start_zero EQUALS .;
1002    /*
1003    /* Carry some info from the RMS structures
1004    /*
1005        fabdev              LONGWORD UNSIGNED;      /* device characteristics from FAB$L_DEV
1006        nameflags STRUCTURE LONGWORD UNSIGNED;      /* the whole flags longword
1007            wildcard                BITFIELD MASK;  /* some field contains a wildcard rms's [nam$v_wildcard]
1008            wild_name               BITFIELD MASK;  /* file name contains a wildcard  rms's [nam$v_wild_name]
1009            wild_type               BITFIELD MASK;  /* file type contains a wildcard  rms's [nam$v_wild_type]
1010            wild_version            BITFIELD MASK;  /* file version has a wildcard    rms's [nam$v_wild_ver]
1011            wild_group              BITFIELD MASK;  /* group number is a wildcard     rms's [nam$v_wild_grp]
1012            wild_member             BITFIELD MASK;  /* member number is a wildcard    rms's [nam$v_wild_mbr]

                                                    15-SEP-1984 23:01:31.50     SDL V2.0                        Page   25
                                                    15-SEP-1984 22:44:23        _$255$DUA28:[EXCHNG.SRC]EXCDEFS.SDL;1
1013    /*
1014            explicit_node           BITFIELD MASK;  /* device name is explicit        rms's [nam$v_node]
1015            explicit_device         BITFIELD MASK;  /* device name is explicit        rms's [nam$v_exp_dev]
1016            explicit_directory      BITFIELD MASK;  /* directory name is explicit     rms's [nam$v_exp_dir]
1017            explicit_name           BITFIELD MASK;  /* file name is explicit          rms's [nam$v_exp_name]
1018            explicit_type           BITFIELD MASK;  /* file type is explicit          rms's [nam$v_exp_type]
1019            explicit_version        BITFIELD MASK;  /* an explicit version number     rms's [nam$v_exp_ver]
1020    /*
1021            concealed_device        BITFIELD MASK;  /* device name is a concealed dev rms's [nam$v_cncl_dev]
1022            rooted_directory        BITFIELD MASK;  /* device name has a root dir     rms's [nam$v_root_dir]
1023            uic_directory           BITFIELD MASK;  /* directory is [GROUP,MEMBER]    rms's [nam$v_grp_mbr]
1024    /*
1025            more_files              BITFIELD MASK;  /* CLI$GET_VALUE gave us the CLI$_COMMA status
1026            bad_pdp_char            BITFIELD MASK;  /* a char in the name or type field isn't valid for RT11 or DOS11
1027            dos_truncate            BITFIELD MASK;  /* name or type too long for DOS11
1028            rt_truncate             BITFIELD MASK;  /* name or type too long for RT11
1029            END nameflags;
1030    /*
1031        next                ADDRESS;                /* Some commands link multiple nambs off this pointer
1032        assoc_volb          ADDRESS;                /* address of mounted volb referencing this name
1033    /*
1034    /* Keep a copy of the volume and record formats.  Since both /VOLUME and /RECORD are local qualifiers, it is only
1035    /* when we fetch the parameters that we can see any explicit qualifiers.  Implied volume formats can be done when
1036    /* we see the parameter.  Implied record formats can't be done until we have the fully expanded filename
1037    /*
1038        devclass            BYTE UNSIGNED;          /* device class (a DC$_xxx symbol)
1039        devtype             BYTE UNSIGNED;          /* device type (a DT$_xxx symbol)
1040        vol_format          BYTE UNSIGNED;          /* volume format code, a constant volb$k_vfmt_xxx
1041        rec_format          BYTE UNSIGNED;          /* record format code, a constant filb$k_rfmt_xxx
1042        transfer_mode       BYTE UNSIGNED;          /* transfer mode code, a constant filb$k_xfrm_xxx
1043        car_control         BYTE UNSIGNED;          /* carriage control code, a constant filb$k_cctl_xxx
1044        fixed_len           LONGWORD UNSIGNED;      /* length of FIXED records
1045        pad_char            BYTE UNSIGNED;          /* pad character, default 0
1046        uic_member          BYTE UNSIGNED;          /* binary member number for uic directories
1047        uic_group           BYTE UNSIGNED;          /* binary group number for uic directories
1048        flags STRUCTURE BYTE UNSIGNED;              /* the whole flags byte
1049            rfmt_explicit           BITFIELD MASK;  /* record format specified
1050            cctl_explicit           BITFIELD MASK;  /* carriage control specified
1051            vfmt_explicit           BITFIELD MASK;  /* volume format specified
1052            END flags;
1053    /*
1054    /* Unique volume identifier
1055    /*
1056        vol_ident_len       LONGWORD UNSIGNED;      /* Length of the next field.  Used for text display only,
1057                                                    /* comparisions look at the entire vol_ident field
1058        vol_ident           CHARACTER LENGTH 128;   /* needs to be as long as a logical name
```

