Appendix A: The LAPB Specification in LOTOS

```
specification spec[dl,phl] (n2,k,resolvecollis : Nat,senddmoption : Bool, tmode : mmode) : noexit
library
    NaturalNumber, Boolean, BitString, Bit
endlib

type Time is
    sorts time
    opns
      start   : -> time
      expired : -> time
      reset   : -> time
endtype

type Termine is
    sorts term
    opns
      termine      : -> term
      causetermine : -> term
endtype

type Linkreset is
    sorts linkreset
    opns
      causelreset : -> linkreset
      lreset      : -> linkreset
endtype

type Packet is BitString
    renamedby
    sortnames packet for BitString
endtype

type Mode is NaturalNumber,Boolean
    sorts mmode
    opns
      basic    :  ->  mmode
      extended :  ->  mmode
      k        : mmode  ->  Nat
      eq       : mmode, mmode -> Bool
    eqns
    ofsort Nat
      k(basic)    = Succ(Succ(0)) ** (Succ(Succ(Succ(0)))) ;
      k(extended) = Succ(Succ(0)) ** (Succ(Succ(Succ(Succ(Succ(Succ(Succ(0)))))))) ;
    ofsort Bool
      eq(basic,extended)  = false;
      eq(basic,basic)     = true;
      eq(extended,basic)  = false;
      eq(extended,extended) = true;
endtype

type Role is Adress, Boolean
    sorts role
    opns
      dte :  -> role
      dce :  -> role
```

```
      adr : role -> adress
      eq  : role,role -> Bool
    eqns
    ofsort adress
      adr(dte) = a;
      adr(dce) = b;
    ofsort Bool
      eq(dte,dte) = true;
      eq(dte,dce) = false;
      eq(dce,dce) = true;
      eq(dce,dte) = false;
endtype

type Adress is Boolean
    sorts adress
    opns
      a              :  ->  adress
      b              :  ->  adress
      is_a           : adress  ->  Bool
      is_b           : adress  ->  Bool
      corrupt_adr    : adress  ->  adress
      is_corrupt_adr  : adress  ->  Bool
      eq             : adress,adress -> Bool
    eqns
    forall x : adress
    ofsort Bool
      is_a(a) = true ;
      is_a(b) = false ;
      is_b(a) = false ;
      is_b(b) = true ;
      is_corrupt_adr(a) = false ;
      is_corrupt_adr(b) = false ;
      is_corrupt_adr(corrupt_adr(x)) = true ;
      eq(a,a)  = true;
      eq(a,b)  = false;
      eq(b,a) = false;
      eq(b,b)  = true;
endtype

type Flag is Boolean
    sorts flag
    opns
      flagging  :       ->  flag
      noflag    : flag  ->  flag
      is_flag   : flag  ->  Bool
    eqns
    forall f: flag
    ofsort Bool
      is_flag(flagging)  =  true;
      is_flag(noflag(f)) =  false;
endtype

type Infor is Boolean
   sorts inforr
   opns
     infor     : -> inforr
     noinfor   : -> inforr
     is_infor : inforr -> Bool
   eqns
```

```
   ofsort Bool
    is_infor(infor) = true;
    is_infor(noinfor) = false;
endtype

type Corrlength is Boolean
     (* type greater for max established for i frame and super unember frame *)
    sorts corrlength
    opns
      correct :  ->  corrlength
      incorrect : -> corrlength
      is_correct : corrlength -> Bool
    eqns
    ofsort Bool
      is_correct(correct) = true;
      is_correct(incorrect) = false;
endtype

type Lesslength is Boolean
    sorts lesslength
    opns
      less :  -> lesslength
      notless :  -> lesslength
      is_not_less  : lesslength  -> Bool
    eqns
    ofsort Bool
      is_not_less(less) = false;
      is_not_less(notless) = true;
endtype

type Abortf is Boolean
    sorts abortf
    opns
      abort    :          ->  abortf
      noabort  :          ->  abortf
      is_abort :  abortf  ->  Bool
    eqns
    ofsort Bool
      is_abort(abort)   = true;
      is_abort(noabort) = false;
endtype

type Control is NaturalNumber, Bit
    sorts  ctl
    opns
      ctl_iframe : Nat, Bit, Nat  ->  ctl
      ctl_sframe : Bit, Nat       ->  ctl
      ctl_uframe : Bit            ->  ctl
      undef         : ctl         ->  ctl
      is_undef_ctl  : ctl         ->  Bool
      nss           : ctl         ->  Nat
      nrr           : ctl         ->  Nat
      pff           : ctl         ->  Bit
    eqns
    forall  s, r : Nat , x : Bit , c : ctl
    ofsort Bool
      is_undef_ctl(undef(c)) = true;
      is_undef_ctl(ctl_iframe(s,x,r))  = false;
      is_undef_ctl(ctl_sframe(x,r))    = false;
```

```
      is_undef_ctl(ctl_uframe(x))      = false;
    ofsort Nat
      nss(ctl_iframe(s,x,r)) = s;
      nrr(ctl_iframe(s,x,r)) = r;
      nrr(ctl_sframe(x,r))   = r;
    ofsort Bit
      pff(ctl_iframe(s,x,r)) = x;
      pff(ctl_sframe(x,r))   = x;
      pff(ctl_uframe(x))     = x;
endtype

type Fcs is Boolean
    sorts fcs
    opns
      fc             :      -> fcs
      corrupt_fcs    : fcs  ->  fcs
      is_corrupt_fcs : fcs  ->  Bool
    eqns
    forall x: fcs
    ofsort Bool
      is_corrupt_fcs(fc)   = false;
      is_corrupt_fcs(corrupt_fcs(x)) = true;
endtype

type Frmrreason is Control,NaturalNumber,Bit
    sorts frmrreason
    opns
      nors        : -> frmrreason
      make_reason : ctl,Bit,Nat,Bit,Nat,Bit,Bit,Bit,Bit,Bit -> frmrreason
endtype

type Frametype is NaturalNumber, Packet, Adress, Flag, Control, Fcs,Frmrreason, Boolean, Infor, Corrlength, Lesslength
    sorts  frame
    opns
      make_iframe : flag, adress, ctl, packet, fcs, flag  -> frame
      make_rr     : flag, adress, ctl,fcs, flag  -> frame
      make_rnr    : flag, adress, ctl, fcs, flag -> frame
      make_rej    : flag, adress, ctl, fcs, flag -> frame
      make_sabm   : flag, adress, ctl, fcs, flag -> frame
      make_sabme  : flag, adress, ctl, fcs, flag -> frame
      make_disc   : flag, adress, ctl, fcs, flag -> frame
      make_ua     : flag, adress, ctl, fcs, flag -> frame
      make_dm     : flag, adress, ctl, fcs, flag -> frame
      make_frmr   : flag, adress, ctl, frmrreason, fcs, flag  -> frame
      get_ctl     : frame  ->  ctl
      get_flag1   : frame  ->  flag
      get_flag2   : frame  ->  flag
      get_adr     : frame  ->  adress
      get_packet  : frame  ->  packet
      get_info    : frame  ->  frmrreason
      get_fcs     : frame  ->  fcs
      is_i        : frame  ->  Bool
      is_rr       : frame  ->  Bool
      is_rnr      : frame  ->  Bool
      is_rej      : frame  ->  Bool
      is_sabm     : frame  ->  Bool
      is_sabme    : frame  ->  Bool
      is_disc     : frame  ->  Bool
      is_ua       : frame  ->  Bool
```

```
  is_dm       : frame  ->  Bool
  is_frmr     : frame  ->  Bool
  h           : frame  ->  Nat
  is_s        : frame  ->  Bool
  is_u        : frame  ->  Bool
eqns
forall f : frame, f1, f2 : flag, a : adress, c : ctl, inf : frmrreason,
      fcc : fcs, p : packet

ofsort Nat
  h(make_iframe(f1,a,c,p,fcc,f2)) = 0;
  h(make_rr(f1,a,c,fcc,f2))       = Succ(0) ;
  h(make_rnr(f1,a,c,fcc,f2))      = Succ(Succ(0));
  h(make_rej(f1,a,c,fcc,f2))      = Succ(Succ(Succ(0)));
  h(make_sabm(f1,a,c,fcc,f2))     = Succ(Succ(Succ(Succ(0))));
  h(make_sabme(f1,a,c,fcc,f2))    = Succ(h(make_sabm(f1,a,c,fcc,f2))) ;
  h(make_disc(f1,a,c,fcc,f2))     = Succ(h(make_sabme(f1,a,c,fcc,f2)));
  h(make_ua(f1,a,c,fcc,f2))       = Succ(h(make_disc(f1,a,c,fcc,f2)));
  h(make_dm(f1,a,c,fcc,f2))       = Succ(h(make_ua(f1,a,c,fcc,f2)));
  h(make_frmr(f1,a,c,inf,fcc,f2))     = Succ(h(make_dm(f1,a,c,fcc,f2)));
ofsort Bool
  is_i(f) = (h(f) eq 0);
  is_rr(f) = (h(f) eq Succ(0));
  is_rnr(f) = (h(f) eq Succ(Succ(0)));
  is_rej(f) = (h(f) eq Succ(Succ(Succ(0))));
  is_sabm(f) = (h(f) eq Succ(Succ(Succ(Succ(0)))));
  is_sabme(f) = (h(f) eq Succ(Succ(Succ(Succ(Succ(0))))));
  is_disc(f) = (h(f) eq Succ(Succ(Succ(Succ(Succ(Succ(0)))))));
  is_ua(f) = (h(f) eq Succ(Succ(Succ(Succ(Succ(Succ(Succ(0))))))));
  is_dm(f)= (h(f) eq (Succ(Succ(0))) ** Succ(Succ(Succ(0))));
  is_frmr(f) = (h(f) eq (Succ(Succ(Succ(0)))) * Succ(Succ(Succ(0))));
  is_s(f) = (is_rr(f) or is_rnr(f)) or (is_rej(f)) ;
  is_u(f) = (((is_sabm(f)) or (is_sabme(f))) or (is_disc(f))) or  (((is_ua(f)) or (is_dm(f))) or (is_frmr(f)))
ofsort ctl
  get_ctl(make_iframe(f1,a,c,p,fcc,f2)) = c;
  get_ctl(make_rr(f1,a,c,fcc,f2)) = c;
  get_ctl(make_rnr(f1,a,c,fcc,f2)) = c;
  get_ctl(make_rej(f1,a,c,fcc,f2)) = c;
  get_ctl(make_sabm(f1,a,c,fcc,f2)) = c;
  get_ctl(make_sabme(f1,a,c,fcc,f2)) = c;
  get_ctl(make_disc(f1,a,c,fcc,f2)) = c;
  get_ctl(make_ua(f1,a,c,fcc,f2)) = c;
  get_ctl(make_dm(f1,a,c,fcc,f2)) = c;
  get_ctl(make_frmr(f1,a,c,inf,fcc,f2)) = c;
ofsort flag
  get_flag1(make_iframe(f1,a,c,p,fcc,f2)) = f1;
  get_flag1(make_rr(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_rnr(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_rej(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_sabm(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_sabme(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_disc(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_ua(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_dm(f1,a,c,fcc,f2)) = f1;
  get_flag1(make_frmr(f1,a,c,inf,fcc,f2)) = f1;
  get_flag2(make_iframe(f1,a,c,p,fcc,f2)) = f2;
  get_flag2(make_rr(f1,a,c,fcc,f2)) = f2;
  get_flag2(make_rnr(f1,a,c,fcc,f2)) = f2;
  get_flag2(make_rej(f1,a,c,fcc,f2)) = f2;
```

```
    get_flag2(make_sabm(f1,a,c,fcc,f2)) = f2;
    get_flag2(make_sabme(f1,a,c,fcc,f2)) = f2;
    get_flag2(make_disc(f1,a,c,fcc,f2)) = f2;
    get_flag2(make_ua(f1,a,c,fcc,f2)) = f2;
    get_flag2(make_dm(f1,a,c,fcc,f2)) = f2;
    get_flag2(make_frmr(f1,a,c,inf,fcc,f2)) = f2;
  ofsort adress
    get_adr(make_iframe(f1,a,c,p,fcc,f2)) = a;
    get_adr(make_rr(f1,a,c,fcc,f2)) = a;
    get_adr(make_rnr(f1,a,c,fcc,f2)) = a;
    get_adr(make_rej(f1,a,c,fcc,f2)) = a;
    get_adr(make_sabm(f1,a,c,fcc,f2)) = a;
    get_adr(make_sabme(f1,a,c,fcc,f2)) = a;
    get_adr(make_disc(f1,a,c,fcc,f2)) = a;
    get_adr(make_ua(f1,a,c,fcc,f2)) = a;
    get_adr(make_dm(f1,a,c,fcc,f2)) = a;
    get_adr(make_frmr(f1,a,c,inf,fcc,f2)) = a;
  ofsort ctl
    get_ctl(make_iframe(f1,a,c,p,fcc,f2)) = c;
    get_ctl(make_rr(f1,a,c,fcc,f2)) = c;
    get_ctl(make_rnr(f1,a,c,fcc,f2)) = c;
    get_ctl(make_rej(f1,a,c,fcc,f2)) = c;
    get_ctl(make_sabm(f1,a,c,fcc,f2)) = c;
    get_ctl(make_sabme(f1,a,c,fcc,f2)) = c;
    get_ctl(make_disc(f1,a,c,fcc,f2)) = c;
    get_ctl(make_ua(f1,a,c,fcc,f2)) = c;
    get_ctl(make_dm(f1,a,c,fcc,f2)) = c;
    get_ctl(make_frmr(f1,a,c,inf,fcc,f2)) = c;
  ofsort packet
    get_packet(make_iframe(f1,a,c,p,fcc,f2)) = p;
  ofsort fcs
    get_fcs(make_iframe(f1,a,c,p,fcc,f2)) = fcc;
    get_fcs(make_rr(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_rnr(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_rej(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_sabm(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_sabme(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_disc(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_ua(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_dm(f1,a,c,fcc,f2)) = fcc;
