

RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAAAAAAAA		DDDDDDDDDDD	
RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAAAAAAAA		DDDDDDDDDDD	
RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAAAAAAAA		DDDDDDDDDDD	
RRR	FR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAA	AAA	DDD	DDD
RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAA	AAA	DDD	DDD
RRRRRRRRRRR		TTTTTTTTTTTT	PPPPPPPPPPP		AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAAAAAAAAAAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAAAAAAAAAAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAAAAAAAAAAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDDDDDDDDDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDDDDDDDDDD	DDD
RRR	RR	TTT	PPP	PPP	AAA	AAA	DDDDDDDDDDD	DDD

```

RRRRRRR      TTTTTTTTT  DDDDDDD  EEEEEEEEE  FFFFFFFF
RRRRRRR      TTTTTTTTT  DDDDDDD  EEEEEEEEE  FFFFFFFF
RR          RR      TT      DD      DD  EE          FF
RR          RR      TT      DD      DD  EE          FF
RR          RR      TT      DD      DD  EE          FF
RR          RR      TT      DD      DD  EE          FF
RRRRRRR      TT          DD      DD  EEEEEEE  FFFFFFFF
RRRRRRR      TT          DD      DD  EEEEEEE  FFFFFFFF
RR  RR       TT          DD      DD  EE          FF
RR  RR       TT          DD      DD  EE          FF
RR  RR       TT          DD      DD  EE          FF
RR  RR       TT          DD      DD  EE          FF
RR          RR      TT          DDDDDDD  EEEEEEEEE  FF
RR          RR      TT          DDDDDDD  EEEEEEEEE  FF

```

```

SSSSSSS  DDDDDDD  LL
SSSSSSS  DDDDDDD  LL
SS        DD      DD  LL
SS        DD      DD  LL
SS        DD      DD  LL
SS        DD      DD  LL
SSSSSS   DD      DD  LL
SSSSSS   DD      DD  LL
          SS     DD  LL
          SS     DD  LL
          SS     DD  LL
          SS     DD  LL
SSSSSSS  DDDDDDD  LLLLLLLLLL
SSSSSSS  DDDDDDD  LLLLLLLLLL

```

```
{
  Version 'V04-000'
```

```
{*
*****
(* CC YRIGHT (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.
```

```
{*
*****
{
  V03-004 JLV0351      Jake VanNoy      10-APR-1984
           Add UNBIND constants.
{
  V03-003 JLV0334      Jake VanNoy      28-FEB-1984
           Add new constants.
{
  V03-002 JLV0293      Jake VanNoy      28-JUL-1983
           Added FLG$ symbols. Include $TSADEF. Add read
           verify and upline broadcast symbols.
{
  V03-001 MHB0091      Mark Bramhall    3-Mar-1983
           Added constant MAXMSG.
```

```
MODULE $RTPADDEF;
```

```
CONSTANT maxmsg EQUALS 1050;  ( Maximum link data message size
```

```
AGGREGATE AST_BLOCK STRUCTURE PREFIX AST$;
```

```
STATE   LONGWORD;      /* AST ROUTINE (STATE)
IOSB    QUADWORD;      /* IOSB
OPCODE  WORD;          /* OPCODE (VMS RTT mode)
OFFSET  WORD;          /* OFFSET (CTERM)
BUFSIZ  WORD;          /* BUFFER SIZE (CTERM)
ODATA   LONGWORD;      /* OUTPUT DATA BUFFER (CTERM)
ITMLST  LONGWORD;      /* ADDRESS OF ITEM LIST FOR READ
```

```
END;
```

```
AGGREGATE CTERM_FLAGS STRUCTURE PREFIX FLG$;
```

```
CTERM   bitfield mask; /* cterm protocol is running
```

```
CTRL_CY      bitfield mask; /* flushing due to ^C or ^Y
CTRL_O       bitfield mask; /* Control O state
LOGGING      bitfield mask; /* Logging output
VAXHOST      bitfield mask; /* HOST is a VAX
CTRLC        bitfield mask; /* enable standard ^C
```

END;

/* FLAGS DEFINED IN 'RTPAD\$LOG'

AGGREGATE RTLOG_DBGFLAGS STRUCTURE PREFIX RTLOG\$;

```
BANNER      bitfield mask; /* protocol banner
TRACE       bitfield mask; /* enable tracing to RTPAD$TRACE
```