```
1059    /*
1060        CONSTANT "length" EQUALS .;                 /* length of data structure
1061
1062    END namb;
1063
```



```
1064    /**
1065    /*
1066    /* $RMSB - RMS structures for a file
1067    /*
1068    /* Contains all the RMS control structures for a file.
1069    /*
1070    /*-
1071
1072    AGGREGATE rmsb STRUCTURE PREFIX rmsb$;
1073
1074    /*
1075    /* Links to the queues of in-use and available blocks, plus standard type and size fields
1076    /*
1077        header STRUCTURE QUADWORD UNSIGNED;         /* queue header
1078            flink   ADDRESS;                        /* forward link
1079            blink   ADDRESS;                        /* reverse link
1080            END link;
1081        size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_rmsb)
1082        type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_rmsb)
1083        std_fill            BYTE FILL;              /* unused
1084    /*
1085        alloc               ADDRESS;                /* link to list of allocated blocks
1086    /*
1087    /* Storage for the following control blocks is allocated at the end of the block
1088    /*
1089        fab         ADDRESS;                        /* pointer to RMS File Access Block
1090        rab         ADDRESS;                        /* pointer to RMS Record Access Block
1091        nam         ADDRESS;                        /* pointer to RMS Name Block
1092        esbuf       ADDRESS;                        /* pointer to RMS Expanded String Buffer
1093        rsbuf       ADDRESS;                        /* pointer to RMS Result String Buffer
1094
1095    /*
1096    /* The rest of the block will be reset to nulls each time the block is allocated
1097    /*
1098        CONSTANT start_zero EQUALS .;
1099    /*
1100    /* Other items which would be in here if SDL supported external literals.  This space is allocated with the
1101    /* rest of the block.  Certain pointers above point to these buffers.  This area is part of the block, it
1102    /* is only due to restrictions in SDL that these are not actually defined here.
1103    /*
1104        /*  buf_fab         CHARACTER LENGTH #fab$k_bln;    /* RMS File access block
1105        /*  buf_rab         CHARACTER LENGTH #rab$k_bln;    /* RMS Record access block
1106        /*  buf_nam         CHARACTER LENGTH #nam$k_bln;    /* RMS Name block
1107        /*  buf_esbuf       CHARACTER LENGTH #nam$c_maxrss; /* RMS expanded name buffer
1108        /*  buf_rsbuf       CHARACTER LENGTH #nam$c_maxrss; /* RMS result name buffer
1109    /*
1110        CONSTANT "length" EQUALS .;                 /* length of data structure
1111
1112    END rmsb;
1113
```