    get_fcs(make_frmr(f1,a,c,inf,fcc,f2)) = fcc;
  ofsort frmrreason
    get_info(make_frmr(f1,a,c,inf,fcc,f2)) = inf;
endtype

type Extframetype is NaturalNumber,Frametype,Infor, Corrlength,
    Lesslength, Abortf, Boolean,Mode
  sorts  eframe
  opns
     _mod_ : Nat,Nat         -> Nat
    _-_   : Nat,Nat         -> Nat
    m : frame,inforr,corrlength,lesslength,abortf -> eframe
    collis    : eframe,eframe -> Bool
    resp      : eframe -> Bool
    is_valid  : eframe -> Bool
    is_reject : eframe -> Bool
    not_abort : eframe -> Bool
    correct   : eframe -> Bool
    incorrect : eframe -> Bool
```

```
  flg1      : eframe -> flag
  flg2      : eframe -> flag
  pack      : eframe -> packet
  adrs      : eframe -> adress
  ctrl      : eframe -> ctl
  fcss      : eframe -> fcs
  reasfrmr  : eframe -> frmrreason
  ns        : eframe -> Nat
  nr        : eframe -> Nat
  pf        : eframe -> Bit
  get_infor : eframe -> inforr
  get_c     : eframe -> corrlength
  get_l     : eframe -> lesslength
  get_f     : eframe -> frame
  get_a     : eframe -> abortf
  is_inforr : eframe -> Bool
  corrl     : eframe -> Bool
  _eq_      : eframe,eframe -> Bool
  notvalid_nrf      : eframe,Nat,Nat,Nat -> Bool
eqns
  forall f: eframe ,f1 : eframe, fr : frame, a : abortf, ir : inforr , c : corrlength, l : lesslength, vs ,lu,x,y, k : Nat
  ofsort flag
    flg1(m(fr,ir,c,l,a)) = get_flag1(fr);
    flg2(m(fr,ir,c,l,a)) = get_flag2(fr);
  ofsort packet
    pack(m(fr,ir,c,l,a)) = get_packet(fr);
  ofsort adress
    adrs(m(fr,ir,c,l,a)) = get_adr(fr);
  ofsort ctl
    ctrl(m(fr,ir,c,l,a)) = get_ctl(fr);
  ofsort fcs
    fcss(m(fr,ir,c,l,a)) = get_fcs(fr);
  ofsort frmrreason
    reasfrmr(m(fr,ir,c,l,a)) = get_info(fr);
  ofsort Nat
    ns(m(fr,ir,c,l,a)) = nss(get_ctl(fr));
    nr(m(fr,ir,c,l,a)) = nrr(get_ctl(fr));
  ofsort Bit
    pf(m(fr,ir,c,l,a)) = pff(get_ctl(fr));
  ofsort abortf
    get_a(m(fr,ir,c,l,a)) = a;
  ofsort inforr
    get_infor(m(fr,ir,c,l,a)) = ir;
  ofsort corrlength
    get_c(m(fr,ir,c,l,a)) = c;
  ofsort lesslength
    get_l(m(fr,ir,c,l,a)) = l;
  ofsort frame
    get_f(m(fr,ir,c,l,a)) = fr;
  ofsort Bool
    (h(get_f(f)) eq h(get_f(f1))) => collis(f,f1)= true;
    (h(get_f(f)) ne h(get_f(f1))) => collis(f,f1)= false;
    (is_sabm(get_f(f)) or is_sabme(get_f(f))) => resp(f) = false;
    is_disc(get_f(f)) => resp(f) = true;
    corrl(f) = is_correct(get_c(f));
    is_inforr(f) = is_infor(get_infor(f));
    is_i(get_f(f))  => is_reject(f) = (is_undef_ctl(ctrl(f)) )  or (not(is_correct(get_c(f))));
    not(is_i(get_f(f))) => is_reject(f) = ((is_undef_ctl(ctrl(f))) or
                                              (not(is_correct(get_c(f)))) ) or is_infor(get_infor(f));
```

```
    is_valid(f) = not(((is_corrupt_adr(adrs(f))) or(is_corrupt_fcs(fcss(f)))) or
        (((not(is_flag(get_flag1(get_f(f))))) or (not(is_flag(get_flag2(get_f(f)))))) or (not(is_not_less(get_l(f))))));
    not_abort(f) = not(is_abort(get_a(f)));
    correct(f) = (is_valid(f) and (not(is_reject(f)))) and not_abort(f);
    incorrect(f) = (not(is_valid(f))) or (is_reject(f) or (not(not_abort(f))));
    f eq f1 = h(get_f(f)) eq h(get_f(f1)) ;
    notvalid_nrf(f,vs,lu,k) = not((((nr(f) ge lu) or (nr(f) le (vs mod k))) and ((vs mod k) lt lu)) or (((nr(f) ge lu)
                        and (nr(f) le (vs mod k))) and ((vs mod k) ge lu)));
   ofsort Nat
    x - x          = 0;
    Succ(x) - y    = Succ(x - y);
    x mod 0        = x;
    x lt y => x mod y = x;
    x mod y        = (x - y) mod y;

endtype

type primitive is Packet,Boolean,NaturalNumber
  sorts primitive
  opns
   dl_est_req :      -> primitive
   dl_dis_req :      -> primitive
   dl_dis_cnf :      -> primitive
   dl_est_ind :      -> primitive
   dl_est_cnf :      -> primitive
   dl_dis_ind :      -> primitive
   dl_data_req : packet  -> primitive
   dl_data_ind : packet  -> primitive
   eval        : primitive -> Nat
   is_dl_est_req : primitive -> Bool
   is_dl_dis_req : primitive -> Bool
   is_dl_dis_cnf : primitive -> Bool
   is_dl_est_ind : primitive -> Bool
   is_dl_est_cnf : primitive -> Bool
   is_dl_dis_ind : primitive -> Bool
   is_dl_data_req : primitive -> Bool
   is_dl_data_ind : primitive -> Bool
   get_pack      : primitive -> packet
  eqns
  forall p : primitive,pa : packet
  ofsort packet
   get_pack(dl_data_req(pa)) = pa;
  ofsort Nat
   eval(dl_est_req) = 0;
   eval(dl_dis_req) = Succ(0);
   eval(dl_dis_cnf) = Succ(Succ(0));
   eval(dl_est_ind) = Succ(Succ(Succ(0)));
   eval(dl_est_cnf) = Succ(Succ(Succ(Succ(0))));
   eval(dl_dis_ind) = Succ(Succ(Succ(Succ(Succ(0)))));
   eval(dl_data_req(pa)) = Succ(Succ(Succ(Succ(Succ(Succ(0))))));
   eval(dl_data_ind(pa)) = Succ(Succ(Succ(Succ(Succ(Succ(Succ(0)))))));
  ofsort Bool
   is_dl_est_req(p) = (eval(p) eq 0);
   is_dl_dis_req(p) = (eval(p) eq Succ(0));
   is_dl_dis_cnf(p) = (eval(p) eq Succ(Succ(0)));
   is_dl_est_ind(p) = (eval(p) eq Succ(Succ(Succ(0))));
   is_dl_est_cnf(p) = (eval(p) eq Succ(Succ(Succ(Succ(0)))));
   is_dl_dis_ind(p) = (eval(p) eq Succ(Succ(Succ(Succ(Succ(0))))));
   is_dl_data_req(p) = (eval(p) eq Succ(Succ(Succ(Succ(Succ(Succ(0)))))));
```

```
      is_dl_data_ind(p) = (eval(p) eq Succ(Succ(Succ(Succ(Succ(Succ(Succ(0))))))));
endtype

type Queue  is Boolean
    formalsorts elt
    sorts queue
    opns
      empty    :               -> queue
      _+--_    : elt,queue  -> queue
      _--+_    : queue,elt   -> queue
      isempty  : queue          -> Bool
      firstone : queue          -> elt
      removefirst : queue       -> queue
    eqns
    forall pq,pq1,pq2 : queue , f , f1, f2 : elt
    ofsort Bool
      isempty(empty)   = true;
      isempty(f +-- pq) = false;
      isempty(pq --+ f) = false;
    ofsort elt
      firstone(pq --+ f) = f;
    ofsort queue
      f +-- empty      = empty --+ f;
      f1 +-- (pq --+ f2) = (f1 +-- pq) --+ f2;
      removefirst(pq --+ f) = pq;
endtype

type Paqueue is
    Queue actualizedby Packet using
    sortnames packet for elt
endtype
type Pacqueue is Paqueue renamedby
    sortnames pacqueue for queue
endtype
type Fqueue is
    Queue actualizedby Extframetype using
    sortnames eframe for elt
endtype
type Framequeue is Fqueue renamedby
    sortnames fqueue for queue
endtype
type Packe is Packet,NaturalNumber
    sorts pac
    opns
      mp : packet,Nat  -> pac
      snn : pac        -> Nat
      packe : pac      -> packet
    eqns
    forall s : Nat,p : packet
    ofsort Nat
      snn(mp(p,s)) = s;
    ofsort packet
      packe(mp(p,s)) = p;
endtype
type Pq is
    Queue actualizedby Packe using
    sortnames pac for elt
endtype
type Pqueue is Pq renamedby
```

```
     sortnames pqueue for queue
endtype

behaviour
    lapb[dl,phl] (adr(dte),adr(dce),n2,k,tmode,resolvecollis,senddmoption)
 where

   process timer[tm]  : exit(Bool,Bool)     :=
     tm ! start
    ;( i
      ;tm ! expired
      ;exit(any Bool,any Bool)
      [>
       tm ! reset
       ;exit(any Bool,any Bool)
     )
   endproc (* timer *)

    process timer1[tm]  : exit(Bool,Bool,eframe)     :=
           tm ! start
          ;( i
          ;tm ! expired
          ;exit(any Bool,any Bool,any eframe)
          [>
             tm ! reset
             ;exit(any Bool,any Bool,any eframe)
          )
   endproc (* timer1 *)

   process lapb[dl,phl] (from : adress, to : adress, n2 : Nat, k : Nat, tmode : mmode,resolvecollis : Nat,
                                                                        senddmoption : Bool) : noexit :=

         [k lt k(tmode)] ->
              (( connectionphase[dl,phl] (from,to,n2,k,tmode,resolvecollis,senddmoption)
            >>
                accept ok : Bool,donotwaitua : Bool in
               ( [ok] -> ( hide pp in
                             ((dataphase[dl,phl,pp] (donotwaitua,from,to,n2,k,tmode,empty of pqueue,empty of pqueue,
                                                  empty of pacqueue,resolvecollis)
                             [>
                                terminationphase[dl,phl,pp] (from,to,n2,k,tmode,resolvecollis)
                             )
                          |[pp]|
                        goterm[pp]
                      ))
                 []
                 [not(ok)] -> exit
               )
            )
            >>  lapb[dl,phl] (from,to,n2,k,tmode,resolvecollis,senddmoption)
       )
    where
     process connectionphase[dl,phl] (from : adress,to : adress, n2 : Nat,k : Nat, tmode : mmode,resolvecollis : Nat,
                                                                 senddmoption : Bool) : exit(Bool,Bool)  :=
       (disconnectedphase[phl] (from,to,n2,k,tmode)
        >> connectionphase[dl,phl] (from,to,n2,k,tmode,resolvecollis,senddmoption)
       )
       []
       calling[dl,phl] (from,to,n2,k,tmode,resolvecollis,senddmoption)
```

```
    []
    called[dl,phl](from,to,n2,k,tmode,resolvecollis)
    []
    (receiveincorrect[phl]
      >> connectionphase[dl,phl] (from,to,n2,k,tmode,resolvecollis,senddmoption))

  where
    process called[dl,phl](from : adress,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                  : exit(Bool,Bool) :=
         (phl ? sframe : eframe[(((is_sabm(get_f(sframe))) and (eq(tmode,basic))) or ((is_sabme(get_f(sframe)))
                                       and (eq(tmode,extended))) ) and (correct(sframe))]
          ;((i;phl ! m(make_ua(flagging,from,ctl_uframe(pf(((sframe)))),fc, flagging),noinfor,correct,notless,noabort)
          ;exit(true,false)
         )
       []
          i;phl ! m(make_dm(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort)
           ;exit(false,false)
        )
      )
    []
     phl ? sframe : eframe [(correct(sframe)) and ((is_dm(get_f(sframe))) and (pf(((sframe))) eq 0))]
     ;( (* request for link set up accepted *)
     i;calloption1[dl,phl](m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor ,correct,notless,noabort),
                                     from,to,n2,k,tmode,resolvecollis)
    []
     (* requeset for link set up refused  *)
    (i;phl ! m(make_disc(flagging,to,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort)
      ;exit(false,false)
    )))
  endproc  (* called *)

   process disconnectedphase[phl] (from : adress,to : adress , n2 : Nat,k : Nat,tmode : mmode) : exit :=