END;

/*
/* Event Flags
/*

```
CONSTANT (
  LINKEFN          /* NET LINK
) EQUALS 1 INCREMENT 1 PREFIX RT$ TAG C;
```

END_MODULE;

```
MODULE $RTEDEF;
```

```
AGGREGATE RTE_BLOCK STRUCTURE PREFIX RTE$;
```

```
CONSTANT buflen EQUALS 80 TAG 'c'; { Maximum message size
fblink ADDRESS; /* forward link
blink ADDRESS; /* backward link
size WORD unsigned; /* size of structure
spare1 WORD; /* spare byte
iosb QUADWORD unsigned; /* IOSB
buf CHARACTER LENGTH 80; /* must match buflen above
CONSTANT length EQUALS . TAG 'c';
END;
```

```
END_MODULE;
```

```
/*
/*
/*
```

MODULE STSADEF;

```

CONSTANT (
    bind,           { bind request
    unbind,         { unbind request
    rebind,         { rebind request
    accept,         { bind accept
    entermode,     { enter mode
    exitmode,      { exit mode
    confirm,       { confirm mode
    nomode,        { no mode
    data,          { data (cterm message)
    mode           { mode message
) EQUALS 1 INCREMENT 1 PREFIX 'PRO$' TAG 'C';

```

AGGREGATE oob STRUCTURE PREFIX oob TAG '':

```

len_exclude LONGWORD TAG '': /* Lengths are not used
exclude     LONGWORD TAG '':
len_include LONGWORD TAG '':
include     LONGWORD TAG '':
len_abort   LONGWORD TAG '':
abort       LONGWORD TAG '':
discard     LONGWORD TAG '': /* discard output mask
echo        LONGWORD TAG '': /* standard echo mask
CONSTANT   len EQUALS . TAG '':

```

END;

```

/*
/* Unbind reason codes
/*

```

```

CONSTANT (
    badvers,       { incompatible version
    noport,        { no portal available
    user,          { user requested unbind (logout)
    disconnect,   { disconnect (setmode hangup)
    unused1,
    unused2,
    proterr       { protocol error
) EQUALS 1 INCREMENT 1 PREFIX 'unbind$' TAG 'c';

```

AGGREGATE cterm STRUCTURE PREFIX ctp\$; /* cterm packet

/* Up to DATSIZE matches RTTDRIVER RBF header

```

flink        LONGWORD; /* forward link
blink        LONGWORD; /* backward link
size         WORD;     /* size of structure
type         BYTE;     /* DYN code (BUFIO)
spare1       BYTE;     /* spare byte
msgdat       LONGWORD; /* message address
usrbfr       LONGWORD; /* user buffer
datsize      WORD;     /* data size
irp          LONGWORD; /* address of associated IRP
jib          LONGWORD; /* address of associated JIB

```

/*

/*

/*

```

    spare2      LONGWORD;      /* spare for RTPAD?
    spare3      LONGWORD;      /* spare for RTPAD?

    #header = .;

/* start of protocol message

    pro_msgtype BYTE;          /* Protocol message type
    pro_fill    BYTE;          /* Protocol fill

/* start of cterm data packet

    msgsize     WORD;          /* length of first message
    #header2 = .;

    msgtype     BYTE;          /* message type

    CONSTANT (
        init,                { Initiate (H <---> S)
        start_rd,            { Start Read (H ---> S)
        read_data,           { Read Data (H <--- S)
        out_band,            { Out-of-Band (H <--- S)
        unread,              { Unread (H ---> S)
        clr_input,           { Clear Input (H ---> S)
        write,               { Write (H ---> S)
        write_com,           { Write Completion (H <--- S)
        dis_state,           { Discard State (H <--- S)
        read_char,           { Read Characteristics (H ---> S)
        char,                { Characteristics (H <---> S)
        check_inp,           { Check Input (H ---> S)
        inp_count,           { Input Count (H <--- S)
        inp_state,           { Input State (H <--- S)
        vmsqio,              { VMS specific QIO (H <---> S)
        vms_brdcst,          { VMS spec broadcast (H <--- S)
        vms_readvfy )        { VMS spec read verify (H ---> S)
    EQUALS T INCREMENT 1 TAG "c_mt";

/*
/* Remainder of block overlaid based on value of msgtype:
/*
    msgfields UNION;