```
1114    /**
1115    /*
```

```
1116    /* $RT11 - volume specific structure for an RT-11 volume
1117    /*
1118    /* Contains all the RT-11 specific control structures for a volume.  This structure would
1119    /* normally hang from VOLB$A_VFMT_SPECIFIC.
1120    /*
1121    /*-
1122
1123    #rt_dirseg = 1024;                          /* Size of an RT11 directory segment
1124    #rt_root   = 6;                             /* Directory starts on block 6 (pbn)
1125
1126    AGGREGATE rt11 STRUCTURE PREFIX rt11$;
1127
1128        dummy               QUADWORD FILL;      /* spacer to put size and type in correct place
1129    /*
1130        size                WORD UNSIGNED;      /* standard size of the block   (symbol EXCHBLK$S_rmsb)
1131        type                BYTE UNSIGNED;      /* type code field              (symbol EXCHBLK$K_rmsb)
1132        std_fill            BYTE FILL;          /* unused
1133    /*
1134        CONSTANT start_zero EQUALS .;
1135    /*
1136    /* Variables to describe RT-11 go here
1137    /*
1138        status STRUCTURE WORD UNSIGNED;         /* complete status word
1139            dir_present     BITFIELD MASK;      /* directory is resident (up to highest block in use)
1140            END status;
1141    /*
1142        CONSTANT end_zero EQUALS .;
1143    /*
1144    /* ************************************************************
1145    /*
1146    /*  From here to the end of the structure are read in the
1147    /*  first few blocks of the disk, containing the home block
1148    /*  and the entire directory
1149    /*
1150    /* ************************************************************
1151    /*
1152    /* Physical block 0 contains the boot block info
1153    /*
1154         block_0            CHARACTER LENGTH 512;
1155    /*
1156    /* Physical block 1 contains the volume home block
1157    /*
1158        block_1_overlay UNION;
1159            block_1         CHARACTER LENGTH 512;
1160            home_block STRUCTURE;
1161                replace_tbl CHARACTER LENGTH 130;
1162                fill_00     WORD FILL;
1163                init_data   CHARACTER LENGTH 38;
1164                bup_data    CHARACTER LENGTH 18;
1165                fill_01     CHARACTER LENGTH 260 FILL;
1166                rte_name    WORD UNSIGNED;
1167                rte_block   WORD UNSIGNED;
1168                fill_02     CHARACTER LENGTH 14 FILL;
1169                cluster     WORD UNSIGNED;
1170                first_seg   WORD UNSIGNED;
```



```
1171                system_vers WORD UNSIGNED;
1172                volume_id   CHARACTER LENGTH 12;
1173                owner_name  CHARACTER LENGTH 12;
1174                system_id   CHARACTER LENGTH 12;
1175                fill_03     WORD FILL;
```

```
1176                checksum    WORD UNSIGNED;
1177                END home_block;
1178            END block_1_overlay;
1179    /*
1180    /* Physical blocks 2-5 are reserved.
1181    /*
1182        block_2             CHARACTER LENGTH 512;
1183        block_3             CHARACTER LENGTH 512;
1184        block_4             CHARACTER LENGTH 512;
1185        block_5             CHARACTER LENGTH 512;
1186    /*
1187    /* Directory segments start here
1188    /*
1189        dire_segments       CHARACTER LENGTH #rt_dirseg*31;
1190    /*
1191        CONSTANT "length" EQUALS .;                 /* length of data structure
1192        CONSTANT dirseglen EQUALS #rt_dirseg;
1193        CONSTANT root_block EQUALS #rt_root;
1194
1195    END rt11;
1196
```



```
1197    /**
1198    /*
1199    /* $RT11HOM - home block structure for an RT-11 volume
1200    /*
1201    /*      ;*  (From RT-11 V5 source file DUPROT.MAC)
1202    /*      ;
1203    /*      ;                           RT-11 V05 Home Block Format
1204    /*      ;
1205    /*      ;      0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
1206    /*      ;      0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
1207    /*      ;000  |a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
1208    /*      ;040  |a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
1209    /*      ;100  |a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
1210    /*      ;140  |a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
1211    /*      ;200  |a|a|   |b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|
1212    /*      ;240  |b|b|b|b|b|b|b|b|b|b|l|l|l|l|l|l|l|l|l|l|l|l|l|l|l|l|l|l|   |   |
1213    /*      ;300  |                                                               |
1214    /*      ;340  |                                                               |
1215    /*      ;400  |                                                               |
1216    /*      ;440  |                                                               |
1217    /*      ;500  |                                                               |
1218    /*      ;540  |                                                               |
1219    /*      ;600  |                                                               |
1220    /*      ;640  |                                                               |
1221    /*      ;700  |c|c|d|d|                       |e|e|f|f|g|g|h|h|h|h|h|h|h|h|
1222    /*      ;740  |h|h|h|h|i|i|i|i|i|i|i|i|i|i|i|i|i|j|j|j|j|j|j|j|j|j|j|j|   |   |k|k|
1223    /*      ;
1224    /*      ;Field  Location (8)    Contents                                    Default
1225    /*      ;
1226    /*      ;  a    000 - 201       Bad block replacement table
1227    /*      ;  b    204 - 251       INITIALIZE/RESTORE data area
1228    /*      ;  l    252 - 273       BUP information area                        (000000)
1229    /*      ;  c    700 - 701       RAD50 RTE (RTEM)                            (000000)
1230    /*      ;  d    702 - 703       Block number of first user file (RTEM)      (000000)
1231    /*      ;  e    722 - 723       Pack cluster size                           (000001)
1232    /*      ;  f    724 - 725       Block number of first directory segment     (000006)
1233    /*      ;  g    726 - 727       System version                              (RAD50 V3A)
1234    /*      ;  h    730 - 743       Volume ID                                   (RT11Abbbbbbb)
1235    /*      ;  i    744 - 757       Owner name                                  (bbbbbbbbbbbb)
```