       ( (phl ? sframe : eframe [((correct(sframe)) and ((pf(sframe) eq 1 of Bit) and (eq(adrs(sframe),from))))
                                       or (is_disc(get_f(sframe)))];exit(pf(sframe)))
        |[phl]|
         (phl ? sframe : eframe [not(((eq(tmode,basic)) and (is_sabm(get_f(sframe)))) or ((eq(tmode,extended))
                                       and (is_sabme(get_f(sframe)))))];exit(pf(sframe)))
          >> accept b : Bit in
            phl ! m(make_dm(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
            ;exit
        )
      []
     (((phl ? sframe : eframe [not(((correct(sframe)) and ((pf(sframe) eq 1 of Bit) and (eq(adrs(sframe),from)))) or
                                     (is_disc(get_f(sframe)))) and not ((is_dm(get_f(sframe))) and (pf(sframe) eq 0 of Bit))]
        ;exit)
      |[phl]|
       (phl ? sframe : eframe [not(((eq(tmode,basic)) and (is_sabm (get_f(sframe)))) or ((eq(tmode,extended))
                                    and (is_sabme(get_f(sframe)))))];exit)
     ) )
 endproc (* disconnectedphase *)
 process calling[dl,phl] (from : adress, to : adress, n2 : Nat, k : Nat, tmode : mmode,resolvecollis : Nat,
                                                                              senddmoption : Bool)   : exit(Bool,Bool) :=
    dl ? p : primitive [is_dl_est_req(p)]
    ;(([eq(tmode,basic)] ->  (calloption1[dl,phl] (m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor
                                                           ,correct,notless,noabort),from, to,n2,k,tmode,resolvecollis))
      []
      [eq(tmode,extended)] -> (calloption1[dl,phl] (m(make_sabme(flagging,to,ctl_uframe(1),fc,flagging)
                                                           ,noinfor ,correct,notless,noabort),from,to,n2,k,tmode,resolvecollis))
     )
```

```
        []
         (calloption2[dl,phl] (from,to, n2,k,tmode,resolvecollis))
        []
         calloption3[dl,phl] (from,to,n2,k,tmode,resolvecollis,senddmoption)
        )
    endproc (* calling *)
    process calloption2[dl,phl] (from : adress,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                : exit(Bool,Bool) :=
         calloption1[dl,phl] (m(make_disc(flagging,to,ctl_uframe(1),fc,flagging),noinfor,correct,
                                   notless,noabort),from,to,n2,k,tmode,resolvecollis)
          >> accept ok : Bool,donotwait : Bool in
           ([ok] ->
             ([eq(tmode,basic)] ->
                (calloption1[dl,phl] (m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor,
                                        correct,notless,noabort),from,to,n2,k,tmode,resolvecollis))
               []
               [eq(tmode,extended)] ->
                (calloption1[dl,phl] (m(make_sabme(flagging,to,ctl_uframe(1),fc,flagging),noinfor,
                                         correct,notless,noabort),from,to,n2,k,tmode,resolvecollis)))
             []
             [not(ok)] ->
                 (exit(false,false))
                  )
      endproc  (* calloption2 *)

     process calloption3[dl,phl] (from : adress,to : adress ,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat,
                                         senddmoption : Bool) : exit(Bool,Bool) :=
          [senddmoption] ->
           (hide pp,p in
            senddm[phl,pp,p](from,to,0 of Nat,n2)
           |[phl,pp,p]|
           waitrespdm[dl,phl,pp,p](from,to,n2,k,tmode,resolvecollis) )
         where
          process senddm[phl,pp,p] (from,to : adress,init : Nat,n2 : Nat)      : exit(Bool,Bool) :=
             ([init lt n2] ->
              (hide t in
              (((((p ! Succ(0) of Nat
                   ;phl ! m(make_dm(flagging,to,ctl_uframe(0),fc,flagging),noinfor ,correct,notless,noabort)
                  ;t ! start
                  ;synchrespdm[phl,pp,t,p]
                 )
                ) [> (t ! expired;exit
                        >> senddm[phl,pp,p](from,to,Succ(init),n2)
                 )
            ))
           |[t]|
           timer[t]
           ))
         []
         [init eq n2] -> ( p ! 0 of Bit
                                ;exit(false,false)
          )               )
       where
         process synchrespdm[phl,pp,t,p] : exit(Bool,Bool) :=
           (pp ! 0 of Nat
            ;t ! reset
            ;exit(any Bool,any Bool)
           )
            []
```

```
        (pp ! Succ(0) of Nat
         ;(exit(any Bool,any Bool)
           []
           wwaituadm[phl]
          )
        )
        []
        (phl ? fram : eframe
         ;synchrespdm[phl,pp,t,p]
        )
       endproc
      endproc (* senddm *)
       process waitrespdm[dl,phl,pp,p](from : adress ,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                          : exit(Bool,Bool) :=
            (p ! Succ(0) of Nat
             ;phl ? f : eframe
             ;waitingresponsedm[dl,phl,pp,p](from,to,n2,k,tmode,resolvecollis)
            ) [> p ! 0 of Bit
                  ;exit(any Bool,any Bool)
       where
        process waitingresponsedm[dl,phl,pp,p](from : adress ,to : adress,n2 : Nat,k : Nat,tmode : mmode,
                               resolvecollis : Nat)    : exit(Bool,Bool) :=
             (notsabmdiscdm[phl,pp,p]
              |[phl,pp,p]|
              isdm[dl,phl,pp,p](from,to,n2,k,tmode,resolvecollis)
             |[phl,pp,p]|
              issabmofdm[dl,phl,pp,p] (from,to,n2,k,tmode)
            |[phl,pp,p]|
             isdiscofdm[dl,phl,pp,p](from,to,n2,k,tmode)
            )
      endproc (* waitingresponsedm *)
     endproc (* waitrespdm *)
    endproc (* calloption3 *)

   process notsabmdiscdm[phl,pp,p] : exit(Bool,Bool) :=
      (( phl ? sframe : eframe [((correct(sframe) and (is_sabm(get_f(sframe))))) or (((is_disc(get_f(sframe))) or
                                          is_dm(get_f(sframe))))]
         ;pp ? n : Nat
         ;(exit(any Bool,any Bool)
          []
          wwaituadm[phl])
      )
      []
      (phl ? sframe : eframe [not(((correct(sframe) and (is_sabm(get_f(sframe))))) or (((is_disc(get_f(sframe))) or
                                        is_dm(get_f(sframe)))))]
       ;notsabmdiscdm[phl,pp,p])
       )[> (p ! Succ(0) of Nat
            ;phl ? f : eframe
            ;notsabmdiscdm[phl,pp,p]
          )
     endproc (* notsabmdmf1uaf1disc *)

    process isdiscofdm[dl,phl,pp,p](from : adress,to : adress,n2 : Nat,k : Nat,tmode : mmode) : exit(Bool,Bool) :=
            ((phl ? f : eframe[not(correct(f) and is_disc(get_f(f)))]
             ;(isdiscofdm[dl,phl,pp,p](from,to,n2,k,tmode)
              []
              pp ? n : Nat
              ;(exit(any Bool,any Bool)
               []
```

```
                wwaituadm[phl])
              ))
           []
          (phl ? f : eframe[correct(f) and is_disc(get_f(f))]
               ;pp ! 0 of Nat
              ;exit(false,false))
           )[> (p ! Succ(0) of Nat
                ;phl ? f : eframe
                ;isdiscofdm[dl,phl,pp,p](from,to,n2,k,tmode)
             )
       endproc (* isdisc *)

       process issabmofdm[dl,phl,pp,p](from : adress,to : adress,n2 : Nat,k : Nat,tmode : mmode) : exit(Bool,Bool) :=
             ((phl ? f : eframe[not(correct(f) and (is_sabm(get_f(f)) or is_sabme(get_f(f))))]
              ;(issabmofdm[dl,phl,pp,p](from,to,n2,k,tmode)
            []
             pp ? n : Nat
             ;(exit(any Bool,any Bool)
             []
             wwaituadm[phl])
           ))
          []
          (phl ? f : eframe[correct(f) and (is_sabm(get_f(f)) or is_sabme(get_f(f)))]
           ;pp ! 0 of Nat
           ;exit(true,false))
          )[> (p ! Succ(0) of Nat
               ;phl ? f : eframe
               ;issabmofdm[dl,phl,pp,p](from,to,n2,k,tmode)
             )
        endproc (* issabm *)

        process isdm[dl,phl,pp,p](from : adress,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                  : exit(Bool,Bool) :=
           ((phl ? f : eframe[not(correct(f) and (is_dm(get_f(f))))]
             ;(isdm[dl,phl,pp,p](from,to,n2,k,tmode,resolvecollis)
            []
             pp ? n : Nat
             ;exit(any Bool,any Bool)
            ))
          []
           (phl ? f : eframe[correct(f) and (is_dm(get_f(f)) and (pf(f) eq 1 of Bit))]
            ;pp ! 0 of Nat
            ;exit(false,false)
          )
         []
         (phl ? f : eframe[correct(f) and (is_dm(get_f(f)) and (pf(f) eq 0 of Bit))]
          ;pp ! Succ(0) of Nat
         ;([eq(tmode,basic)] ->  (calloption1[dl,phl] (m(make_sabm(flagging ,to,ctl_uframe(1),fc,flagging),noinfor,
                                                   correct,notless,noabort),from,to,n2,k,tmode,resolvecollis))
           []
           [eq(tmode,extended)] -> (calloption1[dl,phl] (m(make_sabme(flagging,to,ctl_uframe(1),fc,flagging),noinfor,
                                                     correct,notless,noabort),from,to, n2,k,tmode,resolvecollis))
             ))
         )[> (p ! Succ(0) of Nat
               ;phl ? f : eframe
               ;isdm[dl,phl,pp,p](from,to,n2,k,tmode,resolvecollis)
             )
         endproc
```

```
process sacceptedsabm[med] : exit(eframe) :=
    (* accept frame either SABM,DISC,UA or DM *)
    med ? sframe : eframe [(is_sabm(get_f(sframe))) or ((is_disc(get_f(sframe))) or ((is_ua(get_f(sframe))) or
                                   (is_dm(get_f(sframe)))))]
    ;exit((sframe))
endproc  (* sacceptedsabm *)

 process receiveincorrect[phl] : exit :=
    phl ? f : eframe[not(correct(f))]
    ;exit
 endproc (* receiveincorrect *)
endproc (* connectionphase *)

 process goterm[pp] : exit :=
     (pp ! termine
     ;pp ! causetermine
     ;exit
      )
     []
     exit
 endproc

 process terminationphase[dl,phl,pp](from : adress,to : adress, n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                   : exit :=
     ( dl ! dl_dis_req      (* user entity iniates disconnection *)
       ;calloption1[dl,phl] (m(make_disc(flagging,to,ctl_uframe(1),fc ,flagging),noinfor ,correct,notless,noabort),
                                          from,to,n2,k,tmode,resolvecollis)
        >> accept k : Bool,donotwait : Bool in
       dl ! dl_dis_cnf
      ;exit
    )
    []
   (phl ? sframe : eframe[(correct(sframe)) and (is_disc(get_f(sframe)))]
    ;dl ! dl_dis_ind
    ;phl ! m(make_ua(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort)
    ;exit
   )
   []
   pp ! causetermine
   ;dl ! dl_dis_ind
  ;exit
endproc   (* terminationphase *)
 process wwaituadm[phl] : exit(Bool,Bool) :=
          phl ? fram : eframe
          ;(exit(any Bool,any Bool)
            []
