



\*\*FILE\*\*ID\*\*EXCDEFS

D 5

SSSSSSSS	DDDDDDDD	LL
SSSSSSSS	DDDDDDDD	LL
SS	DD	DD
SSSSSS	DD	DD
SSSSSS	DD	DD
SS	DD	DD
SSSSSSSS	DDDDDDDD	LLLLLLLL
SSSSSSSS	DDDDDDDD	LLLLLLLL

```
*****  
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
* ALL RIGHTS RESERVED.  
*  
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
* TRANSFERRED.  
*  
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
* CORPORATION.  
*  
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
*****
```

```
MODULE $exchdef;  
/*  
/* Version: 'V04-000'  
/*  
/*++  
/*  
/* FACILITY: EXCHANGE - Foreign volume interchange utility  
/*  
/* ABSTRACT: Data structure definitions for EXCHANGE.  
/*  
/* ENVIRONMENT: VAX/VMS operating system, unprivileged user-mode utility  
/*  
/* AUTHOR: CW Hobbs  
/*  
/* DATE: 12-July-1982 Last Edit:  
/*  
/* MODIFIED BY:  
/*  
/*--
```

```
/**+
/* Local constants which define the sizes of adjustable structures in $EXCHDEF
*/
/*-
#io_buf_blocks = 12;           /* must be 5 or more, at least 2 blocks
#rec_buf_size = 512;          /* larger than rec_buf_size, and LEQU 127

/**+
/* DMPDSV - Flag options for EXCH$DBG_DUMP_DOSV debugging routine
*/
/*-
AGGREGATE dmpdsv STRUCTURE PREFIX dmpdsv$;
    dmpdsv STRUCTURE BYTE UNSIGNED;
        volb      BITFIELD MASK;          /* print volb and dosv addresses
        status    BITFIELD MASK;          /* print status flags
        position   BITFIELD MASK;         /* print current position
        entries    BITFIELD MASK;         /* print the directory entries
    END dmpdsv;
END dmpdsv;

/**+
/* DOSNXT - Flag options for EXCH$DOS11_NEXT_ENTRY
*/
/*-
AGGREGATE dosnxt STRUCTURE PREFIX dosnxt$;
    dosnxt STRUCTURE BYTE UNSIGNED;
        rewind    BITFIELD MASK;          /* "rewind" to the start of the tape
        count_blocks BITFIELD MASK;       /* make sure that the block count is valid
    END dosnxt;
END dosnxt;

/**+
/* EXCHBLK - Constant definitions for block type codes
*/
/*-
CONSTANT
{
    copy,                      /* work area for COPY and TYPE verbs
    dire,                      /* DIRECTORY verb work area
    dos11,                     /* volume specifics for DOS-11
    dos11ctx,                  /* DOS-11 file context block
    excg,                      /* global data structure
    filb,                      /* file context block
    init,                      /* INIT verb work area
    moun,                      /* MOUNT verb work area
    namb,                      /* name block
    rmsb,                      /* exchblk$k_rmsb - code for file block
}
```

```
rt11,          /* volume specifics for RT-11
rt11ctx,       /* RT-11 file context block
volb,          /* volume block
rtnam         /* RT-11 name routine work area
)
EQUALS 255 INCREMENT -1 PREFIX exchblk$;

/**+
/* FINDBIN - Return status codes for PDP_FIND_BINARY_RECORD
*/
/*-
CONSTANT
(
success,        /* found a record
eob,            /* end of buffer, no record found
checksum,       /* checksum failed to match
too_big,        /* record larger than output buffer
bad_fmt         /* badly formed record
)
EQUALS 0 INCREMENT 1 PREFIX findbin$;

CONSTANT lobound EQUALS findbin$k_success PREFIX findbin$;
CONSTANT hibound EQUALS findbin$k_bad_fmt PREFIX findbin$;

/**+
/* FINDSTM - Return status codes for PDP_FIND_STREAM_RECORD
*/
/*-
CONSTANT
(
success,        /* found a record
ctrlz_eof,      /* ^Z at beginning of record
eob,            /* end of buffer, no record found
no_term,        /* end of buffer, partial record found
bad_fmt         /* record longer than output buffer
)
EQUALS 0 INCREMENT 1 PREFIX findstm$;

CONSTANT lobound EQUALS findstm$k_success PREFIX findstm$;
CONSTANT hibound EQUALS findstm$k_bad_fmt PREFIX findstm$;

/**+
/* PRSOPT - Flag options for EXCHSCMD_PARSE_FILESPEC
*/
/*-
AGGREGATE prsopt STRUCTURE PREFIX prsopt$;
prsopt STRUCTURE BYTE UNSIGNED;
    virtual_device BITFIELD MASK;           /* parse device name as a virtual device
    no_get_value   BITFIELD MASK;           /* skip CLISGET_VALUE call, use parameter name as value
END prsopt;
```

```
END prsopt;
```

```
/*+
/* RTNXT - Flag options for EXCH$RTACP_NEXT_ENTRY
/*-
AGGREGATE rtnxt STRUCTURE PREFIX rtnxt$;
  rtnxt STRUCTURE BYTE UNSIGNED;
    permanent      BITFIELD MASK;          /* look for permanent entries
    empty          BITFIELD MASK;          /* look for empty entries
    tentative     BITFIELD MASK;          /* look for tentative entries
    unknown        BITFIELD MASK;          /* look for invalid entries
    skip_check    BITFIELD MASK;          /* skip the check for moved entry
    skip_expand   BITFIELD MASK;          /* skip expanding the radix-50 name to ascii
  END rtnxt;
END rtnxt;
```

```

/*+
/*
/* SCOPY - copy block
/*
/* Contains the work area for the COPY and TYPE verbs
/*
/*-
```

AGGREGATE copy STRUCTURE PREFIX copy\$;

```

/* Dynamic character string descriptors for name components. These strings are reclaimed at the start of each copy
/*
default_filename QUADWORD UNSIGNED; /* desc for sticky name
/*
size WORD UNSIGNED; /* standard size of the block (symbol EXCHBLKSS_copy)
type BYTE UNSIGNED; /* type code field (symbol EXCHBLKSK_copy)
std_fill BYTE FILL; /* unused

/*
input_filename QUADWORD UNSIGNED; /* desc for current item from input parameter list
/*
output_filename QUADWORD UNSIGNED; /* desc for output parameter
/*
input_sticky_name QUADWORD UNSIGNED; /* name to make input files sticky
/*
q_boot QUADWORD UNSIGNED; /* qualifier /BOOT=string
q_fdl QUADWORD UNSIGNED; /* qualifier /FDL=string

/*
CONSTANT start_zero EQUALS .: /* here to end is cleared for each copy command

/*
Integer valued command qualifiers
/*
q_allocation LONGWORD UNSIGNED; /* qualifier /ALLOCATION=n (optional)
q_extension LONGWORD UNSIGNED; /* qualifier /EXTENSION=n (optional)
q_start_block LONGWORD UNSIGNED; /* qualifier /START_BLOCK=n

/*
Flags for boolean DCL command qualifiers
/*
qual_flags STRUCTURE LONGWORD UNSIGNED; /* the whole flags longword
(
q_best_try_contiguous BITFIELD MASK;
q_confirm BITFIELD MASK;
q_contiguous BITFIELD MASK;
q_delete BITFIELD MASK;
q_log BITFIELD MASK;
q_nolog_explicit BITFIELD MASK; /* we had an explicit /NOLOG
q_protect BITFIELD MASK; /* value for protect bit, not useful unless explicit also set
q_protect_explicit BITFIELD MASK; /* an explicit /PROTECT or /NOPROTECT was seen
q_replace BITFIELD MASK;
q_rewind BITFIELD MASK;
q_system BITFIELD MASK;
q_truncate BITFIELD MASK;
END qual_flags;

/*
Local copy command flags
/*
copy_flags STRUCTURE LONGWORD UNSIGNED; /* the whole flags longword
multiple_files BITFIELD MASK; /* the input potentially describes multiple files
type_command BITFIELD MASK; /* the command is TYPE rather than COPY

```

```

/* reopen_input BITFIELD MASK;
   reopen_in_progress BITFIELD MASK;
END copy_flags;

/*
 * Other misc data items
 */
max_rec           LONGWORD UNSIGNED;          /* current maximum output length
/*
 * Work area for the current input file
 */
inp_filb          ADDRESS;                   /* pointer to a filb for the input file
inp_namb          ADDRESS;                   /* pointer to a namb describing the input file
/*
 * Work area for the current output file
 */
out_filb          ADDRESS;                   /* pointer to a filb for the output file
out_namb          ADDRESS;                   /* pointer to a namb describing the output file
/*
CONSTANT end_zero EQUALS .;
CONSTANT "length" EQUALS .;
/* no need to clear the buffers (or create pages not to be used)
/* length of data structure

END copy;