```
1236    /*      ;  j    760 - 773       System ID                                   (DECRT11Abbbb)
1237    /*      ;  k    776 - 777       Checksum
1238    /*      ;
1239    /*      ;All other areas in the home block are reserved for future system use.
1240    /*      ;The contents of all other areas are undefined.
1241    /*      ;
1242    /*      ; (End of DUPROT.MAC inclusion)
1243    /*      ;-
1244    /*-
1245
1246    AGGREGATE rt11hom STRUCTURE PREFIX rt11hom$;
1247
1248        replace_tbl CHARACTER LENGTH 130;           /* bad block replacement table
1249        fill_00     WORD FILL;
1250        init_data   CHARACTER LENGTH 38;            /* INITIALIZE /RESTORE data area
1251        bup_data    CHARACTER LENGTH 18;            /* BUP data area
1252        fill_01     CHARACTER LENGTH 260 FILL;
1253        rte_name    WORD UNSIGNED;                  /* Name of RTE to mark RTEM volume

                                                        15-SEP-1984 23:01:31.30     SDL V2.0                        Page   30
                                                        15-SEP-1984 22:44:23        _$255$DUA28:[EXCHNG.SRC]EXCDEFS.SDL;1
1254        rte_block   WORD UNSIGNED;                  /* RTEM block number for first user file
1255        fill_02     CHARACTER LENGTH 14 FILL;
1256        cluster     WORD UNSIGNED;                  /* Pack cluster size
1257        first_seg   WORD UNSIGNED;                  /* PBN of first directory segment
1258        system_vers WORD UNSIGNED;                  /* System version number
1259        volume_id   CHARACTER LENGTH 12;            /* Volume id
1260        owner_name  CHARACTER LENGTH 12;            /* Owner name
1261        system_id   CHARACTER LENGTH 12;            /* System ID
1262        fill_03     WORD FILL;
1263        checksum    WORD UNSIGNED;                  /* Checksum (not really used)
1264    /*
1265        CONSTANT "length" EQUALS .;                 /* length of data structure
1266
1267    END rt11hom;
1268

                                                        15-SEP-1984 23:01:31.30     SDL V2.0                        Page   31
                                                        15-SEP-1984 22:44:23        _$255$DUA28:[EXCHNG.SRC]EXCDEFS.SDL;1

1269    /*
1270    /* Define various RT11 directory structures.  These will not be independent
1271    /* structures, but will be mapped over the directory segments as needed.
1272    /*
1273
1274    /* RT-11 directory segment header
1275
1276    AGGREGATE rt11hdr STRUCTURE PREFIX rt11hdr$;
1277    /*
1278        num_segs            WORD UNSIGNED;          /* Total number of segments in this directory
1279        next_seg            WORD UNSIGNED;          /* Segment number of next logical segment
1280        high_seg            WORD UNSIGNED;          /* Highest segment in use - valid for segment 1 only
1281        extra_bytes         WORD UNSIGNED;          /* Extra bytes per directory entry
1282        start_block         WORD UNSIGNED;          /* Pbn where stored data begins
1283    /*
1284        CONSTANT "length" EQUALS .;                 /* length of data structure
1285
1286    END rt11hdr;
1287
1288    /*
1289    /* RT-11 directory entry
1290    /*
1291
1292    AGGREGATE rt11ent STRUCTURE PREFIX rt11ent$;
```