```

```
/*
/* init message structure (H <---> S)
/*
  init STRUCTURE;
    in_flags      BYTE;          /* no flags defined
    in_version    BYTE;          /* protocol version number
    in_eco        BYTE;          /* ECO number for protocol
    in_mod        BYTE;          /* customer modification number
    in_revision   CHARACTER LENGTH 8; /* software revision number
    in_parmtype   BYTE;          /* purpose of the following value
    in_parmval    BYTE;          /* byte count

    CONSTANT len  EQUALS . TAG 'c_in'; /* length of structure
    CONSTANT msglen EQUALS .-#header TAG 'c_in'; /* length of structure minus header
    CONSTANT prolen EQUALS .-#header2 TAG 'c_in'; /* length of structure minus header

END init;
```



```

/*
/* start read and read verify structure (H ---> S)
/*
start_rd STRUCTURE;
sr_flags_overlay union fill; /* Flags for unread
sr_flags character length 3 TAG 'L'; /* 3 bytes of flags
sr_flag_bits structure fill;
sr_underflo BITFIELD LENGTH 2; /* - underflow handling
CONSTANT (
ignore, { -- ignore underflow
bel, { -- ring bell on underflow
terminate ) { -- terminate on underflow
EQUALS 0 INCREMENT 1 TAG 'm_sr';
sr_purge BITFIELD MASK; /* - purge type ahead
sr_format BITFIELD MASK; /* - formatting flag
sr_trmvert BITFIELD MASK; /* - terminate on vertical
sr_continue BITFIELD MASK; /* - continuation read
#shift = ^;
sr_cvtlow BITFIELD LENGTH 2; /* - raise input
CONSTANT (
no_cvt, { -- use upper/lower characteristic
none_only, { -- none this read only
lowtoup ) { -- Normal lower to upper
EQUALS 0 INCREMENT 1 TAG 'c_sr';
CONSTANT no_cvt EQUALS ctp$c_sr_no_cvt@#shift TAG 'm_sr';
CONSTANT none_only EQUALS ctp$c_sr_none_only@#shift TAG 'm_sr';
CONSTANT lowtoup EQUALS ctp$c_sr_lowtoup@#shift TAG 'm_sr';
#shift = ^;
sr_control BITFIELD LENGTH 3; /* - disable control
CONSTANT (
no_ctrl, { -- no control characters disabled
u_and_r, { -- ^U and ^R disabled
edit, { -- all edit control characters
allbutx, { -- all but XON/XOFF
all ) { -- all
EQUALS 0 INCREMENT 1 TAG 'c_sr';
CONSTANT no_ctrl EQUALS ctp$c_sr_no_ctrl@#shift TAG 'm_sr';
CONSTANT u_and_r EQUALS ctp$c_sr_u_and_r@#shift TAG 'm_sr';
CONSTANT edit EQUALS ctp$c_sr_edit@#shift TAG 'm_sr';
CONSTANT allbutx EQUALS ctp$c_sr_allbutx@#shift TAG 'm_sr';
CONSTANT all EQUALS ctp$c_sr_all@#shift TAG 'm_sr';
sr_noecho BITFIELD MASK; /* - no echo read
sr_trmecho BITFIELD MASK; /* - terminator echo
sr_timed BITFIELD MASK; /* - read timeout
#shift = ^;
sr_term_set BITFIELD LENGTH 2; /* - termination set
CONSTANT (
prevterm, { -- use previous read terminators
thisterm, { -- use this read terminators
normterm ) { -- use normal terminators
EQUALS 0 INCREMENT 1 TAG 'c_sr';
CONSTANT prevterm EQUALS ctp$c_sr_prevterm@#shift TAG 'm_sr';
CONSTANT thisterm EQUALS ctp$c_sr_thisterm@#shift TAG 'm_sr';
CONSTANT normterm EQUALS ctp$c_sr_normterm@#shift TAG 'm_sr';
sr_noescape BITFIELD MASK; /* - don't recognize escape
sr_escape BITFIELD MASK; /* - recognize escape