```

```
/*
/* Generic file context block. This is the front part of both the RT11CTX and DOS11CTX blocks, and is used
/* by the PDP record routines shared by both RT-11 and DOS-11 formats
*/
AGGREGATE ctx STRUCTURE PREFIX ctx$;
/*
/* Links to the queues of in-use and available blocks, plus standard type and size fields
/*
    header STRUCTURE QUADWORD UNSIGNED;           /* queue header
        flink ADDRESS;                          /* forward link
        blink ADDRESS;                          /* reverse link
    END link;

/*
    size          WORD UNSIGNED;                /* standard size of the block (symbol EXCHBLKSS_dos11ctx)
    type          BYTE UNSIGNED;               /* type code field (symbol EXCHBLKSK_dos11ctx)
    std_fill      BYTE FILL;                  /* unused
/*
    alloc         ADDRESS;                   /* link to list of allocated blocks
/*
/* Addresses of associated control structures
/*
    assoc_filb   ADDRESS;                  /* address of the filb
    assoc_volb   ADDRESS;                  /* address of the volb
/*
/* I/O buffer for this record stream
/*
    buffer        ADDRESS;                  /* Pointer to the buffer
/*
    CONSTANT buffer_blocks EQUALS #io_buf_blocks; /* size of the buffer in blocks
    CONSTANT buffer_length EQUALS #io_buf_blocks*512; /* size of the buffer in bytes
/*
    CONSTANT start_zero EQUALS .;          /* here to end is cleared for each copy command
/*
/* Data to describe a record stream in the file.
/*
    cur_block     LONGWORD UNSIGNED;        /* pbn of current block
    eof_block     LONGWORD UNSIGNED;        /* pbn of last block in file
    cur_byte      LONGWORD UNSIGNED;        /* offset into the block
/*
/* Flags for an open file. This longword will be zero when file is not open
/*
    flags STRUCTURE LONGWORD UNSIGNED;      /* the whole flags longword
        stream_active BITFIELD MASK;        /* block currently describes an open file
        output_file   BITFIELD MASK;        /* block currently describes an output file
        flush         BITFIELD MASK;        /* tell advance routine to flush last block
        pointer       BITFIELD MASK;        /* tape position is at this file
    END flags;

/*
/* Pointers to the I/O buffer for this record stream
/*
    buf_base_block LONGWORD UNSIGNED;        /* Pbn of first block in buffer
    buf_high_block LONGWORD UNSIGNED;        /* Pbn of last block in buffer
    high_block_written LONGWORD UNSIGNED;    /* Pbn of highest block to the file
/*
    CONSTANT "length" EQUALS .;             /* length of data structure
```

EXCDEFS.SDL;1

16-SEP-1984 16:40:48.79 Page 8

END ctx;

```
/*
/* $DIRE - Directory block
/* Contains the work area for the DIRECTORY verb
*/
/*- AGGREGATE dire STRUCTURE PREFIX dire$;
/* filename QUADWORD UNSIGNED; /* file name parameter to command
/* size WORD UNSIGNED; /* standard size of the block (symbol EXCHBLKSS_dire)
/* type BYTE UNSIGNED; /* type code field (symbol EXCHBLKSK_dire)
/* std_fill BYTE FILL; /* unused
/* Dynamic character string descriptors for name components
/* output_file QUADWORD UNSIGNED; /* desc for /OUTPUT file name
/* default_name QUADWORD UNSIGNED; /* desc for sticky name
/* CONSTANT start_zero EQUALS .; /* here to end is cleared for each directory command
/* Some pointers to RMS structures
/* outrmsb ADDRESS; /* A $RMSB structure for the /OUTPUT file block
/* outfab ADDRESS; /* Ref to fab for output
/* outrab ADDRESS; /* Ref to rab for output
/* Integer valued command qualifiers
/* q_columns LONGWORD UNSIGNED; /* qualifier /COLUMNS=n (always present)
/* Command qualifier flags
/* qual_flags STRUCTURE LONGWORD UNSIGNED; /* the whole flags longword
{ \
    q_all BITFIELD MASK; /* qualifier /ALL
    q_badblocks BITFIELD MASK; /* qualifier /BADBLOCKS
    q_blocks BITFIELD MASK; /* qualifier /BLOCKS
    q_brief BITFIELD MASK; /* qualifier /BRIEF
    q_date BITFIELD MASK; /* qualifier /DATE
    q_deleted BITFIELD MASK; /* qualifier /DELETED
    q_free BITFIELD MASK; /* qualifier /FREE
    q_full BITFIELD MASK; /* qualifier /FULL
    q_octal BITFIELD MASK; /* qualifier /OCTAL
    q_output BITFIELD MASK; /* qualifier /OUTPUT=<filespec>
    q_owner BITFIELD MASK; /* qualifier /OWNER
    q_printer BITFIELD MASK; /* qualifier /PRINTER
    q_size BITFIELD MASK; /* qualifier /SIZE
    q_summary BITFIELD MASK; /* qualifier /SUMMARY
}
END qual_flags;
/* Local directory command flags
/* dire_flags STRUCTURE LONGWORD;
/* item_to_print BITFIELD MASK; /* an item is waiting to be printed
```

```
    items_printed BITFIELD MASK; /* one or more items have been printed
END dire_flags;
/*
/* Other misc data items
/*
item_width      LONGWORD UNSIGNED;      /* width of a single directory listing item
list_width       LONGWORD UNSIGNED;      /* maximum listing width
cur_width        LONGWORD UNSIGNED;      /* current listing width
cur_column       LONGWORD UNSIGNED;      /* current column in listing buffer
list_buffer      CHARACTER LENGTH 512;   /* buffer where listing lines are assembled
CONSTANT item_spacing EQUALS 3;
/*
CONSTANT "length" EQUALS .;           /* length of data structure
END dire;
```

```

/*+
/* $DOS11 - volume specific structure for an DOS-11 volume
/*
/* Contains all the DOS-11 specific control structures for a volume. This structure would
/* normally hang from VOLB$A_VFMT_SPECIFIC.
/*-
/*++

AGGREGATE dos11 STRUCTURE PREFIX dos11$;
    dummy          QUADWORD FILL;           /* filler to put the size and type fields in the right spot
/*
    size           WORD UNSIGNED;          /* standard size of the block      (symbol EXCHBLK$S_rmsb)
    type           BYTE UNSIGNED;          /* type code field                (symbol EXCHBLK$K_rmsb)
    std_fill       BYTE FILL;              /* unused
/*
    CONSTANT start_zero EQUALS .;
/*
/* Variables to describe the state of the DOS-11 device go here
/*
    status STRUCTURE WORD UNSIGNED;        /* complete status word
    position_valid BITFIELD MASK;          /* value in dos11$!current_file is valid
    beg_of_tape   BITFIELD MASK;          /* last operation rewound the magtape
    tape_mark     BITFIELD MASK;          /* magtape read found a tape mark
    end_of_tape   BITFIELD MASK;          /* magtape read found two consecutive tape marks
    directory     BITFIELD MASK;          /* magtape "directory" is complete and valid
    current_marked BITFIELD MASK;         /* file underneath the heads marked with '>'-
    error_rewind  BITFIELD MASK;          /* tape has been rewound due to error
END status;

/*
/* Remember which file on the tape was last read. The file at BOT is file #0. The current_file
/* number in conjunction with the position bits marks a location on the tape.
/*
    current_file   LONGWORD UNSIGNED;
/*
/* Directory entries are linked from here. This is really just a queue of the labels
/*
    entry_header STRUCTURE QUADWORD UNSIGNED; /* queue header
    entry_flink   ADDRESS;                 /* forward link
    entry_blink   ADDRESS;                 /* reverse link
END entry_header;

/*
/* The address of the current label for wildcard operations. Note that this will not necessarily correspond
/* to the current_file position.
/*
    current_entry  ADDRESS;               /* address of the current entry in the queue
/*
/* Various bits and pieces for IO operations on tapes
/*
    iosb STRUCTURE QUADWORD UNSIGNED;
    iosb_status   WORD UNSIGNED;          /* returned status
    counts UNION;
        iosb_skipcnt    WORD;              /* number of files or records skipped (signed)
        iosb_bytectn   WORD UNSIGNED;      /* size of transfer
    END counts;
    iosb_char     LONGWORD UNSIGNED;      /* characteristics buffer