            wwaituadm[phl]
           )
 endproc

 process wwaitua[phl] : exit(Bool,Bool,eframe) :=
          phl ? fram : eframe
         ;(exit(any Bool,any Bool,m(make_ua(flagging,a,ctl_uframe(1),fc,flagging),noinfor,correct,notless,noabort))
          []
          wwaitua[phl]
         )
  endproc

 process notsabmdiscuaf1dmf1[phl,pp,p] : exit(Bool,Bool,eframe) :=
```

```
   ((( phl ? sframe : eframe [ ((correct(sframe) and (is_sabm(get_f(sframe))) or (((is_disc(get_f(sframe))) or
                                   ((is_ua(get_f(sframe))) and (pf(sframe) eq 1 of Bit)))) or
                                   (((is_dm(get_f(sframe))) and (pf(sframe) eq 1 of Bit)))))]
  ;((phl ? f : eframe
   ;pp ? n : Nat
   ;exit(any Bool,any Bool,any eframe)
    )
    []
    (pp ? n : Nat
   ;exit(any Bool,any Bool,any eframe)
    ))
   )
   []
   (phl ? sframe : eframe [not( ((correct(sframe) and  (is_sabm(get_f(sframe)))) or ((is_disc(get_f(sframe))) or
                                 (((((is_ua(get_f(sframe))) and (pf(sframe) eq 1 of Bit)))) or  (is_dm(get_f(sframe)) and
                                 (pf(sframe) eq 1 of Bit))))))]
    ;notsabmdiscuaf1dmf1[phl,pp,p]
   ))
   [> (p ! Succ(Succ(0)) of Nat
       ;phl ? f : eframe
       ;(exit(any Bool,any Bool,any eframe)
         []
         wwaitua[phl]
      ))
    )[> (p ! Succ(0) of Nat
         ;phl ? f : eframe
         ;notsabmdiscuaf1dmf1[phl,pp,p]
        )
endproc (* notsabmdmf1uaf1disc *)

process isuaf1[phl,pp,p] : exit(Bool,Bool,eframe) :=
     ((((phl ? f : eframe[not(correct(f) and (is_ua(get_f(f)) and (pf(f) eq 1 of Bit)))]
         ;(isuaf1[phl,pp,p]
            []
            pp ? n : Nat
            ;exit(any Bool,any Bool,any eframe)
           )
     ))
 []
 (phl ? f : eframe[correct(f) and (is_ua(get_f(f)) and  (pf(f) eq 1 of Bit))]
  ;pp ! 0 of Nat
 ;exit(true,false,f)
 ))
  [> (p ! Succ(Succ(0)) of Nat
      ;phl ? f : eframe
      ;(exit(any Bool,any Bool,any eframe)
        []
        wwaitua[phl]
     ))
   )[> (p ! Succ(0) of Nat
        ;phl ? f : eframe
        ;isuaf1[phl,pp,p]
       )
endproc (* isuaf1 *)

process isdmf1[phl,pp,p](fr : eframe) : exit(Bool,Bool,eframe) :=
    ((((phl ? f : eframe[not(correct(f) and (is_dm(get_f(f)) and  (pf(f) eq 1 of Bit)))]
        ;(isdmf1[phl,pp,p](fr)
     []
```

```
       pp ? n : Nat
      ;exit(any Bool,any Bool,any eframe)
    )))
    []
   (phl ? f : eframe[correct(f) and (is_dm(get_f(f)) and
    (pf(f) eq 1 of Bit))]
   ;pp ! 0 of Nat
   ;exit(resp(fr),false,f)
   ))
    [> (p ! Succ(Succ(0)) of Nat
        ;phl ? f : eframe
        ;(exit(any Bool,any Bool,any eframe)
          []
          wwaitua[phl]
      ))
     )[> (p ! Succ(0) of Nat
          ;phl ? f : eframe
          ;isdmf1[phl,pp,p](fr)
        )
  endproc (* isdmf1 *)

  process issabm[dl,phl,pp,p](fr : eframe,from :adressto : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                   : exit(Bool,Bool,eframe) :=
      (((((phl ? f : eframe[correct(f) and (is_sabm(get_f(f)) or is_sabme(get_f(f)))]
            ;exit(collis(f,fr),f)
            >> accept coll : Bool,f : eframe in
        [not(coll)] ->
                    (pp ! Succ(0) of Nat
                    ;p ! Succ(Succ(0)) of Nat
                    ;phl ! m(make_dm(flagging,from,ctl_uframe(pf(fr)),fc,flagging),noinfor,correct,notless,noabort)
                    ;exit(true,false,f)
                   )
        []
        [coll] -> (collision[dl,phl,pp,p](fr,from,to,n2,k,tmode,resolvecollis)
                   >> accept b1 : Bool, b2 : Bool in
                       exit(b1,b2,f)
  )))
  []
  (phl ? f : eframe[not(correct(f) and (is_sabm(get_f(f))
   or is_sabme(get_f(f))))]
   ;(issabm[dl,phl,pp,p](fr,from,to,n2,k,tmode,resolvecollis)
   []
    pp ? n : Nat
    ;exit(any Bool,any Bool,any eframe)
   )))
  [> (p ! Succ(Succ(0)) of Nat
        ;phl ? f : eframe
        ;(exit(any Bool,any Bool,any eframe)
          []
          wwaitua[phl]
      ))
     )[> (p ! Succ(0) of Nat
          ;phl ? f : eframe
          ;issabm[dl,phl,pp,p](fr,from,to,n2,k,tmode,resolvecollis)
        ))
 endproc (* issabm *)

 process isdisc[dl,phl,pp,p](fr : eframe,from : adress ,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                   : exit(Bool,Bool,eframe) :=
```

```
    ((((phl ? f : eframe[not(correct(f) and is_disc(get_f(f)))]
        ;(isdisc[dl,phl,pp,p](fr,from,to,n2,k,tmode,resolvecollis)
         []
         pp ? n : Nat
   ;exit(any Bool,any Bool,any eframe)
  )))
  []
  ((phl ? f : eframe[correct(f) and is_disc(get_f(f))]
   ;exit(collis(f,fr),f)
   >> accept coll : Bool,f : eframe in
        ([not(coll)] ->
                    (pp ! Succ(0) of Nat
                    ;p ! Succ(Succ(0)) of Nat
                    ;phl ! m(make_dm(flagging,from,ctl_uframe(pf(fr)),fc,flagging),noinfor,correct,notless,noabort)
                    ;exit(false,false,f)
                    )
      []
      [coll] -> (collision[dl,phl,pp,p](fr,from,to,n2,k,tmode,resolvecollis)
                     >> accept b1 : Bool, b2 : Bool in
                        exit(b1,b2,f))
  ))))
  [> (p ! Succ(Succ(0)) of Nat
     ;phl ? f : eframe
      ;(exit(any Bool,any Bool,any eframe)
        []
        wwaitua[phl]
     ))
    )[> (p ! Succ(0) of Nat
         ;phl ? f : eframe
         ;isdisc[dl,phl,pp,p](fr,from,to,n2,k,tmode,resolvecollis)
       )
endproc (* isdisc *)

process collision[dl,phl,pp,p](f : eframe,from : adress ,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                      : exit(Bool,Bool) :=
       pp ! Succ(0) of Nat
       ;p ! Succ(Succ(0)) of Nat
       ;phl ! m(make_ua(flagging,from,ctl_uframe(pf(f)),fc,flagging) ,noinfor,correct,notless,noabort)
       ;([resolvecollis eq 0 of Nat] ->
             waitua[dl,phl,pp,p](f,from,to,n2,k,tmode,resolvecollis)
        []
        [resolvecollis eq Succ(0) of Nat] ->
             waituaortimeout[dl,phl,pp,p]
        []
        [resolvecollis eq Succ(Succ(0)) of Nat] ->
             donotwait
    )
endproc (* collision *)

process waitua[dl,phl,pp,p](f : eframe,from : adress ,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                      : exit(Bool,Bool) :=
      hide t in
       ((( t ! start
           ;waitua1[dl,phl,pp,p,t]
         )
    [>( t ! expired
         ;calloption1[dl,phl](f,from,to,n2,k,tmode,resolvecollis)
       )
       )
```

```
      |[t]|
      timer[t])
  endproc (* waitua *)

    process waitua1[dl,phl,pp,p,t] : exit(Bool,Bool) :=
           ( (phl ? sframe : eframe [((is_ua(get_f(sframe))) and (pf((sframe)) eq 1)) and correct(sframe)]
              ;t ! reset
           ;exit(true,false)
           )
          []
            (* ignore received frame if not ua with f=1 *)
          (phl ? sframe : eframe [(incorrect(sframe))  or (((is_ua(get_f(sframe))) and (pf((sframe)) eq 0))
                                           or (not(is_ua(get_f(sframe)))))]
            ;waitua1[dl,phl,pp,p,t]
          ))
    endproc  (* waitua1 *)

  process waituaortimeout[dl,phl,pp,p] : exit(Bool,Bool) :=
      hide t in
       ((( t ! start
            ;waitua1[dl,phl,pp,p,t]
         )
        [>
        (t ! expired    (* returns true if timeout occurs *)
         ;exit(true,false))
    )
    |[t]|
     timer[t])
 endproc  (* waituaortimeout  *)

 process donotwait : exit(Bool,Bool) :=
     exit(true,false)
 endproc (* donotwait *)

 process sendcde[phl,pp,p] (f : eframe,init : Nat,n2 : Nat)  : exit(Bool,Bool,eframe) :=
      ([init lt n2] ->
         (hide t in
           ((((p ! Succ(0) of Nat
                ;phl ! f
                ;t ! start
                ;synchresp[phl,pp,t,p]
               ) [> (t ! expired;exit
                       >> sendcde[phl,pp,p](f,Succ(init),n2)
               )
         ))
        |[t]|
        timer1[t]
        ))
      []
      [init eq n2] -> ( p ! 0 of Bit
                     ;exit(false,false,m(make_ua(flagging,a,ctl_uframe(1),fc,flagging),noinfor,correct,notless,noabort))
    )              )
      where
        process synchresp[phl,pp,t,p] : exit(Bool,Bool,eframe) :=
            (( pp ! 0 of Nat
             ;t ! reset
             ;exit(any Bool,any Bool,any eframe)
            )
            []
```

```
               (pp ! Succ(0) of Nat
                ;t ! reset
                ;p ! Succ(Succ(0)) of Nat
                ;phl ? fram : eframe
                ;(exit(any Bool,any Bool,any eframe)
                  []
                  wwaitua[phl]
                 )
               ))
              [] (phl ? fram : eframe
                ;synchresp[phl,pp,t,p]
                 )
          endproc
     endproc (* sendcde *)

 process calloption1[dl,phl] (fr : eframe,from : adress ,to : adress,n2 : Nat , k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                          : exit(Bool,Bool) :=
       hide pp,p in
         sendcde[phl,pp,p](fr,0 of Nat,n2)
         |[phl,pp,p]|
         waitresp[dl,phl,pp,p](fr,from,to,n2,k,tmode,resolvecollis)
            >> accept b1 : Bool, b2 : Bool,f : eframe in
        ([is_dm(get_f(f)) and is_sabm(get_f(fr))] ->
              (i;calloption1[dl,phl](fr,from,to,n2,k,tmode,resolvecollis)
              []
              i;exit(b1,b2))
          []
          [not(is_dm(get_f(f)) and is_sabm(get_f(fr)))] ->
            exit(b1,b2))
       where
  process waitresp[dl,phl,pp,p](f : eframe,from : adress,to : adress,n2 : Nat,k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                 : exit(Bool,Bool,eframe) :=
          (p ! Succ(0) of Nat
           ;phl ? f : eframe
          ;waitingresponse[dl,phl,pp,p](f,from,to,n2,k,tmode,resolvecollis)
           ) [> p ! 0 of Bit
                ;exit(any Bool,any Bool,any eframe)
        where
        process waitingresponse[dl,phl,pp,p](f : eframe ,from : adress,to : adress,n2 : Nat,k : Nat,tmode : mmode,
                                                          resolvecollis : Nat)   : exit(Bool,Bool,eframe) :=
             (issabm[dl,phl,pp,p] (f,from,to,n2,k,tmode,resolvecollis)
             |[phl,pp,p]|
              isdisc[dl,phl,pp,p](f,from,to,n2,k,tmode,resolvecollis)
             |[phl,pp,p]|
              notsabmdiscuaf1dmf1[phl,pp,p]
             |[phl,pp,p]|
              isuaf1[phl,pp,p]
              |[phl,pp,p]|
               isdmf1[phl,pp,p](f)
            )
       endproc (* waitingresponse *)
      endproc (* waitresp *)
    endproc (* calloption1 *)

  process dataphase[dl,phl,pp](notwait : Bool,from: adress,to : adress,n2 : Nat,k : Nat,tmode : mmode,q : pqueue,