```

```

/* VMS specific bits follow

sr_noedit    BITFIELD MASK;    /* - disable editing
sr_norecall  BITFIELD MASK;    /* - disable recall

end sr_flag_bits;
end sr_flags_overlay;

sr_max_len   WORD;             /* max length of read
sr_end_data  WORD;             /* end of data in read buffer
sr_timeout   WORD;             /* timeout value
sr_end_prmt  WORD;             /* end of prompt
sr_str_disp  WORD;             /* start of display
sr_lo_water  WORD;             /* low water mark

TWO_READS structure;
  TWO_READS_OVERLAY union fill;

    sr_term      CHARACTER LENGTH 1;    /* termination set (byte counted field)
    /* read data starting position (after term set)

    CONSTANT len    EQUALS . TAG 'c_sr';    /* length of structure
    CONSTANT msglen EQUALS .-#header TAG 'c_sr'; /* length of structure minus header
    CONSTANT prolen EQUALS .-#header2 TAG 'c_sr'; /* length of structure minus header

  READ_VERIFY structure ;

    sr2_altechsize WORD UNSIGNED;    /* alt echo size
    sr2_picstrsize WORD UNSIGNED;    /* picture string size
    sr2_editflags  WORD UNSIGNED;    /* flags
    sr2_fillchar   WORD UNSIGNED;    /* fill characters
    sr2_term       CHARACTER LENGTH 1; /* terminator set

    CONSTANT len    EQUALS . TAG 'c_sr2';    /* length of structure
    CONSTANT msglen EQUALS .-#header TAG 'c_sr2'; /* length of structure minus header
    CONSTANT prolen EQUALS .-#header2 TAG 'c_sr2'; /* length of structure minus header
  end READ_VERIFY;
end TWO_READS_OVERLAY;
end TWO_READS;

END start_rd;

```

```

/*
/* read data structure (H <--- S)
/*
read_data STRUCTURE;
  rd_flags_overlay union fill; /* Flags for unread
  rd_flags byte unsigned;
  rd_flag_bits structure fill;
  rd_com_code BITFIELD LENGTH 4; /* - completion code
  CONSTANT (
    normal, ( -- normal terminator
    valesc, ( -- valid escape
    invesc, ( -- invalid escape
    outband, ( -- out of band
    inpfll, ( -- input buffer full
    timeout, ( -- read timed out
    unread, ( -- unread request received
    underflo, ( -- underflow
    abstoken, ( -- absentee token
    vert_cng, ( -- vertical position change
    lineBrk, ( -- line break
    framerr, ( -- frame error
    parity, ( -- parity error
    overrun ) ( -- receiver over-run
  )
  rd_mor_data BITFIELD MASK; /* - more data in typeahead
end rd_flag_bits;
end rd_flags_overlay;
rd_lo_water WORD; /* low water
rd_vert_cng BYTE; /* vertical change since read started
rd_curs_pos BYTE; /* cursor position from EOL
rd_term_pos WORD; /* position of terminator
rd_data CHARACTER LENGTH 0; /* start of read data

CONSTANT len EQUALS . TAG 'c_rd'; /* length of structure
CONSTANT msglen EQUALS .-#header TAG 'c_rd'; /* length of structure minus header
CONSTANT prolen EQUALS .-#header2 TAG 'c_rd'; /* length of structure minus header

END read_data;