```

```
END iosb;  
/* We need a scratch buffer to read in our tape labels  
/* label_buf CHARACTER LENGTH 14;           /* anything over 14 is invalid  
/* The routine EXCH$IO_DOS11_COUNT_BLOCKS stores the block count in the next field  
/* block_count WORD UNSIGNED;  
/* CONSTANT end_zero EQUALS ::  
CONSTANT "length" EQUALS ::                /* length of data structure  
END dos11;
```

```

/*
/* DOS-11 "directory" entry
*/
AGGREGATE dos11ent STRUCTURE PREFIX dos11ent$;
/*
/* Links to the rest of the entries for a tape
/*
    header STRUCTURE QUADWORD UNSIGNED;          /* queue header
        flink   ADDRESS;                         /* forward link
        blink   ADDRESS;                         /* reverse link
    END link;

/*
/* The entry info will be copied into the dos11ctx entry, put a marker
/* so that we can automatically compare the lengths
/*
    CONSTANT "start_entry" EQUALS .;

/*
/* Next seven words map directly onto the 14-byte mag tape label
/*
    filename_1      LONGWORD UNSIGNED;           /* 6 Radix-50 characters
    filetype        WORD UNSIGNED;                /* 3 Radix-50 characters
    uic STRUCTURE WORD UNSIGNED;                 /* User ID code
        member        BYTE UNSIGNED;               /* Member number
        group         BYTE UNSIGNED;               /* Group number
    END uic;
    protection      WORD UNSIGNED;                /* Protection code
    date            WORD UNSIGNED;                /* Creation date field
    filename_2      WORD UNSIGNED;                /* 3 Radix-50 characters

/*
/* Rest of fields are computed
/*
    blocks          WORD UNSIGNED;                /* Total length of file
    file_number     WORD UNSIGNED;                /* Position of file on tape, first file is #0

/*
    entry_status STRUCTURE WORD UNSIGNED;
        blocks_valid BITFIELD MASK;              /* The "BLOCKS" field is correct
    END entry_status;

/*
    CONSTANT "end_entry" EQUALS .;

/*
    CONSTANT "length" EQUALS .;                  /* length of data structure

END dos11ent;

```

```
/*
/* DOS-11 file context block This is a combination of the FAB, RAB and NAM block
/* for an DOS-11 random access file
*/

AGGREGATE dos11ctx STRUCTURE PREFIX dos11ctx$;
/*
/* Links to the queues of in-use and available blocks, plus standard type and size fields
/*
    header STRUCTURE QUADWORD UNSIGNED;           /* queue header
        flink   ADDRESS;                         /* forward link
        blink   ADDRESS;                         /* reverse link
    END link;

/*
    size          WORD UNSIGNED;                /* standard size of the block (symbol EXCHBLKSS_dos11ctx)
    type          BYTE UNSIGNED;               /* type code field (symbol EXCHBLKSK_dos11ctx)
    std_fill     BYTE FILL;                   /* unused
/*
    alloc         ADDRESS;                    /* link to list of allocated blocks
/*
/* Addresses of associated control structures
/*
    assoc_filb   ADDRESS;                  /* address of the filb
    assoc_volb   ADDRESS;                  /* address of the volb
/*
/* I/O buffer for this record stream
/*
    buffer        ADDRESS;                  /* Pointer to the buffer
/*
    CONSTANT start_zero EQUALS .:          /* here to end is cleared for each copy command
/*
/* Data to describe a record stream in the file.
/*
    cur_block    LONGWORD UNSIGNED;        /* pbn of current block
    eof_block    LONGWORD UNSIGNED;        /* pbn of last block in file
    cur_byte     LONGWORD UNSIGNED;        /* offset into the block
/*
/* Flags for an open file. This longword will be zero when file is not open
/*
    flags STRUCTURE LONGWORD UNSIGNED;      /* the whole flags longword
        stream_active BITFIELD MASK;        /* block currently describes an open file
        output_file   BITFIELD MASK;        /* block currently describes an output file
        flush         BITFIELD MASK;        /* tell advance routine to flush last block
        pointer       BITFIELD MASK;        /* tape position is at this file
    END flags;

/*
/* Pointers to the I/O buffer for this record stream
/*
    buf_base_block LONGWORD UNSIGNED;        /* Pbn of first block in buffer
    buf_high_block LONGWORD UNSIGNED;        /* Pbn of last block in buffer
    high_block_written LONGWORD UNSIGNED;    /* Pbn of highest block to the file
/*
/* The next part is a duplicate of the directory entry
/*
    current_entry  ADDRESS;                  /* the address of this entry in the "directory" queue
    entry_union UNION;
```

```
"entry"      CHARACTER LENGTH dos11ent$k_end_entry - dos11ent$k_start_entry; /* The overall entry
entry_fields STRUCTURE;
/*
/* Next seven words map directly onto the 14-byte mag tape label
/*
filename_1      LONGWORD UNSIGNED;      /* 6 Radix-50 characters
filetype        WORD UNSIGNED;        /* 3 Radix-50 characters
uic STRUCTURE WORD UNSIGNED;          /* User ID code
member          BYTE UNSIGNED;        /* Member number
group           BYTE UNSIGNED;        /* Group number
END uic;
protection      WORD UNSIGNED;        /* Protection code
date            WORD UNSIGNED;        /* Creation date field
filename_2      WORD UNSIGNED;        /* 3 Radix-50 characters
/*
/* Rest of fields are computed
/*
blocks          WORD UNSIGNED;        /* Total length of file
file_number      WORD UNSIGNED;        /* Position of file on tape, first file is #0
/*
entry_status    STRUCTURE WORD UNSIGNED;
blocks_valid    BITFIELD MASK;        /* The 'BLOCKS' field is correct
END entry_status;
END entry_fields;
END entry_union;
/*
/* Expanded name fields
/*
exp_fullname_len  LONGWORD UNSIGNED;  /* Length of the concatenated name
exp_name_len     LONGWORD UNSIGNED;  /* Length of the name field
exp_type_len     LONGWORD UNSIGNED;  /* Length of the file extension (type)
exp_directory    CHARACTER LENGTH 10; /* UIC format directory "[000,000]"
exp_fullname     CHARACTER LENGTH 13; /* Name in normal form, no embedded blanks
exp_name          CHARACTER LENGTH 9; /* Blank padded name
exp_type          CHARACTER LENGTH 3; /* Blank padded extension
exp_date          CHARACTER LENGTH 11; /* VMS format ascii date
/*
CONSTANT end_zero EQUALS :;
CONSTANT "length" EQUALS :;           /* length of data structure
END dos11ctx;
```

```
/**+
/* $EXCG - EXCHANGE Global data
/*
/* Contains all the global data necessary for the EXCHANGE facility
/*
/*-
/* AGGREGATE excg STRUCTURE PREFIX excg$;
/*
/* Flags for command line qualifiers and local stat
/*
  flags STRUCTURE WORD UNSIGNED;           /* whole flags word
    control_c      BITFIELD MASK;          /* Control/C AST has been received
    q_cache        BITFIELD MASK;          /* EXCHANGE /CACHE is in effect for RT-11
    q_message      BITFIELD MASK;          /* EXCHANGE /MESSAGE is in effect
    foreign_command BITFIELD MASK;         /* exchange is running in single-command mode
    exiting        BITFIELD MASK;          /* exit handler is active
  END flags;

/*
/* Channel for input terminal, used for setting control/c asts.  0 if input is not a terminal device
/*
  tt_channel      WORD UNSIGNED;
/*
  copy_work       ADDRESS;                /* pointer to the work area for COPY and TYPE
/*
  size            WORD UNSIGNED;          /* standard size of the block (symbol EXCHBLK$S_EXCG)
  type            BYTE UNSIGNED;          /* type code field (symbol EXCHBLK$K_EXCG)
  std_fill        BYTE FILL;              /* unused
/*
/* Work areas for the individual command verbs (1 of these at top for alignment)
/*
  dire_work       ADDRESS;                /* pointer to the work area for DIRECTORY
  init_work       ADDRESS;                /* pointer to the work area for INIT
  moun_work       ADDRESS;                /* pointer to the work area for MOUNT
  rtnam_work      ADDRESS;                /* pointer to the work area for DELETE and RENAME
/*
/* Process identification items
/*
  uic_member      WORD UNSIGNED;          /* binary member number for uic directories
  uic_group       WORD UNSIGNED;          /* binary group number for uic directories
  username        CHARACTER LENGTH 12;    /* process' username
/*
/* Exit handler for per-command activity, exh_routine is 0 if no handler is declared
/*
  exit_block STRUCTURE;
    exh_flink      ADDRESS;                /* system-maintained forward link
    exh_routine    ADDRESS;                /* address of exit handling routine
    exh_arg_count  LONGWORD UNSIGNED;     /* number of arguments for routine
    exh_status     ADDRESS;                /* address to store status code, reason for exit
    exh_volv      ADDRESS;                /* exit handler specific argument (volb address)
    exh_condvalu  LONGWORD UNSIGNED;     /* put the status code here
  END exit_block;
/*
/* Exit handler for global /CACHE actions
/*-
```

```

cachexit_block STRUCTURE:
  cachexh_flink ADDRESS;
  cachexh_routine ADDRESS;
  cachexh_arg_count LONGWORD UNSIGNED;
  cachexh_status ADDRESS;
  cachexh_condvalu LONGWORD UNSIGNED;
END cachexit_block;

/*
/* Queue headers for managed resources
*/
dos11ctx_alloc ADDRESS;
dos11ctx_use STRUCTURE QUADWORD UNSIGNED;
  dos1Tctx_use_flink ADDRESS;
  dos11ctx_use_blink ADDRESS;
END dos1Tctx_use;
dos11ctx_avl STRUCTURE QUADWORD UNSIGNED;
  dos1Tctx_avl_flink ADDRESS;
  dos11ctx_avl_blink ADDRESS;
END dos1Tctx_avl;

/*
filb_alloc ADDRESS;
filb_use STRUCTURE QUADWORD UNSIGNED;
  filb_use_flink ADDRESS;
  filb_use_blink ADDRESS;
END filb_use;
filb_avl STRUCTURE QUADWORD UNSIGNED;
  filb_avl_flink ADDRESS;
  filb_avl_blink ADDRESS;
END filb_avl;

/*
namb_alloc ADDRESS;
namb_use STRUCTURE QUADWORD UNSIGNED;
  namb_use_flink ADDRESS;
  namb_use_blink ADDRESS;
END namb_use;
namb_avl STRUCTURE QUADWORD UNSIGNED;
  namb_avl_flink ADDRESS;
  namb_avl_blink ADDRESS;
END namb_avl;

/*
rmsb_alloc ADDRESS;
rmsb_use STRUCTURE QUADWORD UNSIGNED;
  rmsb_use_flink ADDRESS;
  rmsb_use_blink ADDRESS;
END rmsb_use;
rmsb_avl STRUCTURE QUADWORD UNSIGNED;
  rmsb_avl_flink ADDRESS;
  rmsb_avl_blink ADDRESS;
END rmsb_avl;

/*
rt11ctx_alloc ADDRESS;
rt11ctx_use STRUCTURE QUADWORD UNSIGNED;
  rt1Tctx_use_flink ADDRESS;
  rt11ctx_use_blink ADDRESS;
END rt1Tctx_use;
rt11ctx_avl STRUCTURE QUADWORD UNSIGNED;
/* Head of list of all allocated $rt11ctxs
/* Head of queue of $rt11ctxs in-use queue
/* Forward link
/* Backward link
/* Head of queue of all available $rt11ctxs
/* Forward link
/* Backward link
/* Head of list of all allocated $filBs
/* Head of queue of file blocks in use queue
/* Forward link
/* Backward link
/* Head of queue of all available $filBs
/* Forward link
/* Backward link
/* Head of list of all allocated $NAMBs
/* Head of queue of name blocks in use queue
/* Forward link
/* Backward link
/* Head of queue of all available $NAMBs
/* Forward link
/* Backward link
/* Head of list of all allocated $RMSBs
/* Head of queue of $RMSBs in-use queue
/* Forward link
/* Backward link
/* Head of queue of all available $RMSBs
/* Forward link
/* Backward link
/* Head of list of all allocated $dos11ctxs
/* Head of queue of $dos11ctxs in-use queue
/* Forward link
/* Backward link
/* put the status code here