```
1293    /*
1294        fill_00             BYTE FILL;
1295        type_union UNION;
1296            type_byte       BYTE UNSIGNED;          /* Type of directory entry
1297            type            BITFIELD LENGTH 4;      /* Entry is integer in first part
1298            type_bits STRUCTURE;
1299                typ_tentative   BITFIELD MASK;      /* Tentative file
1300                typ_empty       BITFIELD MASK;      /* Empty entry
1301                typ_permanent   BITFIELD MASK;      /* Permanent file
1302                typ_end_segment BITFIELD MASK;      /* Last (possibly incomplete) entry in segment
1303                typ_fill_00     BITFIELD LENGTH 3 FILL;
1304                typ_protected   BITFIELD MASK;      /* File is protected
1305                END type_bits;
1306            END type_union;
1307        filename            LONGWORD UNSIGNED;      /* 6 Radix-50 characters
1308        filetype            WORD UNSIGNED;          /* 3 Radix-50 characters
1309        blocks              WORD UNSIGNED;          /* Total length of file
1310        channel             BYTE UNSIGNED;          /* <not meaningful for EXCHANGE>
1311        job                 BYTE UNSIGNED;          /* <not meaningful for EXCHANGE>
1312        date STRUCTURE WORD UNSIGNED;               /* Creation date field
1313            year            BITFIELD LENGTH 5;      /* Years since 1972
1314            day             BITFIELD LENGTH 5;      /* In decimal, 1-31
1315            month           BITFIELD LENGTH 5;      /* In decimal, 1-12
1316            END date;
1317    /*
1318        CONSTANT ''length'' EQUALS .;               /* length of data structure
1319
1320    END rt11ent;
1321
```



```
1322    /*
1323    /* RT-11 file context block  This is a combination of the FAB, RAB and NAM block
1324    /* for an RT11 random access file
1325    /*
1326
1327    AGGREGATE rt11ctx STRUCTURE PREFIX rt11ctx$;
1328    /*
1329    /* Links to the queues of in-use and available  blocks, plus standard type and size fields
1330    /*
1331        header STRUCTURE QUADWORD UNSIGNED;         /* queue header
1332            flink   ADDRESS;                        /* forward link
1333            blink   ADDRESS;                        /* reverse link
1334            END link;
1335    /*
1336        size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_rt11ctx)
1337        type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_rt11ctx)
1338        std_fill            BYTE FILL;              /* unused
1339    /*
1340        alloc               ADDRESS;                /* link to list of allocated blocks
1341    /*
1342    /* Addresses of associated control structures
1343    /*
1344        assoc_filb          ADDRESS;                /* address of the filb
1345        assoc_volb          ADDRESS;                /* address of the volb
1346    /*
1347    /* I/O buffer for this record stream
1348    /*
1349        buffer              ADDRESS;                /* Pointer to the buffer
1350    /*
1351        CONSTANT start_zero EQUALS .;               /* here to end is cleared for each copy command
1352    /*
```

```
1353    /* Data to describe a record stream in the file.
1354    /*
1355        cur_block           LONGWORD UNSIGNED;      /* pbn of current block
1356        eof_block           LONGWORD UNSIGNED;      /* pbn of last block in file
1357        cur_byte            LONGWORD UNSIGNED;      /* offset into the block
1358    /*
1359    /* Flags for an open file.  This longword will be zero when file is not open
1360    /*
1361        flags STRUCTURE LONGWORD UNSIGNED;          /* the whole flags longword
1362            stream_active           BITFIELD MASK;  /* block currently describes an open file
1363            output_file             BITFIELD MASK;  /* block currently describes an output file
1364            flush                   BITFIELD MASK;  /* tell advance routine to flush last block
1365            pointer         BITFIELD MASK;          /* tape position is at this file
1366            END flags;
1367    /*
1368    /* Pointers to the I/O buffer for this record stream
1369    /*
1370        buf_base_block      LONGWORD UNSIGNED;      /* Pbn of first block in buffer
1371        buf_high_block      LONGWORD UNSIGNED;      /* Pbn of last block in buffer
1372        high_block_written  LONGWORD UNSIGNED;      /* Pbn of highest block to the file
1373    /*
1374    /* The next part is a duplicate of the directory entry
1375    /*
1376        entry_union UNION;
1377            "entry"         CHARACTER LENGTH 14;        /* The overall entry
1378            entry_fields STRUCTURE;
```