```

```
/*
/* out of band structure (M <--- S)
/*
out_band STRUCTURE;
  ob_flags_overlay union fill; /* Flags for unread
    ob_flags byte unsigned;
    ob_flag_bits structure fill;
    ob_linebrk BITFIELD MASK; /* - line break occurred
  end ob_flag_bits;
end ob_flags_overlay;

ob_char BYTE; /* one byte of data

CONSTANT len EQUALS . TAG 'c_ob'; /* length of structure
CONSTANT msglen EQUALS .-#header TAG 'c_ob'; /* length of structure minus header
CONSTANT prolen EQUALS .-#header2 TAG 'c_ob'; /* length of structure minus header

END out_band;
```

```
/*
/* unread structure (H ---> S)
/*
unread STRUCTURE;
  ur_flags_overlay union fill; /* Flags for unread
  ur_flags byte unsigned;
  ur_flag_bits structure fill;
  ur_cond BITFIELD MASK; /* - unread conditional
  end ur_flag_bits;
end ur_flags_overlay;

CONSTANT len EQUALS . TAG 'c_ur'; /* length of structure
CONSTANT msglen EQUALS .-#header TAG 'c_ur'; /* length of structure minus header
CONSTANT prolen EQUALS .-#header2 TAG 'c_ur'; /* length of structure minus header

END unread;
```

```
/*
/* clear input structure (M ---> S)
/*
  clr_input STRUCTURE;
    ci_flags      BYTE;          /* no flags defined
    CONSTANT len  EQUALS . TAG 'c_ci'; /* length of structure
    CONSTANT msglen EQUALS .-#header TAG 'c_ci'; /* length of structure minus header
    CONSTANT prolen EQUALS .-#header2 TAG 'c_ci'; /* length of structure minus header
END clr_input;
```

```

/*
/* write structure (H ---> S)
/*
write STRUCTURE;
  wr_flags_overlay union fill; /* Flags for write
  wr_flags word unsigned;
  wr_flag_bits structure fill;
  wr_lock BITFIELD LENGTH 2; /* - locking
    CONSTANT (
      noaction, { -- no locking action
      before, { -- lock before, leave locked
      befaff, { -- lock before, unlock after
      befaffre ) { -- lock before, unlock after, redisplay
    EQUALS 0 INCREMENT 1 TAG 'm_wr';
  wr_newline BITFIELD MASK; /* - VMS specific, newline modifier
  wr_discard BITFIELD MASK; /* - cancel ^O
  wr_begin BITFIELD MASK; /* - beginning of write
  wr_end BITFIELD MASK; /* - end of write
  #shift = ^;
  wr_prefix BITFIELD LENGTH 2; /* - prefix code
    CONSTANT (
      no_fix, { -- no prefix
      newlinecnt, { -- new line count
      char ) { -- character prefix
    EQUALS 0 INCREMENT 1 TAG 'c_wr';
    CONSTANT no_fix EQUALS ctp$c_wr_no_fix@#shift TAG 'm_wr';
    CONSTANT newlinecnt EQUALS ctp$c_wr_newlinecnt@#shift TAG 'm_wr';
    CONSTANT char EQUALS ctp$c_wr_char@#shift TAG 'm_wr';
  wr_postfix BITFIELD LENGTH 2; /* - postfix code
  wr_status BITFIELD MASK; /* - return status
  wr_transparent BITFIELD MASK; /* - write passall
END wr_flag_bits;
END wr_flags_overlay;
wr_prefix BYTE; /* prefix value
wr_postfix BYTE; /* postfix value
wr_data CHARACTER LENGTH 0; /* start of data

CONSTANT len EQUALS . TAG 'c_wr'; /* length of structure
CONSTANT msglen EQUALS .-#header TAG 'c_wr'; /* length of structure minus header
CONSTANT prolen EQUALS .-#header2 TAG 'c_wr'; /* length of structure minus header

END write;

```

```
/*
/* write completion structure (H <--- S)
/*
write_com STRUCTURE;
  wc_flags_overlay union fill; /* Flags for unread
    wc_flags byte unsigned;
    wc_flag_bits structure fill;
    wc_discard BITFIELD MASK; /* - discard state
  end wc_flag_bits;
end wc_flags_overlay;

wc_horpos WORD; /* horizontal position
wc_verpos WORD; /* vertical position

CONSTANT len EQUALS .TAG "c_wc"; /* Length of structure
CONSTANT msglen EQUALS .-#header TAG "c_wc"; /* length of structure minus header
CONSTANT prolen EQUALS .-#header2 TAG "c_wc"; /* length of structure minus header

END write_com;
```



```
/*
/* read characteristics structure (H ---> S)
/*
  read_char STRUCTURE;
    rc_flags      BYTE;          /* no flags defined
    rc_selector   WORD;         /* selector start position

    CONSTANT len  EQUALS . TAG 'c_rc'; /* length of structure
    CONSTANT msglen EQUALS .-#header TAG 'c_rc'; /* length of structure minus header
    CONSTANT prolen EQUALS .-#header2 TAG 'c_rc'; /* length of structure minus header

END read_char;
```

```
/*
/* characteristics structure (H <---> S)
/*
char STRUCTURE;
  ch_flags      BYTE;          /* no flags defined
  ch_param      WORD;         /* start of characteristics
  ch_value      CHARACTER LENGTH 0; /* value

CONSTANT len     EQUALS . TAG 'c_ch'; /* length of structure
CONSTANT msglen  EQUALS .-#header TAG 'c_ch'; /* length of structure minus header
CONSTANT prolen  EQUALS .-#header2 TAG 'c_ch'; /* length of structure minus header

/* characteristics selector types

CONSTANT (
  physical,
  logical,
  cterm
) EQUALS 0 INCREMENT 1 PREFIX 'CHS' TAG C;

/* characteristics selectors, type = physical

CONSTANT (
  in_speed,
  out_speed,
  char_size,
  parity_enable,
  parity_type,
  modem_present,
  autobaud,
  manage_guar,
  swchar1,
  swchar2,
  eightbit,
  manage_ena
) EQUALS 1 INCREMENT 1 PREFIX 'CHS' TAG 'C_PH';

/* characteristics selectors, type = logical

CONSTANT (
  mode_writing,
  term_bits,
  term_type,
  output_flow,
  page_stop,
  flow_char_pass,
  input_flow,
  loss_notif,
  line_width,
  page_length,
  stop_length,
  cr_fill,
  lf_fill,
  wrap,
  hor_tab,
  vert_tab,
```

```

form feed
) EQUALS 1 INCREMENT 1 PREFIX 'CH$' TAG 'C_LG';