```

```
    rt11ctx_avl_flink      ADDRESS:      /* Forward link
    rt11ctx_avl_blink      ADDRESS:      /* Backward link
/*                                         END rt11ctx_avl;

/* volb_alloc      ADDRESS:      /* Head of list of all allocated $VOLBs
volb_use STRUCTURE QUADWORD UNSIGNED; /* Head of queue of volume blocks in use queue
    volb_use_flink      ADDRESS:      /* Forward link
    volb_use_blink      ADDRESS:      /* Backward link
/*                                         END volb_use;
volb_avl STRUCTURE QUADWORD UNSIGNED; /* Head of queue of all available $VOLBs
    volb_avl_flink      ADDRESS:      /* Forward link
    volb_avl_blink      ADDRESS:      /* Backward link
/*                                         END volb_avl;

/* RMS data structures for command output, these point to memory at the end of the block
/*
sysout_fab      ADDRESS:      /* pointer to RMS File access block
sysout_rab      ADDRESS:      /* pointer to RMS Record access block
sysout_nam      ADDRESS:      /* pointer to RMS Name block
sysout_ebuf     ADDRESS:      /* pointer to RMS expanded name buffer
sysout_rbuf     ADDRESS:      /* pointer to RMS result name buffer
/*
/* Buffer for formatting
/*
fao_buffer      CHARACTER LENGTH 258; /* used by util_fao_buffer
/*
Other items which would be in here if SDL supported external literals. This space is allocated with the
/* rest of the block. Certain pointers above point to these buffers. This area is part of the block, it
/* is only due to restrictions in SDL that these are not actually defined here.
/*
/* buf_sysout_fab  CHARACTER LENGTH #fab$k_bln; /* RMS File access block
/* buf_sysout_rab   CHARACTER LENGTH #rab$k_bln; /* RMS Record access block
/* buf_sysout_nam   CHARACTER LENGTH #nam$k_bln; /* RMS Name block
/* buf_sysout_ebuf  CHARACTER LENGTH #nam$e_maxrss; /* RMS expanded name buffer
/* buf_sysout_rbuf  CHARACTER LENGTH #nam$e_maxrss; /* RMS result name buffer
/*
CONSTANT "length" EQUALS .;           /* length of data structure

END excg;
```

```
/*
/* $FILB - File Block
/* Contains data to describe a particular active file
/*-
/* AGGREGATE filb STRUCTURE PREFIX filbs;
/*
/* Links to the queues of in-use and available blocks, plus standard type and size fields
/*
  header STRUCTURE QUADWORD UNSIGNED;           /* queue header
    flink  ADDRESS;                            /* forward link
    blink  ADDRESS;                            /* reverse link
  END link;
  size      WORD UNSIGNED;                    /* standard size of the block (symbol EXCHBLK$S_NAMB)
  type      BYTE UNSIGNED;                   /* type code field (symbol EXCHBLK$K_NAMB)
  std_fill  BYTE FILL;                      /* unused
/*
  alloc     ADDRESS;                        /* link to list of allocated blocks
/*
/* The following name string has several uses depending on the routine. For example, it is used to hold the
/* result name from LIB$FIND_FILE routine for EXCH$FIL11_OPEN_FILE.
/*
  name_string QUADWORD UNSIGNED;
/*
/* The rest of the block will be reset to nulls each time the block is allocated
/*
  CONSTANT start_zero EQUALS ..;
/*
/*
  assoc_namb   ADDRESS;                     /* namb for this file
  assoc_volv   ADDRESS;                    /* address of mounted volb for this name, 0 for Files-11
/*
  context      ADDRESS;                   /* pointer to volume-specific structure defining file context
                                         /* DOS11CTX structure for DOS-11 tape files
                                         /* RT11CTX structure for RT-11 disk files
                                         /* RMSB for Files-11 files
  fil11_wcc    ADDRESS;                   /* wildcard context for LIB$FIND_FILE routine for Files-11 files
/*
/* The record format for this file (the volume format byte is in the assoc_namb)
/*
  rec_format   BYTE UNSIGNED;             /* record format code, a constant filb$k_rfmt_xxx
  CONSTANT (
    rfmt_invalid,                         /* value 0, format not set to known value
    rfmt_binary,                          /* formatted binary records
    rfmt_fixed,                           /* fixed length records
    rfmt_stream                           /* stream format records
  ) EQUALS 0 INCREMENT 1;
/*
  CONSTANT rfmt_lobound EQUALS 0;          /* low bound for case statement
  CONSTANT rfmt_hibound EQUALS filb$k_rfmt_stream; /* high bound
/*
  transfer_mode  BYTE UNSIGNED;           /* transfer mode code, a constant filb$k_xfrm_xxx
```

```

CONSTANT (
    xfrm_automatic,
    xfrm_block,
    xfrm_record
) EQUALS 0 INCREMENT 1;
/*
/* The carriage control for this file
/*
    car_control      BYTE UNSIGNED:           /* carriage control code, a constant filb$k_cctl_xxx
    CONSTANT (
        cctl_cr,
        cctl_fortran,
        cctl_none
    ) EQUALS 0 INCREMENT 1;
/*
    CONSTANT cctl_lobound EQUALS 0;          /* low bound for case statement
    CONSTANT cctl_hibound EQUALS filb$k_cctl_none; /* high bound
/*
/* Record attribute and other misc flags
/*
    flags STRUCTURE WORD UNSIGNED:          /* the whole flags word
        rfmt_explicit      BITFIELD MASK;   /* record format specified
        cctl_explicit      BITFIELD MASK;   /* carriage control specified
        file_erased         BITFIELD MASK;   /* file has been erased (e.g. due to write errors)
        files_found         BITFIELD MASK;   /* one or more files have been found using this filb
        files_created       BITFIELD MASK;   /* one or more files have been created using this filb
        explicit_version    BITFIELD MASK;   /* file name contained an explicit version
        delete_previous     BITFIELD MASK;   /* delete previous version of RT11 file during close
        name_change          BITFIELD MASK;   /* the output name was changed from the input
        q_best_try_contiguous BITFIELD MASK; /* there is a /BEST on the input file
        q_contiguous         BITFIELD MASK;   /* there is a /CONT on the input file
        q_truncate          BITFIELD MASK;   /* there is a /TRUN on the input file
    END flags;

/*
    q_allocation        LONGWORD UNSIGNED;  /* a /ALLOC was on the input file
    q_extension         LONGWORD UNSIGNED;  /* a /EXTEN was on the input file
    fixed_len           LONGWORD UNSIGNED;  /* length of FIXED records
    pad_char             BYTE UNSIGNED;    /* pad character, default 0
/*
    result_name_len     LONGWORD UNSIGNED;  /* length of the result name
/*
    block_count          LONGWORD UNSIGNED;  /* size of an existing file
/*
/* Length of record, record address
/*
    record_len           LONGWORD UNSIGNED;  /* length of current record
    record               ADDRESS;           /* address of current record
/*
/* Addresses of action routines for this file
/*
    close_routine        ADDRESS;           /* address of routine to close the file
    delete_routine       ADDRESS;           /* address of routine to delete the file
    get_routine          ADDRESS;           /* routine to get records, 0 for output-only files
    put_routine          ADDRESS;           /* routine to put records, 0 for input-only files
/*
    CONSTANT end_zero EQUALS .;