                                          qdls : pqueue,qdlr : pacqueue,resolvecollis : Nat)     :      noexit :=
       hide p in
     (
      ( inforphase[dl,phl,p](notwait,false,0 of Nat,0 of Nat,from,to,n2,k,tmode,0 of Nat,0 of Nat,0 of Nat,q,
```

```
                                   qdls,false,false,false,false,qdlr,false)
      [> linkreset[dl,phl,p,pp](from,to,n2,k,tmode,resolvecollis)
    )
  |[p]|
  causelinkreset[p]
  ) >> stop
  where
   process receiveinvalid1[phl] : exit :=
     phl ? f : eframe[not(is_valid(f))]
     ;exit
   endproc (* receiveinvalid *)

   process receiveinvalid[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                                        retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode)
                                     : exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,
                                          Bool,Nat,frmrreason,Nat,Bit) :=
      phl ? f : eframe[not(is_valid(f))]
      ;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,
           0 of Nat,0 of Bit)
   endproc (* receiveinvalid *)

   process causelinkreset[p] : exit :=
      p ! lreset ? n : Nat ? reasonfrmr : frmrreason ? q : pqueue ? qdls : pqueue ? qdlr : pacqueue ? b : Bit
              ? vs : Nat ? vr : Nat ? lu : Nat  ;p ! causelreset ! n  ! reasonfrmr ! q ! qdls ! qdlr ! b  ! vs  ! vr  ! lu
       ;exit
   endproc

   process linkreset[dl,phl,p,pp](from : adress,to : adress,n2 : Nat,  k : Nat,tmode : mmode,resolvecollis : Nat)
                                                                                                                      : noexit :=
     p ! causelreset ? n : Nat ? reasonfrmr : frmrreason ? q : pqueue ? qdls : pqueue ? qdlr : pacqueue
                     ? b : Bit ? vs : Nat ? vr : Nat ? lu : Nat
     ;([n eq 0 of Nat] ->
      (calloption1[dl,phl] (m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor,
                                                  correct,notless,noabort),from,to,n2,k,tmode,resolvecollis)
        >> accept ok : Bool,notwait : Bool in
          ([ok] -> dataphase[dl,phl,pp] (notwait,from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
            []
            [not(ok)] ->
                      pp ! termine
                      ;stop
     )
    )
   []
   [(n eq Succ(0))] ->    (* frmr sent *)
     sendfrmr1[dl,phl,pp,p](vs,vr,lu,0 of Nat,from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis)
   []
   [((n eq Succ(Succ(0)) of Nat) or (n eq Succ(Succ(Succ(0))) of Nat))] ->
      rcvfrmroruaordm[dl,phl,pp](from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
   []
   [n eq (Succ(Succ(0)) * Succ(Succ(0)) of Nat)] ->
     rcvsabm[dl,phl,pp](from,to,n2,k,tmode,q,qdls,qdlr,b,resolvecollis)
  )
   where
     process rcvfrmroruaordm[dl,phl,pp](from : adress,to : adress,n2 : Nat ,k : Nat,tmode : mmode,q : pqueue,
                                                  qdls : pqueue,qdlr : pacqueue,resolvecollis : Nat) : noexit :=
            (* ask peer entity to initiate link set up and enter disconnected phase *)
         ( i; phl ! m(make_dm(flagging,from,ctl_uframe(0),fc,flagging),noinfor,correct,notless,noabort)
           ;pp ! termine
           ;stop
```

```
      )
     []
      (* initiate link reset procedure *)
      i; calloption1[dl,phl](m(make_sabm(flagging ,to,ctl_uframe(1),fc,flagging),noinfor
                              ,correct,notless,noabort),from,to,n2,k,tmode,resolvecollis)
          >> accept ok : Bool,notwait : Bool in
         ([ok] -> dataphase[dl,phl,pp](notwait,from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
          []
          [not(ok)] ->
                         pp ! termine
                         ;stop
       )
    endproc (* rcvfrmroruaordm *)

   process rcvsabm[dl,phl,pp](from : adress,to : adress,n2 : Nat, k : Nat,tmode : mmode,q : pqueue,
                                               qdls : pqueue,qdlr : pacqueue,b : Bit,resolvecollis : Nat) : noexit :=
          (* remain in information transfer phase *)
        (i;phl ! m(make_ua(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
          ;dataphase[dl,phl,pp](false,from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
       )
     []
            (* enters disconnected *)
          i;phl ! m(make_dm(flagging,to,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
           ;pp ! termine
           ;stop
  endproc   (* rcvsabm *)

  process sendfrmr1[dl,phl,pp,p](vs : Nat,vr : Nat,lu : Nat,init : Nat,from : adress,to : adress,
                                                   n2 : Nat,k : Nat,tmode : mmode,reasonfrmr : frmrreason,q : pqueue,
                                                   qdls : pqueue,qdlr : pacqueue,resolvecollis : Nat) : noexit :=
     [init lt n2] ->
      (hide t in
       ((phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
          ;t ! start
          ;waitrespfrmr[dl,phl,pp,t,p](vs,vr,lu,0 of Nat,from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis)
        )
      [> t ! expired
         ;sendfrmr1[dl,phl,pp,p](vs,vr,lu,Succ(init),from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis)
     )
    |[t]|
    timer[t]
    )
  []
  [init eq n2] ->       (* if frmr sent n2 times without response reset the link *)
     (calloption1[dl,phl](m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor,
                                      correct,notless,noabort),from,to,n2,k,tmode,resolvecollis)
      >> accept ok : Bool,notwait : Bool in
        ([ok] -> dataphase[dl,phl,pp](notwait,from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
         []
         [not(ok)] -> (* enter disconnected phase *)
                   pp ! termine
                  ;stop
       )
     )
  endproc (* sendfrmr1 *)

  process waitrespfrmr[dl,phl,pp,t,p]  (vs : Nat,vr : Nat,lu : Nat,call : Nat,from : adress,to : adress,
                                                          n2 : Nat,k : Nat,tmode : mmode,reasonfrmr : frmrreason,q : pqueue,
                                                          qdls : pqueue,qdlr : pacqueue,resolvecollis : Nat) : noexit :=
```

```
    (receiveinvalid1[phl]
      >> waitrespfrmr[dl,phl,pp,t,p](vs,vr,lu,call,from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis)
 )
 []
 (sendfrmr[phl](vs,vr,lu,k,from,to,n2,q,qdls,qdlr)
  >> accept ind : Nat,rs : frmrreason,q : pqueue,qdls : pqueue,qdlr : pacqueue,b : Bit in
      phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
      ;waitrespfrmr[dl,phl,pp,t,p](vs,vr,lu,call,from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis)
 )
 []
 ((phl ?  f : eframe[(correct(f) and (eq(adrs(f),from))) and (pf(f) eq 1 of Bit)];exit
  |[phl]|
  notfrmrsabmdmdisc[phl]
 ) >>
    phl ! m(make_frmr(flagging,from,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
   ;waitrespfrmr[dl,phl,pp,t,p](vs,vr,lu,call,from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis)
 )
 []
 ((phl ? f : eframe[correct(f) and ((eq(adrs(f),from)) or ((eq(adrs(f),to)) and (pf(f) eq 0 of Bit)))];exit
   |[phl]|
   notfrmrsabmdmdisc[phl]
  ) >> waitrespfrmr[dl,phl,pp,t,p](vs,vr,lu,call,from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis)
 )
 []
((phl ? f : eframe[correct(f)];exit(f)
  |[phl]|
  frmrsabmdmdisc[phl]
 ) >> accept f : eframe in
    (([call eq 0 of Nat] ->( t ! reset
                                        ;exit
                                       )
     []
     [call eq Succ(0) of Nat] -> exit
    )
    >> ([(is_sabm(get_f(f)) or is_sabme(get_f(f)))] ->
      ( (i;phl ! m(make_ua(flagging,to,ctl_uframe(pf(f)),fc,flagging),noinfor,correct,notless,noabort)
        ;dataphase[dl,phl,pp](false,from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
        )
       []
       (i;phl ! m(make_dm(flagging,to,ctl_uframe(pf(f)),fc,flagging),noinfor,correct,notless,noabort)
        ;pp ! termine
        ;stop
        )

     )
     []
     ([is_dm(get_f(f)) and (pf(f) eq 1 of Bit)] ->
          (waitrespfrmr[dl,phl,pp,t,p](vs,vr,lu,call,from,to,n2,k,tmode,reasonfrmr,q,qdls,qdlr,resolvecollis))
     )
     []
     ([is_dm(get_f(f)) and (pf(f) eq 0 of Bit)] ->         (* remote entitty asks local entity to initiate lin set up *)
       ((i;phl ! m(make_disc(flagging,to,ctl_uframe(1),fc,flagging),noinfor,correct,notless,noabort)
        ;pp ! termine
        ;stop
        )
        []
       (i;calloption1[dl,phl](m(make_sabm(flagging ,to,ctl_uframe(1),fc,flagging),noinfor,
                                        correct,notless,noabort),from,to,n2,k,tmode,resolvecollis)
        >> accept ok : Bool,notwait : Bool in
```

```
            ([not(ok)] ->
                    ( pp ! termine
                     ;stop
                    )
             []
             [ok] ->
               ((hide pp in
                 (dataphase[dl,phl,pp](notwait,from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
                  [> terminationphase[dl,phl,pp](from,to,n2,k,tmode,resolvecollis)
                 )
                 |[pp]|
                 goterm[pp]
               ) >> stop
               )
            )
        )
       ))
      []
      ([is_frmr(get_f(f))] ->         (* collision *)
        (* send DM to ask remote entitty link set up and enter disconnected phase *)
        ((i;phl ! m(make_dm(flagging,from,ctl_uframe(0),fc,flagging),noinfor,correct,notless,noabort)
        ;pp ! termine
        ;stop
        )
       []
       (* or cause link reset *)
       i;calloption1[dl,phl](m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor,
                                        correct,notless,noabort),from,to,n2,k,tmode,resolvecollis)
           >> accept ok : Bool,notwait : Bool in
            ([ok] ->
                  dataphase[dl,phl,pp](notwait,from,to,n2,k,tmode,q,qdls,qdlr,resolvecollis)
             []
             [not(ok)] ->  (* enter disconnect state *)
                 pp ! termine
                ;stop
            )
       )))))
    where
       process frmrsabmdmdisc[phl] : exit(eframe) :=
            phl ? f : eframe[((is_sabm(get_f(f)) or is_sabme(get_f(f))) or (is_frmr(get_f(f)) or is_dm(get_f(f))))]
           ;exit(f)
       endproc (* frmrsabmdm  *)

         process notfrmrsabmdmdisc[phl] : exit :=
              phl ? f : eframe[not(((is_sabm(get_f(f)) or is_sabme(get_f(f))) or (is_frmr(get_f(f)) or
                                     is_dm(get_f(f)))) or is_disc(get_f(f)))]
              ;exit
         endproc (* notfrmrsabmdm  *)
  endproc  (* waitrespfrmr *)
 endproc (* linkreset *)