```
1379                fill_00             BYTE FILL;
1380                type_union UNION;
1381                    type_byte       BYTE UNSIGNED;      /* Type of directory entry
1382                    type            BITFIELD LENGTH 4;  /* Entry is integer in first part
1383                    type_bits STRUCTURE;
1384                        typ_tentative   BITFIELD MASK;  /* Tentative file
1385                        typ_empty       BITFIELD MASK;  /* Empty entry
1386                        typ_permanent   BITFIELD MASK;  /* Permanent file
1387                        typ_end_segment BITFIELD MASK;  /* Last (possibly incomplete) entry in segment
1388                        typ_fill_00     BITFIELD LENGTH 3 FILL;
1389                        typ_protected   BITFIELD MASK;  /* File is protected
1390                        END type_bits;
1391                    END type_union;
1392                filename            LONGWORD UNSIGNED;  /* 6 Radix-50 characters
1393                filetype            WORD UNSIGNED;      /* 3 Radix-50 characters
1394                blocks              WORD UNSIGNED;      /* Total length of file
1395                channel             BYTE UNSIGNED;      /* <not meaningful for EXCHANGE>
1396                job                 BYTE UNSIGNED;      /* <not meaningful for EXCHANGE>
1397                date STRUCTURE WORD UNSIGNED;           /* Creation date field
1398                    year            BITFIELD LENGTH 5;  /* Years since 1972
1399                    day             BITFIELD LENGTH 5;  /* In decimal, 1-31
1400                    month           BITFIELD LENGTH 5;  /* In decimal, 1-12
1401                    END date;
1402                END entry_fields;
1403            END entry_union;
1404    /*
1405    /* Expanded name fields
1406    /*
1407        exp_fullname_len    LONGWORD UNSIGNED;      /* Length of the concatenated name
1408        exp_name_len        LONGWORD UNSIGNED;      /* Length of the name field
1409        exp_type_len        LONGWORD UNSIGNED;      /* Length of the file extension (type)
1410        exp_protected       CHARACTER LENGTH 2;     /*  "  " for normal files, "P " for protected
1411        exp_fullname        CHARACTER LENGTH 10;    /* Name in normal form, no embedded blanks
1412        exp_name            CHARACTER LENGTH 6;     /* Blank padded name
```

```
1413        exp_type            CHARACTER LENGTH 3;     /* Blank padded extension
1414        exp_date            CHARACTER LENGTH 11;    /* VMS format ascii date
1415    /*
1416    /* The following items define the location of the above entry.  They can be used to restore to a position in the
1417    /* RT11 directory, such as to continue wildcard processing of the directory.  Note, however, that the RT11
1418    /* directory might be reorganized for many different reasons, such as the creation of a file.  The routine
1419    /* EXCH$RT11_CHECK_POSITION will check to make sure that these values still describe the location of the entry,
1420    /* adjusting them to the new location if the entry is not in the same position.
1421    /*
1422        start_block         LONGWORD UNSIGNED;      /* starting pbn of file
1423        seg_number          LONGWORD UNSIGNED;      /* directory segment containing this entry
1424        seg_address         ADDRESS;                /* address of start of current segment
1425        ent_address         ADDRESS;                /* address of next directory entry
1426    /*
1427        CONSTANT end_zero EQUALS .;
1428        CONSTANT "length" EQUALS .;                 /* length of data structure
1429
1430    END rt11ctx;
1431
```



```
1432    /**
1433    /*
1434    /* $RTNAM
1435    /*
1436    /* Contains the work area for the DELETE and RENAME verbs
1437    /*
1438    /*-
1439
1440    AGGREGATE rtnam STRUCTURE PREFIX rtnam$;
1441    /*
1442    /* Dynamic character string descriptors for name components.  These strings are reclaimed at the start of each command
1443    /*
1444        input_filename      QUADWORD UNSIGNED;      /* desc for current item from input parameter list
1445    /*
1446        size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_rtnam)
1447        type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_rtnam)
1448        std_fill            BYTE FILL;              /* unused
1449    /*
1450        output_filename     QUADWORD UNSIGNED;      /* desc for output parameter
1451    /*
1452        input_sticky_name   QUADWORD UNSIGNED;      /* name to make input files sticky
1453    /*
1454        CONSTANT start_zero EQUALS .;               /* here to end is cleared for each command
1455    /*
1456    /*
1457    /* Flags for boolean DCL command qualifiers
1458    /*
1459        qual_flags STRUCTURE LONGWORD UNSIGNED;     /* the whole flags longword
1460    {\      q_confirm       BITFIELD MASK;
1461            q_log           BITFIELD MASK;
1462            q_protect       BITFIELD MASK;          /* value for protect bit, not useful unless explicit also set
1463            q_protect_explicit BITFIELD MASK;       /* an explicit /PROTECT or /NOPROTECT was seen
1464            q_system        BITFIELD MASK;
1465            END qual_flags;
1466    /*
1467    /* Local command flags
1468    /*
1469        rtnam_flags STRUCTURE LONGWORD UNSIGNED;    /* the whole flags longword
1470            multiple_files  BITFIELD MASK;          /* the input potentially describes multiple files
1471            delete_command  BITFIELD MASK;          /* the command is DELETE
1472            rename_command  BITFIELD MASK;          /* the command is RENAME
```