```

```
/*
/* No need to zero all the buffers
/*
    result_name      CHARACTER LENGTH 256; /* buffer for the result name
/*
    and actual record for when we need move mode
/*
    record_buffer    CHARACTER LENGTH #rec_buf_size;
    guard_byte       BYTE FILL;           /* eof logic much easier if we can write 1 byte past end of buffer
    CONSTANT "length" EQUALS .;          /* length of data structure
END filb;
```

```
/*+
/* $INIT - Init block
/* Contains the work area for the INIT verb
/*-
/* AGGREGATE init STRUCTURE PREFIX init$;
/*
    namb          ADDRESS;           /* pointer to our name block
    volb          ADDRESS;           /* pointer to our volume block
/*
    size          WORD UNSIGNED;    /* standard size of the block (symbol EXCHBLK$S_INIT)
    type          BYTE UNSIGNED;    /* type code field (symbol EXCHBLK$K_INIT)
    std_fill      BYTE FILL;        /* unused
/*
/* Character string descriptors for name components
/*
    device        QUADWORD UNSIGNED; /* desc for device name, or filename for INIT /CREATE
    volumeid      QUADWORD UNSIGNED; /* desc for volume id (for RT-11)
/*
/* Command qualifier variables and flags
/*
    q_allocation   LONGWORD UNSIGNED; /* size of virtual device to create /ALLOC=n
    q_extra_words  LONGWORD UNSIGNED; /* number of extra words for directory entries /EXTRA=n
    q_segments     LONGWORD UNSIGNED; /* number of directory segments /SEGMENTS=n
    flags STRUCTURE LONGWORD UNSIGNED; /* the whole flags longword
    {
        q_create      BITFIELD MASK; /* /CREATE
        q_message     BITFIELD MASK; /* /MESSAGE
        q_badblocks   BITFIELD MASK; /* /BADBLOCKS
        q_badblocks_retain BITFIELD MASK; /* /BADBLOCKS=RETAIN
        q_replace     BITFIELD MASK; /* /REPLACE
        q_replace_retain BITFIELD MASK; /* /REPLACE=RETAIN
    } END flags;
/*
    CONSTANT "length" EQUALS .;      /* length of data structure
END init;
```

```
/*+
/* $MOUN - Mount block
/*
/* Contains the work area for the MOUNT verb
/*-
/* AGGREGATE moun STRUCTURE PREFIX moun$;
/*
    namb          ADDRESS;           /* pointer to our name block
    volb          ADDRESS;           /* pointer to our volume block
/*
    size          WORD UNSIGNED;    /* standard size of the block (symbol EXCHBLK$S_MOUN)
    type          BYTE UNSIGNED;   /* type code field (symbol EXCHBLK$K_MOUN)
    std_fill      BYTE FILL;        /* unused
/*
/* Character string descriptors for name components
/*
    device        QUADWORD UNSIGNED; /* desc for device name
    filename      QUADWORD UNSIGNED; /* desc for virtual device input file name
/*
/* Command qualifier flags
/*
    q_write       LONGWORD UNSIGNED; /* write protection on the device (cli$_present tested)
    flags STRUCTURE LONGWORD UNSIGNED; /* the whole flags longword
    q_read_check  BITFIELD MASK;    /* /DATA_CHECK=READ
    q_write_check  BITFIELD MASK;   /* /DATA_CHECK=WRITE
    q_foreign     BITFIELD MASK;    /* /FOREIGN device mount
    q_virtual     BITFIELD MASK;   /* /VIRTUAL virtual device mount
    q_message     BITFIELD MASK;   /* /MESSAGE was given
END flags;
/*
    CONSTANT "length" EQUALS .;      /* length of data structure
END moun;
```

```

/*+
/* $NAMB - File Name Block
/*
/* Contains a parsed file specification
/*
/*-
AGGREGATE namb STRUCTURE PREFIX namb$;
/*
/* Links to the queues of in-use and available blocks, plus standard type and size fields
/*
    header STRUCTURE QUADWORD UNSIGNED;           /* queue header
    flink   ADDRESS;                            /* forward link
    blink   ADDRESS;                            /* reverse link
    END link;
    size     WORD UNSIGNED;                      /* standard size of the block (symbol EXCHBLK$S_NAMB)
    type    BYTE UNSIGNED;                      /* type code field (symbol EXCHBLK$K_NAMB)
    std_fill BYTE FILL;                          /* unused
/*
    alloc      ADDRESS;                         /* link to list of allocated blocks
/*
/* Dynamic character string descriptors for name strings
/*
    input      QUADWORD UNSIGNED;                /* desc for input name
    fullname   QUADWORD UNSIGNED;                /* desc for most complete (exp or res)
    expanded   QUADWORD UNSIGNED;                /* desc for expanded name
    result     QUADWORD UNSIGNED;                /* desc for result name
    device_dvi QUADWORD UNSIGNED;                /* desc for canonical device
/*
/* Descriptors of individual components of name. These might be fixed descriptors pointing into one of the
/* above strings (usual case), or they might be dynamic copies of the components (the namb was cloned).
/*
    node      QUADWORD UNSIGNED;                /* desc for node name
    device    QUADWORD UNSIGNED;                /* desc for device name
    directory QUADWORD UNSIGNED;                /* desc for directory name
    name     QUADWORD UNSIGNED;                /* desc for name name
    type     QUADWORD UNSIGNED;                /* desc for type name
    version   QUADWORD UNSIGNED;                /* desc for version name
/*
/* The rest of the block will be reset to nulls each time the block is allocated. The routine EXCH$CMD_NAMB_CLONE
/* copies the rest of the block to a new namb when it clones the namb.
/*
    CONSTANT start_zero EQUALS .;
/*
/* Carry some info from the RMS structures
/*
    fabdev        LONGWORD UNSIGNED;            /* device characteristics from FABSL_DEV
    nameflags STRUCTURE LONGWORD UNSIGNED;       /* the whole flags longword
    wildcard      BITFIELD MASK;               /* some field contains a wildcard rms's [nam$v_wildcard]
    wild_name     BITFIELD MASK;               /* file name contains a wildcard rms's [nam$v_wild_name]
    wild_type     BITFIELD MASK;               /* file type contains a wildcard rms's [nam$v_wild_type]
    wild_version  BITFIELD MASK;               /* file version has a wildcard rms's [nam$v_wild_ver]
    wild_group    BITFIELD MASK;               /* group number is a wildcard rms's [nam$v_wild_grp]