```

```
/* characteristics selectors, type = cterm
```

```

CONSTANT (
  ignore_input,
  char_att,           /* see tty_attributes, etc. below
  ctrl0_pass,
  raise_input,
  normal_echo,
  input_esc,
  output_esc,
  input_count,
  auto_prompt,
  error_processing
) EQUALS 1 INCREMENT 1 PREFIX 'CH$' TAG 'C_CT';

```

```

CONSTANT (
  even,
  odd,
  clear,             /* no support for set or clear on VMS
  set
) EQUALS 1 INCREMENT 1 PREFIX 'CH$' TAG C_PARITY;

```

```

tty_attributes STRUCTURE;
  ch_known    BITFIELD MASK;
  ch_scope    BITFIELD MASK;
end tty_attributes;

```

```

oob_handling STRUCTURE;
  ch_oo       bitfield mask length 2; /* out of band handling
  ch_i        bitfield mask;         /* include character
  ch_d        bitfield mask;         /* discard output
  ch_ee       bitfield mask length 2; /* echo control
  ch_f        bitfield mask;         /* special enable
end oob_handling;

```

```

oob_data STRUCTURE;
  ch_char     BYTE;                 /* data character
  ch_mask     BYTE;                 /* mask for attributes
  ch_attr     BYTE;                 /* attributes
end oob_data;

```

```

CONSTANT (
  cancel,        { - out of band flags
  iclear,        { -- cancel previous      0
  dclear,        { -- immediate clear      1
  hello }        { -- deferred clear      2
                { -- hello                3
EQUALS 0 INCREMENT 1 TAG 'c_ch';

```

```

CONSTANT (
  echonone,      { echo control
  echoself,      { -- don't echo
  echostandard, { -- echo character as self
                { -- standard echo

```

```
echoboth )  
EQUALS 0 INCREMENT 1 TAG "c_ch"; ( -- echo both  
END char;
```