    process sendfrmr[phl](vs : Nat,vr : Nat,lu : Nat,k : Nat,from : adress,to: adress,n2 : Nat,q : pqueue,
                                      qdls : pqueue, qdlr : pacqueue) : exit(Nat,frmrreason,pqueue,pqueue,pacqueue,Bit) :=
             (( (phl ? f : eframe[(is_valid(f) and not_abort(f)) and (((not(is_i(get_f(f)))) and (is_inforr(f))) or
                                       (not(corrl(f))))]
               ;exit(f)
            )
           []
            ( phl ? f : eframe[(is_valid(f) and not_abort(f)) and is_undef_ctl(get_ctl(get_f(f)))]
```

```
                ;exit(f)
              )
             []
               ( phl ? f : eframe[(is_valid(f) and not_abort(f)) and notvalid_nrf(f,vs,lu,k)]
       ;exit(f)           (* nr of frame not valid *)
     )
     []
     (phl ? f : eframe[(is_valid(f) and not_abort(f)) and (is_i(get_f(f)) and  (not(corrl(f))))]
       ;exit(f)
     ))
    >> accept f : eframe in
     ( ( ([(is_valid(f) and not_abort(f)) and (((not(is_i(get_f(f)))) and (is_inforr(f))) or (not(corrl(f))))] ->
               (exit(1,any Bit,any Bit,1,any Bit)) (* returns x=1,w=1 *)
         []
         [not((is_valid(f) and not_abort(f)) and (((not(is_i(get_f(f)))) and (is_inforr(f))) or (not(corrl(f)))))] ->
                 ([(is_valid(f) and not_abort(f)) and is_undef_ctl(get_ctl(get_f(f)))] ->
                         (exit(0 of Bit,any Bit,any Bit,1,any Bit)) (* returns x=0 *)
                  []
                  [(is_valid(f) and not_abort(f)) and is_undef_ctl(get_ctl(get_f(f)))] ->
                          (exit(0 of Bit,any Bit,any Bit,0 of Bit,any Bit)) (* returns x=0 w=0 *)
                )
       )
       |||
       ([(is_valid(f) and not_abort(f)) and notvalid_nrf(f,vs,lu,k)] ->
                   (exit(any Bit,any Bit,1,any Bit,any Bit)) (* returns z=1 *)
        []
        [not((is_valid(f) and not_abort(f)) and notvalid_nrf(f,vs,lu,k))] ->
                   (exit(any Bit,any Bit,0 of Bit,any Bit,any Bit)) (* returns z=0 *)
      )
       |||
       ([(is_valid(f) and not_abort(f)) and (is_i(get_f(f)) and (not(corrl(f))))] ->
                  (exit(any Bit,1,any Bit,any Bit,any Bit)) (* returns y=1 *)
         []
         [not((is_valid(f) and not_abort(f)) and (is_i(get_f(f)) and (not(corrl(f)))))] ->
              (exit(any Bit,0 of Bit,any Bit,any Bit,any Bit)) (* returns y =0 *)
       )
       |||
       ([(pf(f) eq 1) and (eq(adrs(f),from))] ->  (* commande frame so c/r = 0 *)
                 (exit(any Bit,any Bit,any Bit,any Bit,0 of Bit))
        []
        [not((pf(f) eq 1) and (eq(adrs(f),from)))] ->  (* response frame so c/r=1 *)
                 (exit(any Bit,any Bit,any Bit,any Bit,1))
       )
      ) >> accept x, y, z, w, cr  : Bit in
        exit(Succ(0) of Nat,make_reason(ctrl(f),0 of Bit,vs,cr,vr,x,y,z,w,0 of Bit),q,qdls,qdlr,1 of Bit)
    )
    )
   endproc  (* sendfrmr *)

   process inforphase[dl,phl,p] (notwait : Bool,reject : Bool,vsp : Nat,init : Nat,from : adress,to : adress, n2 : Nat,
                                                k : Nat,tmode : mmode,vs : Nat, vr : Nat, lu : Nat,q : pqueue,qdls : pqueue,starttime :
                                               Bool, rbusy : Bool,lbusy : Bool,chkpt : Bool,qdlr: pacqueue,retrans : Bool) : noexit :=
         hide t in
           (infphase[dl,phl,p,t](notwait,false,0 of Nat,0 of Nat,from,to,n2,k,tmode,0 of Nat,0 of Nat,0 of Nat,q,
                                        qdls,false,false,false,false,qdlr,false)
           |[t]|
           timer[t]
          )
  where
```

```
  process infphase[dl,phl,p,t] (notwait : Bool,reject : Bool,vsp : Nat,init : Nat,from : adress,to : adress, n2 : Nat,
                                     k : Nat,tmode : mmode,vs : Nat, vr : Nat, lu : Nat,q : pqueue,qdls : pqueue,starttime :
                                     Bool,rbusy : Bool,lbusy : Bool,chkpt : Bool,qdlr: pacqueue,retrans : Bool) : noexit :=
        (comX.25[dl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
         []
   sendframe[phl,t](vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
  []
   receiveframe[phl,t](vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
 []
 timeout[phl,t] (n2,vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
 )[> rcvtoreset[phl,p](notwait,chkpt,from,to,n2,k,tmode,vs,vr,lu,q,qdls,qdlr,rbusy)
  >>  accept
     vs : Nat,vr : Nat,lu : Nat,vsp : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,starttime : Bool, rbusy : Bool,
     lbusy : Bool,chkpt : Bool,retransBool,reject : Bool,init : Nat,notwait : Bool,typ : Nat,
     rs : frmrreason,ind : Nat,b : Bit
 in
([typ eq 0 of Nat] ->
 (infphase[dl,phl,p,t]   (notwait,reject,vsp,init,from,to,n2,k,tmode,vs,vr,lu,q,qdls,starttime,
                            rbusy,lbusy,chkpt,qdlr,retrans)
 )
 []
 [typ eq Succ(0) of Nat] ->
     (hide t in
      t ! start
      ;infphase[dl,phl,p,t] (notwait,reject,vsp,init,from,to,n2,k,tmode,vs,vr,lu,q,qdls,starttime,
                            rbusy,lbusy,chkpt,qdlr,retrans)
     )
   []
    [typ eq Succ(Succ(0)) of Nat] ->
     (hide t in
       t ! start
        ;infphase[dl,phl,p,t] (notwait,reject,vsp,init,from,to,n2,k,tmode,vs,vr,lu,q,qdls,starttime,
                            rbusy,lbusy,chkpt,qdlr,retrans)
     |[t]|
      timer[t]
     )
   []
   [typ eq Succ(Succ(Succ(0))) of Nat] ->
    (hide t in
       infphase[dl,phl,p,t]  (notwait,reject,vsp,init,from,to,n2,k,tmode,vs,vr,lu,q,qdls,starttime,
                            rbusy,lbusy,chkpt,qdlr,retrans)
      |[t]|
     timer[t]
     )
    []
    [typ eq Succ(Succ(Succ(Succ(0)))) of Nat] ->
       (p ! lreset ! ind ! rs ! empty of pqueue ! qdls ! qdlr ! b !  vs ! vr ! lu
       ;stop
       )
 )
     where
   process comX.25[dl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                          reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode)
                          : exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,
                                 Bool,Nat,frmrreason,Nat,Bit) :=
       recvpack[dl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
        []
      deliver[dl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
  endproc
```

```

process sendframe[phl,t]  (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                                reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode)
                : exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,Bool,
                                                                        Nat,frmrreason,Nat,Bit) :=
    sendi[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
    []
    sendrr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
    []
    sendrnr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
endproc

process receiveframe[phl,t](vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr :pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                                reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode)
                                :  exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                                        Nat,Bool,Nat,frmrreason,Nat,Bit)  :=
      receiveinvalid[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,
                              notwait,init,k,to,from,tmode)
     []
      receivevalidi[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,
                              notwait,init,k,to,from,tmode)
     []
      rcviandbusy[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,
                              notwait,init,k,to,from,tmode)
     []
      receivernr[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,
                           notwait,init,k,to,from,tmode)
     []
      receiverr[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,
                          notwait,init,k,to,from,tmode)
     []
      receiverej[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,
                          notwait,init,k,to,from,tmode)
     []
    sendrej[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,
                     notwait,init,k,to,from,tmode)
 endproc

process rcvtoreset[phl,p] (notwait : Bool,chkpt : Bool, from : adress,to : adress,n2 : Nat ,k : Nat,tmode :mmode,
                                vs : Nat,vr : Nat,lu : Nat,q : pqueue,qdls : pqueue,qdlr  : pacqueue,rbusy : Bool): noexit
      (receiveunsolicit[phl](notwait,chkpt,to,q,qdls,qdlr)
      []
      receivesabm[phl] (q,qdls,qdlr)
      []
       receiveuaordm[phl] (q,qdls,qdlr)
       []
       sendfrmr[phl](vs,vr,lu,k,from,to,n2,q,qdls,qdlr)
       []
       receivefrmr[phl](from,to,n2,k,tmode,vs,vr,lu,q,qdls,qdlr,rbusy)
         ) >> accept ind : Nat,rs : frmrreason,q : pqueue,qdls : pqueue,qdlr : pacqueue,b : Bit in
       (p ! lreset ! ind ! rs ! empty of pqueue ! qdls ! qdlr ! b ! vs ! vr ! lu
         ;stop
        )
where
process receiveunsolicit[phl] (notwait : Bool, chkpt : Bool,to : adress,q : pqueue,qdls : pqueue,qdlr : pacqueue)
                                                                    : exit(Nat,frmrreason,pqueue,pqueue,pacqueue,Bit) :=
   [not(chkpt) and not(notwait)] ->
       (phl ? f : eframe[(((correct(f) and (not(is_sabm(get_f(f))))) and (not(is_sabme(get_f(f))))) and
                          (not(is_disc(get_f(f)))))    and ((eq(adrs(f),to)) and (pf(f) eq 1 of Bit))]
```

```
                ;exit((Succ(Succ(Succ(0))) of Nat),make_reason(ctrl(f),0 of Bit,0 of Nat,
                     0 of Bit,0 of Nat,0 of Bit,0 of Bit, 0 of Bit,0 of Bit,0 of Bit),q,qdls,qdlr,1 of Bit)
            )
    endproc  (* receiveunsolicit *)
  process receivesabm[phl](q : pqueue,qdls : pqueue,qdlr : pacqueue) :
                                                                exit(Nat,frmrreason,pqueue,pqueue,pacqueue,Bit) :=
        phl ? f : eframe[correct(f) and (is_sabm(get_f(f)) or is_sabme(get_f(f)))]
        ;exit((Succ(Succ(0)) * Succ(Succ(0)) of Nat),make_reason(ctrl(f),0 of Bit,0 of Nat,
                0 of Bit,0 of Nat,0 of Bit,0 of Bit, 0 of Bit,0 of Bit,0 of Bit),q,qdls,qdlr,pf(f))
  endproc  (* receivesabm *)

  process receiveuaordm[phl] (q : pqueue,qdls : pqueue,qdlr : pacqueue) :
                                                                exit(Nat,frmrreason,pqueue,pqueue,pacqueue,Bit) :=
      phl ? f : eframe[correct(f) and (is_ua(get_f(f)) or is_dm(get_f(f)))]
      ;exit((Succ(Succ(Succ(0))) of Nat),make_reason(ctrl(f),0 of Bit,0 of Nat,
              0 of Bit,0 of Nat,0 of Bit,0 of Bit, 0 of Bit,0 of Bit,0 of Bit),q,qdls,qdlr,1 of Bit)