```

```

    wild_member           BITFIELD MASK: /* member number is a wildcard      rms's [nam$v_wild_mbr]
/*
    explicit_node        BITFIELD MASK: /* device name is explicit          rms's [nam$v_node]
    explicit_device      BITFIELD MASK: /* device name is explicit          rms's [nam$v_exp_dev]
    explicit_directory   BITFIELD MASK: /* directory name is explicit       rms's [nam$v_exp_dir]
    explicit_name         BITFIELD MASK: /* file name is explicit           rms's [nam$v_exp_name]
    explicit_type         BITFIELD MASK: /* file type is explicit           rms's [nam$v_exp_type]
    explicit_version     BITFIELD MASK: /* an explicit version number      rms's [nam$v_exp_ver]
/*
    concealed_device     BITFIELD MASK: /* device name is a concealed dev rms's [nam$v_cncl_dev]
    rooted_directory     BITFIELD MASK: /* device name has a root dir     rms's [nam$v_root_dir]
    uic_directory        BITFIELD MASK: /* directory is [GROUP, MEMBER]   rms's [nam$v_grp_mbr]
/*
    more_files           BITFIELD MASK: /* CLI$GET_VALUE gave us the CLI$ COMMA status
    bad_pdp_char         BITFIELD MASK: /* a char in the name or type field isn't valid for RT11 or DOS11
    dos_truncate          BITFIELD MASK: /* name or type too long for DOS11
    rt_truncate          BITFIELD MASK: /* name or type too long for RT11
END nameflags;

/*
    next                 ADDRESS:          /* Some commands link multiple nambs off this pointer
    assoc_volt           ADDRESS:          /* address of mounted volb referencing this name
/*
/* Keep a copy of the volume and record formats. Since both /VOLUME and /RECORD are local qualifiers, it is only
/* when we fetch the parameters that we can see any explicit qualifiers. Implied volume formats can be done when
/* we see the parameter. Implied record formats can't be done until we have the fully expanded filename
/*
    devclass              BYTE UNSIGNED;    /* device class (a DC$_xxx symbol)
    devtype               BYTE UNSIGNED;    /* device type (a DT$_xxx symbol)
    vol_format             BYTE UNSIGNED;    /* volume format code, a constant volb$k_vfmt_xxx
    rec_format             BYTE UNSIGNED;    /* record format code, a constant filb$k_rfmt_xxx
    transfer_mode          BYTE UNSIGNED;    /* transfer mode code, a constant filb$k_xfrm_xxx
    car_control            BYTE UNSIGNED;    /* carriage control code, a constant filb$k_cctl_xxx
    fixed_len              LONGWORD UNSIGNED; /* length of FIXED records
    pad_char               BYTE UNSIGNED;    /* pad character, default 0
    uic_member              BYTE UNSIGNED;    /* binary member number for uic directories
    uic_group               BYTE UNSIGNED;    /* binary group number for uic directories
    flags STRUCTURE BYTE UNSIGNED;
        rfmt_explicit        BITFIELD MASK;  /* record format specified
        cctl_explicit        BITFIELD MASK;  /* carriage control specified
        vfmt_explicit        BITFIELD MASK;  /* volume format specified
    END flags;

/*
/* Unique volume identifier
/*
    vol_ident_len          LONGWORD UNSIGNED; /* Length of the next field. Used for text display only,
                                                /* comparisions look at the entire vol_ident field
    vol_ident               CHARACTER LENGTH 128; /* needs to be as long as a logical name
/*
    CONSTANT "length" EQUALS ..;                  /* length of data structure

END namb;

```

\*\*

```
/*
/* $RMSB - RMS structures for a file
/*
/* Contains all the RMS control structures for a file.
/*
/*-
```

AGGREGATE rmsb STRUCTURE PREFIX rmsb\$;

```
/*
/* Links to the queues of in-use and available blocks, plus standard type and size fields
/*
    header STRUCTURE QUADWORD UNSIGNED;           /* queue header
        flink ADDRESS;                          /* forward link
        blink ADDRESS;                          /* reverse link
    END link;
    size WORD UNSIGNED;                      /* standard size of the block      (symbol EXCHBLK$S_rmsb)
    type BYTE UNSIGNED;                     /* type code field                (symbol EXCHBLK$K_rmsb)
    std_fill BYTE FILL;                     /* unused
/*
    alloc ADDRESS;                         /* link to list of allocated blocks
/*
/* Storage for the following control blocks is allocated at the end of the block
/*
    fab ADDRESS;                           /* pointer to RMS File Access Block
    rab ADDRESS;                           /* pointer to RMS Record Access Block
    nam ADDRESS;                           /* pointer to RMS Name Block
    esbuf ADDRESS;                          /* pointer to RMS Expanded String Buffer
    rsbuf ADDRESS;                          /* pointer to RMS Result String Buffer

/*
/* The rest of the block will be reset to nulls each time the block is allocated
/*
    CONSTANT start_zero EQUALS .;

/*
/* Other items which would be in here if SDL supported external literals. This space is allocated with the
/* rest of the block. Certain pointers above point to these buffers. This area is part of the block, it
/* is only due to restrictions in SDL that these are not actually defined here.
/*
    /* buf_fab CHARACTER LENGTH #fab$k_bln;    /* RMS File access block
    /* buf_rab CHARACTER LENGTH #rab$k_bln;    /* RMS Record access block
    /* buf_nam CHARACTER LENGTH #nam$k_bln;    /* RMS Name block
    /* buf_esbuf CHARACTER LENGTH #nam$c_maxrss; /* RMS expanded name buffer
    /* buf_rsbuf CHARACTER LENGTH #nam$c_maxrss; /* RMS result name buffer

/*
    CONSTANT "length" EQUALS .;             /* length of data structure

END rmsb;
```

```
/**+
/* $RT11 - volume specific structure for an RT-11 volume
/* Contains all the RT-11 specific control structures for a volume. This structure would
/* normally hang from VOLBSA_VFMT_SPECIFIC.
/*-
/* rt_dirseg = 1024;           /* Size of an RT11 directory segment
/* rt_root   = 6;             /* Directory starts on block 6 (pbn)

AGGREGATE rt11 STRUCTURE PREFIX rt11$;

    dummy          QUADWORD FILL;      /* spacer to put size and type in correct place
/*
    size           WORD UNSIGNED;     /* standard size of the block (symbol EXCHBLK$S_rmsb)
    type           BYTE UNSIGNED;    /* type code field (symbol EXCHBLK$K_rmsb)
    std_fill       BYTE FILL;        /* unused

/*
    CONSTANT start_zero EQUALS .;
/*
    Variables to describe RT-11 go here
/*
    status STRUCTURE WORD UNSIGNED; /* complete status word
        dir_present BITFIELD MASK; /* directory is resident (up to highest block in use)
    END status;
/*
    CONSTANT end_zero EQUALS .;

/*
***** From here to the end of the structure are read in the
/* first few blocks of the disk, containing the home block
/* and the entire directory
/*
***** Physical block 0 contains the boot block info
/*
    block_0        CHARACTER LENGTH 512;
/*
    Physical block 1 contains the volume home block
/*
    block_1 overlay UNION;
        block_1        CHARACTER LENGTH 512;
        home_block STRUCTURE;
            replace_tbl CHARACTER LENGTH 130;
            fill_00        WORD FILL;
            init_data     CHARACTER LENGTH 38;
            bup_data      CHARACTER LENGTH 18;
            fill_01        CHARACTER LENGTH 260 FILL;
            rte_name      WORD UNSIGNED;
            rte_block     WORD UNSIGNED;
            fill_02        CHARACTER LENGTH 14 FILL;
            cluster       WORD UNSIGNED;
```

```
first_seg WORD UNSIGNED;
system_vers WORD UNSIGNED;
volume_id CHARACTER LENGTH 12;
owner_name CHARACTER LENGTH 12;
system_id CHARACTER LENGTH 12;
fill_03 WORD FILL;
checksum WORD UNSIGNED;
END home_block;
END block_1_overlay;
/*
/* Physical blocks 2-5 are reserved.
/*
block_2 CHARACTER LENGTH 512;
block_3 CHARACTER LENGTH 512;
block_4 CHARACTER LENGTH 512;
block_5 CHARACTER LENGTH 512;
/*
/* Directory segments start here
/*
dire_segments CHARACTER LENGTH #rt_dirseg*31;
/*
CONSTANT "length" EQUALS .;           /* Length of data structure
CONSTANT dirseglen EQUALS #rt_dirseg;
CONSTANT root_block EQUALS #rt_root;
END rt11;
```

AGGREGATE rt11hom STRUCTURE PREFIX rt11hom\$;

```

replace_tbl CHARACTER LENGTH 130;           /* bad block replacement table
fill_00 WORD FILL;
init_data CHARACTER LENGTH 38;             /* INITIALIZE /RESTORE data area
bup_data CHARACTER LENGTH 18;              /* BUP data area
filt_01 CHARACTER LENGTH 260 FILL;