D1

RE

```
/*
/* check input structure (H ---> S)
/*
check_inp STRUCTURE;
  cR_flags      BYTE;          /* no flags defined
  CONSTANT len  EQUALS . TAG 'c_ck'; /* length of structure
  CONSTANT msglen EQUALS .-#header TAG 'c_ck'; /* length of structure minus header
  CONSTANT prolen EQUALS .-#header2 TAG 'c_ck'; /* length of structure minus heade
END check_inp;
```

```
/*
/* input count structure (H <--- S)
/*
inp_count STRUCTURE;
  ic_flags    BYTE;          /* no flags defined
  ic_count    WORD;         /* input count

  CONSTANT len    EQUALS . TAG 'c_ic'; /* length of structure
  CONSTANT msglen EQUALS .-#header TAG 'c_ic'; /* length of structure minus header
  CONSTANT prolen EQUALS .-#header2 TAG 'c_ic'; /* length of structure minus header

END inp_count;
```

```

/*
/* input state structure (H <--- S)
/*
inp_state STRUCTURE;
  is_flags_overlay union fill; /* Flags for unread
    is_flags byte unsigned;
    is_flag_bits structure fill;
    is_nonzero BITFIELD MASK; /* - count change to non-zero
  end is_flag_bits;
end is_flags_overlay;

CONSTANT len EQUALS . TAG 'c_is'; /* length of structure
CONSTANT msglen EQUALS .-#header TAG 'c_is'; /* length of structure minus header
CONSTANT prolen EQUALS .-#header2 TAG 'c_is'; /* length of structure minus header

END inp_state;

/*
/* VMS QIO REQUEST (H ---> S)
/*
vmsqio STRUCTURE;
  vms_flags_overlay union fill; /* Flags for unread
    vms_flags byte unsigned;
    vms_flag_bits structure fill;
    vms_useiosb BITFIELD MASK; /* use iosb to determine length
    vms_readlen BITFIELD MASK; /* - read-type iosb buffer length
  end vms_flag_bits;
end vms_flags_overlay;

vms_reqid LONGWORD; /* qio request id

vmsfields UNION;
  VMSREQ STRUCTURE; /* request
    vms_func WORD; /* qio function code
    vms_plen WORD; /* these four are repeated...
    vms_pcode WORD; /* for each parameter
    vms_pflags STRUCTURE TAG W; /*
      vms_ref BITFIELD MASK; /* - pass by reference
      vms_item BITFIELD MASK; /* - item list or Pn
      vms_buffer BITFIELD MASK; /* - this is return buffer
      vms_fill2 BITFIELD LENGTH 16-^; /* - fill to 1 word
    END vms_pflags;
    vms_pdata CHARACTER LENGTH 0; /* value

    CONSTANT len EQUALS . TAG 'c_vms'; /* length of structure
    CONSTANT msglen EQUALS .-#header TAG 'c_vms'; /* length of structure minus header
    CONSTANT prolen EQUALS .-#header2 TAG 'c_vms'; /* length of structure minus header
  END VMSREQ;

  VMSRESP STRUCTURE;
    vms_iosb QUADWORD; /* iosb
    vms_rdata CHARACTER LENGTH 0; /* start of data
  END VMSRESP;

END vmsfields;

```


RTDEF.SDL:1

16-SEP-1984 16:44:45.^{K 10}₁₀ Page 23

END vmsqio;

R



```
/*
/* upline broadcast (H <--- S)
/*
  broadcast STRUCTURE;
    br_flags      WORD UNSIGNED;      /* no flags defined
    br_msgcode    WORD UNSIGNED;      /* mailbox message code
    br_unitnum    WORD UNSIGNED;      /* unit number
    br_devname    CHARACTER LENGTH 16; /* device name
    br_msglen     WORD UNSIGNED;      /* length of text
    br_msgtxt     CHARACTER LENGTH 0;  /* start of text

    CONSTANT len  EQUALS . TAG 'c_br'; /* length of structure
    CONSTANT msglen EQUALS .-#header TAG 'c_br'; /* length of structure minus header
    CONSTANT prolen EQUALS .-#header2 TAG 'c_br'; /* length of structure minus header

  END broadcast;
END msgfields; /* end of protocol messages
END cterm;
```

AGGREGATE VMSQIO STRUCTURE PREFIX vms\$;

plen	WORD;	/* these four are repeated...
pflags	WORD;	/* ..
pcode	WORD;	/* .. for each parameter
pdata	CHARACTER LENGTH 0;	/* value

END; /* VMSQIO

END_MODULE;

This image displays a grid of 100 small terminal window screenshots, arranged in 10 rows and 10 columns. Each window shows a different software utility or command-line interface, with various text, data, and graphical elements. The utilities shown include:

- RPCMSGTX LIS
- RPCMOVE3 LIS
- RPCSORT LIS
- RPCOPEN LIS
- RPCGATOR LIS
- RPCMSGPTR LIS
- RPCVECTOR LIS
- RPCPRINT LIS
- RPCGUATE LIS
- DTE_DF03 MAP
- RTORIVER MAP
- RTPAD
- RTPAD MAP
- RTPADMACS MAR
- RTDEF SOL
- DTE_DF03 MAR
- CTORIVER LIS

The screenshots are arranged in a grid, with each window showing a different utility or command-line interface. The text is small and difficult to read, but the overall layout is consistent across the grid.