  endproc (* receiveuaordm *)

   process receivefrmr[phl](from : adress,to : adress,n2 : Nat ,k : Nat,tmode : mmode,
                              vs : Nat,vr : Nat,lu : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,rbusy : Bool) :
                                                              exit(Nat,frmrreason,pqueue,pqueue,pacqueue,Bit) :=
       phl ? f : eframe[correct(f) and is_frmr(get_f(f))]
       ;exit((Succ(Succ(0)) of Nat),make_reason(ctrl(f),0 of Bit,0 of Nat,
              0 of Bit,0 of Nat,0 of Bit,0 of Bit, 0 of Bit,0 of Bit,0 of Bit),q,qdls,qdlr,1 of Bit)
    endproc (* receivefrmr *)
 endproc (* rcvtoreset *)

  process deliver[dl]  (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                          reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                             exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,
                                  Bool,Nat,frmrreason,Nat,Bit) :=
     [not(isempty(qdlr))] ->
      (dl ! dl_data_ind(firstone(qdlr))
       ;exit(vs,vr,lu,vsp,q,qdls,removefirst(qdlr),starttime,rbusy,lbusy,chkpt,retrans,reject,init,
             notwait,0 of Nat,nors,0 of Nat,0 of Bit)
      )
    endproc
  process retransmission[phl] (vs,vr,lu,vsp : Nat,q,intq,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                                retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                              exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,
                                   Bool,Nat,frmrreason,Nat,Bit) :=
     [not(rbusy)] ->
      ( ([not(isempty(intq))] ->
         (phl ! m(make_iframe(flagging,to,ctl_iframe(vs,0,(vr mod k)),packe(firstone(intq)),fc,flagging),
                 infor,correct,notless,noabort)
       ;retransmission[phl] ((vs + Succ(0) of Nat),vr,lu,vsp,q,removefirst(intq),qdls,qdlr,starttime,
                             rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
        )
       []
       [isempty(intq)] ->
        (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,Succ(0) of Nat,
              nors,0 of Nat,0 of Bit))
     [> (receiverej[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,to,from,tmode)
         >>
       accept vs : Nat,vr : Nat,lu : Nat,vsp : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,starttime : Bool,
              rbusy : Bool,lbusy : Bool,chkpt : Bool,retrans : Bool,reject : Bool,init : Nat,notwait : Bool,typ : Nat,
              rs : frmrreason,ind : Nat,b : Bit in
          (phl ! m(make_iframe(flagging,to,ctl_iframe(snn(firstone(intq)),0,(vr mod k)),packe(firstone(intq)),
                               fc,flagging),infor,correct,notless,abort)
```

```
              ;retransmission[phl] (vs,vr,lu,vsp,q,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,
                                       init,k,to,from,tmode)
             )
           )))
     []
     [rbusy] ->
       (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,Succ(0) of Nat,
              nors,0 of Nat,0 of Bit))
    endproc
     process  recvpack[dl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                                   retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                                  exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,Bool,
                                       Nat,frmrreason,Nat,Bit) :=
         [((Succ(vsp) mod k) ne lu)] ->
           (dl ? p : primitive [is_dl_data_req(p)]
            ;([rbusy and not(starttime)] ->
              (exit(vs,vr,lu,((vsp + Succ(0) mod k) ),q,((mp(get_pack(p),vsp)) +-- qdls) of pqueue,qdlr,
                                 true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,(Succ(0) of Nat),nors,0 of Nat,0 of Bit))
             []
             [not(rbusy) or starttime] ->
               (exit(vs,vr,lu,((vsp + Succ(0) mod k) ),q,((mp(get_pack(p),vsp)) +-- qdls) of pqueue,qdlr,
                        starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,0 of Nat,0 of Bit))
            ))
       endproc
      process sendi[phl,t] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                                 reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode)
                               : exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,Bool,
                                     Nat,frmrreason,Nat,Bit) :=
         ((( (  ( i;phl ! m(make_iframe(flagging,to,ctl_iframe(vs,0,(vr mod  k)),packe(firstone(qdls)),fc,flagging),
                               noinfor,correct,notless,noabort)  [not(isempty(qdls)) and not(rbusy)]
                  ;exit(snn(firstone(qdls)))
                 >> accept sn  : Nat in
               ([not(starttime)] ->
                  (t! start
                   ;exit(((vs + Succ(0) of Nat) mod k),vr,lu,vsp,(firstone(qdls) +-- q),removefirst(qdls)
                          of pqueue,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,0 of Nat,0 of Bit)
                 )
                []
                [starttime] ->
                   exit(((vs + Succ(0) of Nat) mod k),vr,lu,vsp,(firstone(qdls) +-- q),removefirst(qdls)
                          of pqueue,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,0 of Nat,0 of Bit)
               )
          )
        )
        []
      (
       (i;phl ! m(make_iframe(flagging,to,ctl_iframe(vs,1,(vr mod k)),packe(firstone(qdls)),fc,flagging),
                     infor,correct,notless,noabort) [not(isempty(qdls)) and not(rbusy)]
          ;exit(snn(firstone(qdls)))
          >> accept sn : Nat in
             ([not(starttime)] ->
               (t! start
                ;exit(((vs + Succ(0) of Nat) mod k),vr,lu,vsp,(firstone(qdls) +-- q),removefirst(qdls) of pqueue,
                       qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,0 of Nat,0 of Bit)
                )
              []
```

```
            [starttime] ->
              exit(((vs + Succ(0) of Nat) mod k),vr,lu,vsp,(firstone(qdls) +-- q),removefirst(qdls) of pqueue,
                     qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,0 of Nat,0 of Bit)
          )
         )
       ))
         [> (receiverej[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                k,to,from,tmode)
          >> accept
          vs : Nat,vr : Nat,lu : Nat,vsp : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,starttime : Bool,rbusy : Bool,
           lbusy : Bool,chkpt : Bool,retrans : Bool,reject : Bool,init : Nat,notwait : Bool,typ : Nat,rs : frmrreason,
           ind : Nat,b : Bit in
         (phl ! m(make_iframe(flagging,to,ctl_iframe(vs,1,(vr mod k)),packe(firstone(qdls)),fc,flagging),
                  infor,correct,notless,abort)
           ;retransmission[phl](vs,vr,lu,vsp,q,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,
            to,from,tmode)
          )
           )
          )
        [> (receivernr[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                 k,to,from,tmode))
           )
     endproc (* sendei *)

      process releasequeue[phl](k : Nat,nr : Nat,q : pqueue) : exit(Nat,pqueue) :=
           [isempty(q)]  -> exit(nr,q)
           []
           [not(isempty(q))] ->
              ([(nr ne (snn(firstone(q))))] ->
                     i
                     ;releasequeue[phl](k,nr,removefirst(q))
              )
              []
              [nr eq (snn(firstone(q)))] ->
                        exit(nr,q)
      endproc (* releasequeue *)

       process validnrf[phl](vs : Nat,lu : Nat,k : Nat,q : pqueue) : exit(Nat,pacqueue,eframe,pqueue) :=
            phl ? f : eframe[(((nr(f) ge lu) or (nr(f) le (vs mod k))) and ((vs mod k) lt lu)) or (((nr(f) ge lu)
                                and (nr(f) le (vs mod k))) and ((vs mod k) ge lu))]
            ;exit(nr(f),any pacqueue,any eframe,q)
      endproc

  process  receivevalidi[phl,t] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                                     retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                                exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,Bool,
                                     Nat,frmrreason,Nat,Bit) :=
       [not(lbusy)] ->
          (((( phl ? f : eframe[((pf(f) eq 0 of Bit)   and (correct(f) and (is_i(get_f(f)) and (ns(f) eq (vr mod k) of Nat))))]
                ;exit(nr(f),qdlr,f,q) (* no poll requested *)
            |[phl]|
              validnrf[phl](vs,lu,k,q)
            ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
                  exit(nrcv,qdlr,f,q,false,false)
          )
         []
        (( phl ? f : eframe[((pf(f) eq 1 of Bit) and (((correct(f)) and ((is_i(get_f(f))) and (ns(f) eq (vr mod k) of Nat)))))]
             ;exit(nr(f),qdlr,f,q)
        |[phl]|
```

```
           validnrf[phl](vs,lu,k,q)
          ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
             ((phl ! m(make_rr(flagging,from,ctl_sframe(1,((vr + Succ(0)) mod k)),fc,flagging),
                          noinfor,correct,notless,noabort)
              ;exit(nrcv,qdlr,f,q,true,false)
             )
            [] (* local entity becomes busy *)
             (phl ! m(make_rnr(flagging,from,ctl_sframe(1,((vr + Succ(0)) mod k)),fc,flagging),
                         noinfor,correct,notless,noabort)
              ;exit(nrcv,qdlr,f,q,true,true)
             ))
            ))
           >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue,chk : Bool,lbusy : Bool in
            ([not(isempty(q)) and (nrcv ne lu)] ->
                       (* receive nr higher than last received nr acking some i frames *)
              ( t ! reset              (* stop timer *)
               ;releasequeue[phl] (k,nrcv,q)
               >> accept lu : Nat, q : pqueue in
                  exit(lu,vr + Succ(0),q,false,pack(f) +-- qdlr)
                 >> accept lu : Nat,vr : Nat,q : pqueue,starttime : Bool,qdlr : pacqueue in
                  ([isempty(q)] ->  (* if no outstanding frames unacked do not start timer again *)
                    ((([not(isempty(qdls))] ->
                       ([not(chk)] ->
                        (i;sendi[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                       k,to,from,tmode)
                        []
                        i;sendrr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                     k,to,from,tmode)
                        []
                        i;sendrnr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                      k,to,from,tmode)