```

```
rte_name WORD UNSIGNED;          /* Name of RTE to mark RTEM volume
rte_block WORD UNSIGNED;          /* RTEM block number for first user file
fill_02 CHARACTER LENGTH 14 FILL; /* Pack cluster size
cluster WORD UNSIGNED;           /* PBN of first directory segment
first_seg WORD UNSIGNED;         /* System version number
system_vers WORD UNSIGNED;       /* Volume id
volume_id CHARACTER LENGTH 12;   /* Owner name
owner_name CHARACTER LENGTH 12;  /* System ID
system_id CHARACTER LENGTH 12;   /* Checksum (not really used)
fill_03 WORD FILL;              /* Length of data structure
checksum WORD UNSIGNED;
/* CONSTANT "length" EQUALS ..;
END rt11hom;
```

```
/*
/* Define various RT11 directory structures. These will not be independent
/* structures, but will be mapped over the directory segments as needed.
*/

/* RT-11 directory segment header

AGGREGATE rt11hdr STRUCTURE PREFIX rt11hdr$;
/*
    num_segs          WORD UNSIGNED;           /* Total number of segments in this directory
    next_seg          WORD UNSIGNED;          /* Segment number of next logical segment
    high_seg          WORD UNSIGNED;          /* Highest segment in use - valid for segment 1 only
    extra_bytes       WORD UNSIGNED;          /* Extra bytes per directory entry
    start_block       WORD UNSIGNED;          /* Pbn where stored data begins
/*
    CONSTANT "length" EQUALS .:              /* length of data structure

END rt11hdr;

/*
/* RT-11 directory entry
*/

AGGREGATE rt11ent STRUCTURE PREFIX rt11ent$;
/*
    fill_00            BYTE FILL;
    type_union UNION;
        type_byte      BYTE UNSIGNED;          /* Type of directory entry
        type           BITFIELD LENGTH 4;      /* Entry is integer in first part
        type_bits STRUCTURE;
            typ_tentative BITFIELD MASK;        /* Tentative file
            typ_empty     BITFIELD MASK;        /* Empty entry
            typ_permanent BITFIELD MASK;        /* Permanent file
            typ_end_segment BITFIELD MASK;       /* Last (possibly incomplete) entry in segment
            typ_fill_00    BITFIELD LENGTH 3 FILL; /* FILL
            typ_protected  BITFIELD MASK;        /* File is protected
        END type_bits;
    END type_union;
    filename          LONGWORD UNSIGNED;      /* 6 Radix-50 characters
    filetype          WORD UNSIGNED;          /* 3 Radix-50 characters
    blocks            WORD UNSIGNED;          /* Total length of file
    channel           BYTE UNSIGNED;          /* <not meaningful for EXCHANGE>
    job               BYTE UNSIGNED;          /* <not meaningful for EXCHANGE>
    date STRUCTURE WORD UNSIGNED;           /* Creation date field
        year             BITFIELD LENGTH 5;   /* Years since 1972
        day              BITFIELD LENGTH 5;   /* In decimal, 1-31
        month            BITFIELD LENGTH 5;   /* In decimal, 1-12
    END date;
/*
    CONSTANT "length" EQUALS .:              /* length of data structure

END rt11ent;
```

```
/*
/* RT-11 file context block. This is a combination of the FAB, RAB and NAM block
/* for an RT11 random access file
*/

AGGREGATE rt11ctx STRUCTURE PREFIX rt11ctx$;
/*
/* Links to the queues of in-use and available blocks, plus standard type and size fields
/*
    header STRUCTURE QUADWORD UNSIGNED;           /* queue header
        flink    ADDRESS;                         /* forward link
        blink    ADDRESS;                         /* reverse link
    END link;

/*
    size          WORD UNSIGNED;                /* standard size of the block      (symbol EXCHBLKSS_rt11ctx)
    type         BYTE UNSIGNED;                /* type code field               (symbol EXCHBLKSK_rt11ctx)
    std_fill     BYTE FILL;                   /* unused
/*
    alloc         ADDRESS;                    /* link to list of allocated blocks

/*
/* Addresses of associated control structures
/*
    assoc_filb   ADDRESS;                  /* address of the filb
    assoc_volb   ADDRESS;                  /* address of the volb

/*
/* I/O buffer for this record stream
/*
    buffer        ADDRESS;                  /* Pointer to the buffer
/*
    CONSTANT start_zero EQUALS .;          /* here to end is cleared for each copy command

/*
/* Data to describe a record stream in the file.
/*
    cur_block    LONGWORD UNSIGNED;        /* pbn of current block
    eof_block    LONGWORD UNSIGNED;        /* pbn of last block in file
    cur_byte     LONGWORD UNSIGNED;        /* offset into the block

/*
/* Flags for an open file. This longword will be zero when file is not open
/*
    flags STRUCTURE LONGWORD UNSIGNED;      /* the whole flags longword
        stream_active    BITFIELD MASK;       /* block currently describes an open file
        output_file      BITFIELD MASK;       /* block currently describes an output file
        flush            BITFIELD MASK;       /* tell advance routine to flush last block
        pointer          BITFIELD MASK;       /* tape position is at this file
    END flags;

/*
/* Pointers to the I/O buffer for this record stream
/*
    buf_base_block LONGWORD UNSIGNED;        /* Pbn of first block in buffer
    buf_high_block LONGWORD UNSIGNED;        /* Pbn of last block in buffer
    high_block_written LONGWORD UNSIGNED;    /* Pbn of highest block to the file

/*
/* The next part is a duplicate of the directory entry
/*
    entry_union UNION;
        "entry" CHARACTER LENGTH 14;          /* The overall entry
```

```

entry fields STRUCTURE:
    fill_00          BYTE FILL;
    type_union UNION;
        type_byte      BYTE UNSIGNED;           /* Type of directory entry
        type            BITFIELD LENGTH 4;       /* Entry is integer in first part
        type_bits STRUCTURE:
            typ_tentative BITFIELD MASK;         /* Tentative file
            typ_empty      BITFIELD MASK;         /* Empty entry
            typ_permanent  BITFIELD MASK;         /* Permanent file
            typ_end_segment BITFIELD MASK;        /* Last (possibly incomplete) entry in segment
            typ_fill_00     BITFIELD LENGTH 3 FILL;
            typ_protected   BITFIELD MASK;         /* File is protected
        END type_bits;
    END type_union;
filename          LONGWORD UNSIGNED;           /* 6 Radix-50 characters
filetype          WORD UNSIGNED;              /* 3 Radix-50 characters
blocks            WORD UNSIGNED;              /* Total length of file
channel           BYTE UNSIGNED;             /* <not meaningful for EXCHANGE>
job               BYTE UNSIGNED;             /* <not meaningful for EXCHANGE>
date STRUCTURE WORD UNSIGNED;           /* Creation date field
    year            BITFIELD LENGTH 5;        /* Years since 1972
    day             BITFIELD LENGTH 5;        /* In decimal, 1-31
    month           BITFIELD LENGTH 5;        /* In decimal, 1-12
END date;
END entry_fields;
END entry_union;
/*
/* Expanded name fields
/*
exp_fullname_len  LONGWORD UNSIGNED;           /* Length of the concatenated name
exp_name_len       LONGWORD UNSIGNED;           /* Length of the name field
exp_type_len       LONGWORD UNSIGNED;           /* Length of the file extension (type)
exp_protected      CHARACTER LENGTH 2;         /* "N" for normal files, "P" for protected
exp_fullname       CHARACTER LENGTH 10;          /* Name in normal form, no embedded blanks
exp_name            CHARACTER LENGTH 6;          /* Blank padded name
exp_type            CHARACTER LENGTH 3;          /* Blank padded extension
exp_date            CHARACTER LENGTH 11;         /* VMS format ascii date
/*
/* The following items define the location of the above entry. They can be used to restore to a position in the
/* RT11 directory, such as to continue wildcard processing of the directory. Note, however, that the RT11
/* directory might be reorganized for many different reasons, such as the creation of a file. The routine
/* EXCH$RT11_CHECK_POSITION will check to make sure that these values still describe the location of the entry.
/* adjusting them to the new location if the entry is not in the same position.
/*
start_block        LONGWORD UNSIGNED;           /* starting pbn of file
seg_number         LONGWORD UNSIGNED;           /* directory segment containing this entry
seg_address        ADDRESS;                   /* address of start of current segment
ent_address        ADDRESS;                   /* address of next directory entry
/*
CONSTANT end_zero EQUALS ::;
CONSTANT "length" EQUALS ::;                  /* length of data structure
END rt11ctx;