```
1473            out_fetched     BITFIELD MASK;          /* output filename for rename has been fetched
1474            END rtnam_flags;
1475    /*
1476    /* Other misc data items
1477    /*
1478    /* Work area for the current input file
1479    /*
1480        inp_filb            ADDRESS;                /* pointer to a filb for the input file
1481        inp_namb            ADDRESS;                /* pointer to a namb describing the input file
1482    /*
1483    /* Work area for the current output file
1484    /*
1485        out_filb            ADDRESS;                /* pointer to a filb for the output file
1486        out_namb            ADDRESS;                /* pointer to a namb describing the output file
1487    /*
1488        CONSTANT end_zero EQUALS .;                 /* no need to clear the buffers (or create pages not to be used)
```



```
1489        CONSTANT "length" EQUALS .;                 /* length of data structure
1490
1491    END rtnam;
1492
```



```
1493    /*+
1494    /*
1495    /* $VOLB - Volume Control Block
1496    /*
1497    /* One entry per active foreign volume.  A partially filled VOLB is used to describe RMS files.
1498    /*
1499    /*-
1500
1501    AGGREGATE volb STRUCTURE PREFIX volb$;
1502
1503    /*
1504    /* Links to the queues of in-use and available  blocks, plus standard type and size fields
1505    /*
1506        header STRUCTURE QUADWORD UNSIGNED;          /* queue header
1507            flink   ADDRESS;                        /* forward link
1508            blink   ADDRESS;                        /* reverse link
1509            END link;
1510        size                WORD UNSIGNED;          /* standard size of the block   (symbol EXCHBLK$S_VOLB)
1511        type                BYTE UNSIGNED;          /* type code field              (symbol EXCHBLK$K_VOLB)
1512        std_fill            BYTE FILL;              /* unused
1513    /*
1514        alloc               ADDRESS;                /* link in list of allocated $VOLBs
1515    /*
1516    /* Storage for the following control blocks is allocated at the end of the VOLB
1517    /*
1518        fab         ADDRESS;                        /* pointer to RMS File Access Block
1519        rab         ADDRESS;                        /* pointer to RMS Record Access Block
1520        nam         ADDRESS;                        /* pointer to RMS Name Block
1521        esbuf       ADDRESS;                        /* pointer to RMS Expanded String Buffer
1522        rsbuf       ADDRESS;                        /* pointer to RMS Result String Buffer
1523    /*
1524        assoc_namb  ADDRESS;                        /* namb from the volume mount
1525    /*
1526    /* The rest of the block will be reset to nulls each time the block is allocated
1527    /*
1528        CONSTANT start_zero EQUALS .;
1529    /*
```

```
1530    /* Device characteristics variables (from $GETDVI or invented for virtual devices)
1531    /*
1532        devbufsiz   LONGWORD UNSIGNED;          /* device buffer size
1533        devchar     LONGWORD UNSIGNED;          /* device characteristics ($devdef symbols)
1534        devclass    LONGWORD UNSIGNED;          /* device class           ($dcdef (DC$) symbols)
1535        devdepend   LONGWORD UNSIGNED;          /* device dependent characteristics
1536        devnamlen   LONGWORD UNSIGNED;          /* device name length
1537        devtype     LONGWORD UNSIGNED;          /* device type            ($dcdef (DT$) symbols)
1538        devmaxblock LONGWORD UNSIGNED;          /* maximum number of blocks on device
1539        volmaxblock LONGWORD UNSIGNED;          /* maximum number of blocks on volume
1540    /*
1541    /* Other bits and pieces
1542    /*
1543        status STRUCTURE WORD UNSIGNED;         /* status flags for volume
1544            connected       BITFIELD MASK;      /* RMS Record stream connected to volume
1545            read_check      BITFIELD MASK;      /* /DATA_CHECK=READ
1546            write_check     BITFIELD MASK;      /* /DATA_CHECK=WRITE
1547            foreign         BITFIELD MASK;      /* Mounted as a foreign volume
1548            virtual         BITFIELD MASK;      /* Native mode file mounted as a virtual volume
1549            write           BITFIELD MASK;      /* Open volume for READ/WRITE access
```