```

```
/*
/* $RTNAM
/*
/* Contains the work area for the DELETE and RENAME verbs
/*
AGGREGATE rtnam STRUCTURE PREFIX rtnam$;
/*
/* Dynamic character string descriptors for name components. These strings are reclaimed at the start of each command
/*
  input_filename     QUADWORD UNSIGNED;           /* desc for current item from input parameter list
/*
  size               WORD UNSIGNED;             /* standard size of the block      (symbol EXCHBLKSS_rtnam)
  type               BYTE UNSIGNED;            /* type code field                (symbol EXCHBLK$K_rtnam)
  std_fill           BYTE FILL;                 /* unused
/*
  output_filename    QUADWORD UNSIGNED;          /* desc for output parameter
/*
  input_sticky_name  QUADWORD UNSIGNED;          /* name to make input files sticky
/*
  CONSTANT start_zero EQUALS .;                  /* here to end is cleared for each command
/*
/*
/* Flags for boolean DCL command qualifiers
/*
  qual_flags STRUCTURE LONGWORD UNSIGNED;        /* the whole flags longword
  {
    q_confirm         BITFIELD MASK;
    q_log             BITFIELD MASK;
    q_protect         BITFIELD MASK;
    q_protect_explicit BITFIELD MASK;           /* value for protect bit, not useful unless explicit also set
    q_system          BITFIELD MASK;            /* an explicit /PROTECT or /NOPROTECT was seen
  }
  END qual_flags;
/*
/*
/* Local command flags
/*
  rtnam_flags STRUCTURE LONGWORD UNSIGNED;        /* the whole flags longword
  multiple_files    BITFIELD MASK;              /* the input potentially describes multiple files
  delete_command    BITFIELD MASK;              /* the command is DELETE
  rename_command    BITFIELD MASK;              /* the command is RENAME
  out_fetched       BITFIELD MASK;              /* output filename for rename has been fetched
  END rtnam_flags;
/*
/*
/* Other misc data items
/*
/*
/* Work area for the current input file
/*
  inp_filb          ADDRESS;                   /* pointer to a filb for the input file
  inp_namb          ADDRESS;                   /* pointer to a namb describing the input file
/*
/*
/* Work area for the current output file
/*
  out_filb          ADDRESS;                   /* pointer to a filb for the output file
  out_namb          ADDRESS;                   /* pointer to a namb describing the output file
/*

```

CONSTANT end\_zero EQUALS :: /\* no need to clear the buffers (or create pages not to be used)  
CONSTANT "length" EQUALS :: /\* length of data structure  
END rtnam;

```

/**+
/* $VOLB - Volume Control Block
/* One entry per active foreign volume. A partially filled VOLB is used to describe RMS files.
/*-
AGGREGATE volb STRUCTURE PREFIX volbs;
/*
/* Links to the queues of in-use and available blocks, plus standard type and size fields
/*
  header STRUCTURE QUADWORD UNSIGNED;           /* queue header
    flink  ADDRESS;                            /* forward link
    blink  ADDRESS;                            /* reverse link
  END link;
  size      WORD UNSIGNED;                   /* standard size of the block (symbol EXCHBLK$S_VOLB)
  type      BYTE UNSIGNED;                  /* type code field (symbol EXCHBLK$K_VOLB)
  std_fill  BYTE FILL;                     /* unused
/*
  alloc     ADDRESS;                      /* link in list of allocated $VOLBs
/*
/* Storage for the following control blocks is allocated at the end of the VOLB
/*
  fab      ADDRESS;                      /* pointer to RMS File Access Block
  rab      ADDRESS;                      /* pointer to RMS Record Access Block
  nam      ADDRESS;                      /* pointer to RMS Name Block
  esbuf   ADDRESS;                      /* pointer to RMS Expanded String Buffer
  rsbuf   ADDRESS;                      /* pointer to RMS Result String Buffer
/*
  assoc_namb ADDRESS;                  /* namb from the volume mount
/*
/* The rest of the block will be reset to nulls each time the block is allocated
/*
  CONSTANT start_zero EQUALS .;
/*
/* Device characteristics variables (from $GETDVI or invented for virtual devices)
/*
  devbufsiz  LONGWORD UNSIGNED;          /* device buffer size
  devchar    LONGWORD UNSIGNED;          /* device characteristics ($devdef symbols)
  devclass   LONGWORD UNSIGNED;          /* device class ($dcdef (DC$) symbols)
  devdepend  LONGWORD UNSIGNED;          /* device dependent characteristics
  devnamlen  LONGWORD UNSIGNED;          /* device name length
  devtype    LONGWORD UNSIGNED;          /* device type ($dcdef (DT$) symbols)
  devmaxblock LONGWORD UNSIGNED;         /* maximum number of blocks on device
  volmaxblock LONGWORD UNSIGNED;         /* maximum number of blocks on volume
/*
/* Other bits and pieces
/*
  status STRUCTURE WORD UNSIGNED;        /* status flags for volume
    connected  BITFIELD MASK;            /* RMS Record stream connected to volume
    read_check BITFIELD MASK;            /* /DATA_CHECK=READ
    write_check BITFIELD MASK;           /* /DATA_CHECK=WRITE
    foreign    BITFIELD MASK;            /* Mounted as a foreign volume
    virtual    BITFIELD MASK;            /* Native mode file mounted as a virtual volume

```

```
        write      BITFIELD MASK;          /* Open volume for READ/WRITE access
        vfmt_explicit      BITFIELD MASK;  /* Volume format specified
        verified      BITFIELD MASK;      /* Directory structure has been verified
        END status;
        channel WORD UNSIGNED;          /* channel assigned to device
/*
/* Some commands chain multiple nambs from a volb
/*
    namb_head      ADDRESS;
/*
/* The following structure keeps a bit for each directory segment (bits 1:31) plus a single bit to flag whether
/* directory write caching is active (bit 0). This is an interim attempt to produce decent performance until
/* full caching is implemented. !??
/*
    dircache STRUCTURE LONGWORD UNSIGNED; /* complete directory cache longword
        dircache_active      BITFIELD MASK; /* caching is in force
                                                /* remaining bits map to directory segments
    END dircache;
/*
/* The following format specific longword would normally point to a structure
/* whose composition depended on the FORMAT byte.
/*
    vfmt_specific      ADDRESS;
/*
    vol_format      BYTE UNSIGNED;      /* constant volb$k_vfmt_xxx
    CONSTANT (
        vfmt_invalid,
        vfmt_dos11,
        vfmt_files11,
        vfmt_rtmt,
        vfmt_rt11
    ) EQUALS 0 INCREMENT 1;
/*
    CONSTANT vfmt_lobound EQUALS 0;      /* low bound for case statement
    CONSTANT vfmt_hibound EQUALS volb$k_vfmt_rt11; /* high bound
/*
/* The vol_type is a text string identifying the volume type, i.e. "RT-11" or "DOS-11"
/*
    vol_type_len      LONGWORD UNSIGNED; /* Length of the volume type
    vol_type      CHARACTER LENGTH 8;    /* Could only be "RT-11" or "DOS-11", doesn't need to be long
/*
/* Identification of the $VOLB. This is a unique string which is used to find $VOLBs. For foreign-mounted
/* devices, this is the actual hardware device name, in the format "Dxcn:" (e.g. "DYAO:") as returned by
/* SGETDVI. For virtual devices, this is the original input device string of the form "name:", which is
/* treated as a logical name. When a virtual device is created, the resultant file name string is stored in the
/* vol_ident so that it can be signalled by the standard "volume has been initialized" message, therefore the
/* vol_ident field is much longer than would normally be expected.
/*
    vol_ident_len      LONGWORD UNSIGNED; /* Length of the next field
    vol_ident      CHARACTER LENGTH 128; /* Volume ident buffer
    devnam      CHARACTER LENGTH 64;    /* device name buffer
/*
/* Other items which would be in here if SDL supported external literals. This space is allocated with the
/* rest of the block. Certain pointers above point to these buffers. This area is part of the block, it
/* is only due to restrictions in SDL that these are not actually defined here.
/*
```

/\* buf\_fab CHARACTER LENGTH #fab\$k\_bln; /\* RMS File access block  
/\* buf\_rab CHARACTER LENGTH #rab\$k\_bln; /\* RMS Record access block  
/\* buf\_nam CHARACTER LENGTH #nam\$k\_bln; /\* RMS Name block  
/\* buf\_esbuf CHARACTER LENGTH #nam\$c\_maxrss; /\* RMS expanded name buffer  
/\* buf\_rdbuf CHARACTER LENGTH #nam\$c\_maxrss; /\* RMS result name buffer  
/\* CONSTANT "length" EQUALS ..; /\* length of data structure  
END volb;  
END\_MODULE;

0159 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