```
1550            vfmt_explicit   BITFIELD MASK;      /* Volume format specified
1551            verified        BITFIELD MASK;      /* Directory structure has been verified
1552            END status;
1553        channel WORD UNSIGNED;                  /* channel assigned to device
1554    /*
1555    /* Some commands chain multiple nambs from a volb
1556    /*
1557        namb_head           ADDRESS;
1558    /*
1559    /* The following structure keeps a bit for each directory segment (bits 1:31) plus a single bit to flag whether
1560    /* directory write caching is active (bit 0).  This is an interim attempt to produce decent performance until
1561    /* full caching is implemented.  !??
1562    /*
1563        dircache STRUCTURE LONGWORD UNSIGNED;       /* complete directory cache longword
1564            dircache_active         BITFIELD MASK;  /* caching is in force
1565                                                    /* remaining bits map to directory segments
1566            END dircache;
1567    /*
1568    /* The following format specific longword would normally point to a structure
1569    /* whose composition depended on the FORMAT byte.
1570    /*
1571        vfmt_specific       ADDRESS;
1572    /*
1573        vol_format  BYTE UNSIGNED;                  /* constant volb$k_vfmt_xxx
1574            CONSTANT (
1575                vfmt_invalid,                       /* value 0, format not set to known value
1576                vfmt_dos11,                         /* DOS-11 magtape
1577                vfmt_files11,                       /* Files-11
1578    {\          vfmt_rtmt,                          /* RT-11 mag tape
1579                vfmt_rt11                           /* RT-11 disk
1580                 ) EQUALS 0 INCREMENT 1;
1581    /*
1582        CONSTANT vfmt_lobound EQUALS 0;                     /* low bound for case statement
1583        CONSTANT vfmt_hibound EQUALS volb$k_vfmt_rt11;      /* high bound
1584    /*
1585    /* The vol_type is a text string identifying the volume type, i.e. "RT-11" or "DOS-11"
1586    /*
1587        vol_type_len        LONGWORD UNSIGNED;      /* Length of the volume type
1588        vol_type            CHARACTER LENGTH 8;     /* Could only be "RT-11" or "DOS-11", doesn't need to be long
1589    /*
```

```
1590    /* Identification of the $VOLB.  This is a unique string which is used to find $VOLBs.  For foreign-mounted
1591    /* devices, this is the actual hardware device name, in the format "_Dxcn:"  (e.g. "_DYA0:") as returned by
1592    /* $GETDVI.  For virtual devices, this is the original input device string of the form "name:", which is
1593    /* treated as a logical name.  When a virtual device is created, the resultant file name string is stored in the
1594    /* vol_ident so that it can be signalled by the standard "volume has been initialized" message, therefore the
1595    /* vol_ident field is much longer than would normally be expected.
1596    /*
1597        vol_ident_len       LONGWORD UNSIGNED;      /* Length of the next field
1598        vol_ident           CHARACTER LENGTH 128;   /* Volume ident buffer
1599        devnam              CHARACTER LENGTH 64;    /* device name buffer
1600    /*
1601    /* Other items which would be in here if SDL supported external literals.  This space is allocated with the
1602    /* rest of the block.  Certain pointers above point to these buffers.  This area is part of the block, it
1603    /* is only due to restrictions in SDL that these are not actually defined here.
1604    /*
1605        /*  buf_fab         CHARACTER LENGTH #fab$k_bln;    /* RMS File access block
1606        /*  buf_rab         CHARACTER LENGTH #rab$k_bln;    /* RMS Record access block
1607        /*  buf_nam         CHARACTER LENGTH #nam$k_bln;    /* RMS Name block

                                                    15-SEP-1984 23:01:31.30     SDL V2.0                        Page   38
                                                    15-SEP-1984 22:44:23        $255$DUA28:[EXCHNG.SRC]EXCDEFS.SDL;1
1608        /*  buf_esbuf       CHARACTER LENGTH #nam$c_maxrss; /* RMS expanded name buffer
1609        /*  buf_rsbuf       CHARACTER LENGTH #nam$c_maxrss; /* RMS result name buffer
1610    /*
1611        CONSTANT "length" EQUALS .;                 /* length of data structure
1612
1613    END volb;
1614
1615
1616    END_MODULE;
```

EXCDEFS
LIS

EXCDIRE
LIS

EXCDOS11
LIS

EXCDBG
LIS