                        )
                       []
                       [chk] ->
                        (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,
                              nors,0 of Nat,0 of Bit))
                     )
                      []
                      [isempty(qdls)] ->  (* dte/dce becomes busy *)
                       ([not(chk)] ->
                         (i;sendrnr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                        k,to,from,tmode)
                         []
                         i;sendrr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                      k,to,from,tmode)
                        )
                      []
                      [chk] ->
                       (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,
                             nors,0 of Nat,0 of Bit))
                     )
                    )
                       )
                    >> accept vs : Nat,vr : Nat,lu : Nat,vsp : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,starttime : Bool,
                              rbusy : Bool,lbusy : Bool,chkpt : Bool,retrans : Bool,reject : Bool,init : Nat,notwait : Bool,
                              typ : Nat,rs : frmrreason,a : Nat,b : Bit in
                (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,typ,nors,0 of Nat,0 of Bit))
                  )
                 []
```

```
          [not(isempty(q))]   -> (* stiill unacked frames start timer again  *)
           (( ((([not(isempty(qdls))] ->
              ([not(chk)] ->
               (i;sendi[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,k,
                               to,from,tmode)
               []
               i;sendrr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                             k,to,from,tmode)
               []
               i;sendrnr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                               k,to,from,tmode)
              )
              []
              [chk] ->
               (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,(Succ(0)
                    of Nat),nors,0 of Nat,0 of Bit))
             )
             []
             [isempty(qdls)] ->
              ([not(chk)] ->
               (i;sendrnr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                               k,to,from,tmode)
               []
               i;sendrr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                             k,to,from,tmode)
               )
               []
              [chk] ->
               (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,(Succ(0)
                    of Nat),nors,0 of Nat,0 of Bit))
              )
             )
            )
          )
          )
          >> accept vs : Nat,vr : Nat,lu : Nat,vsp : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,starttime : Bool,
                   rbusy : Bool,lbusy : Bool,chkpt : Bool,retrans : Bool,reject : Bool,init : Nat,notwait : Bool,
                   typ : Nat,rs : frmrreason,ind : Nat,b : Bit in
       (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,typ,nors,0 of Nat,0 of Bit))
        )
       ))
     []
     [(isempty(q) or (nrcv eq lu))] ->
            (* nr received equal to last nr received *)
        (exit(lu,vr + Succ(0),q,true,pack(f) +-- qdlr)
       >> accept lu : Nat,vr : Nat,q : pqueue,starttime : Bool,qdlr : pacqueue in
         ((( ([not(isempty(qdls))] ->
             ([not(chk)] ->
              (i;sendi[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                              k,to,from,tmode)
              []
              i;sendrr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                            k,to,from,tmode)
              []
              i;sendrnr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                              k,to,from,tmode)
              )
             []
             [chk] ->
```

```
                    (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,
                          0 of Nat,0 of Bit))
                   )
                  []
                  [isempty(qdls)] ->  (* dte/dce becomes busy or send rr since no info frame available *)
                   ([not(chk)] ->
                    (i;sendrnr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                   k,to,from,tmode)
                     []
                     i;sendrr[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                  k,to,from,tmode)
                    )
                    []
                    [chk] ->
                    (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,
                          0 of Nat,0 of Bit))
                   )
               )
                ) )
                 >> accept
              vs : Nat,vr : Nat,lu : Nat,vsp : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,starttime : Bool,rbusy : Bool,
               lbusy : Bool,chkpt : Bool,retrans : Bool,reject : Bool,init : Nat,notwait : Bool,typ : Nat,rs : frmrreason,
                ind : Nat,b : Bit  in
        (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,typ,nors,0 of Nat,0 of Bit))
         ))
        ))
   endproc (* receivevalidi  *)

   process rcviandbusy[phl,t] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                                  retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                          exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,Nat,Bool,
                               Nat,frmrreason,Nat,Bit) :=
     [lbusy] ->
       ((phl ? f : eframe[(correct(f) and is_i(get_f(f))) and  ((pf(f) ne 1 of Bit) and (ns(f) eq (vr mod k)))]
         ;exit(nr(f),qdlr,f,q)
      |[phl]|
       validnrf[phl](vs,lu,k,q)
      )
     []
     (( phl ? f : eframe[(correct(f) and is_i(get_f(f))) and  ((pf(f) eq 1 of Bit) and (not(ns(f) eq (vr mod k))))]
        ;exit(nr(f),qdlr,f,q)
       |[phl]|
        validnrf[phl](vs,lu,k,q)
           ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
              phl ! m(make_rnr(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
              ;exit(nr(f),qdlr,f,q)
     )
   []
   (( phl ? f : eframe[(correct(f) and is_i(get_f(f))) and  ((pf(f) eq 1 of Bit) and (ns(f) eq (vr mod k)))]
      ;exit(nr(f),qdlr,f,q)
    |[phl]|
      validnrf[phl](vs,lu,k,q)
     ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
          phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          ;exit(nr(f),qdlr,f,q)
       ))
       >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
     (   [not(isempty(q)) and (nrcv ne lu)] ->
            (* receive nr higher than last received nr acking some i frames *)
```

```
        ( t ! reset              (* stop timer *)
         ;releasequeue[phl] (k,nrcv,q)
         >> accept lu : Nat, q : pqueue in
            ([isempty(q)] ->
              (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,
                     nors,0 of Nat,0 of Bit))
              )
             []
             [not(isempty(q))] ->
              (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,(Succ(0) of Nat),
                     nors,0 of Nat,0 of Bit))
            )
      []
      [isempty(q) or (nrcv eq lu)] ->
               (* nr received equal to last nr received *)
       (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,
              0 of Nat,0 of Bit))
  )
  endproc      (* rcviandbusy *)

process receiverr[phl,t] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                          retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                  exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                       Nat,Bool,Nat,frmrreason,Nat,Bit)  :=
      ((((phl ? f : eframe[is_rr(get_f(f)) and ((pf(f) eq 0 of Bit))]
          ;exit(nr(f),qdlr,f,q)             (* no poll requested from peer entity *)
        |[phl]|
          validnrf[phl](vs,lu,k,q)
      ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
          exit(nrcv,qdlr,f,q,false,chkpt,init)
      )
     []
      ( (phl ? f : eframe[chkpt and (((pf(f) eq 1 of Bit) and (eq(adrs(f),to))) and (correct(f) and (is_rr(get_f(f)))))]
          ;exit(nr(f),qdlr,f,q) (* no poll requested *)
       |[phl]|
         validnrf[phl](vs,lu,k,q)
       ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
         exit(nrcv,qdlr,f,q,true,false,0 of Nat)
     )
    []
    ((phl ? f : eframe[is_rr(get_f(f)) and ((pf(f) eq 1 of Bit) and (eq(adrs(f),from)))]
      ;exit(nr(f),qdlr,f,q)
    |[phl]|
      validnrf[phl](vs,lu,k,q)
    ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe ,q : pqueue in
  ([not(lbusy)] ->       (* local entitty not busy *)
     ([not(reject)] ->
        (  iphl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          ;exit(nrcv,qdlr,f,q,false,chkpt,init)          (* poll requested so rr with f=1 sent *)
        )
      []
      [reject] ->
      ((  i;phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
         ;exit(nrcv,qdlr,f,q,false,chkpt,init)
       )
       []
       (  i;phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
         ;exit(nrcv,qdlr,f,q,false,chkpt,init)
      )
```

```
      ))
       []
       [lbusy] ->          (* local entity busy send rnrn *)
            phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
           ;exit(nrcv,qdlr,f,q,false,chkpt,init)
      )
     ))
    >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue,retranschk : Bool,chk : Bool,init : Nat in
     ([(not(isempty(q)) and (nrcv ne lu)) and not(retranschk)] ->
              (* receive nr higher than last received nr acking some i frames *)
         ( t ! reset            (* stop timer *)
          ;releasequeue[phl] (k,nrcv,q)
          >> accept lu : Nat, q : pqueue in
             exit(lu,false,q,qdls,qdlr,retranschk,chk)
         )
      []
      [(isempty(q) or (nrcv eq lu)) or retranschk] ->
                  (* nr received equal to last nr received *)
           exit(lu,true,q,qdls,qdlr,retranschk,chk)
      ) >> accept lu : Nat,fin : Bool,q : pqueue,qdls : pqueue,qdlr : pacqueue,retranschk : Bool,chkpt : Bool in
      ([(not(retrans) and not(retranschk)) and fin] ->
           (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,
                  0 of Nat,0 of Bit))
       []
       [(not(retrans) and not(retranschk)) and (not(fin) and isempty(q))] ->
            (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,
                  0 of Nat,0 of Bit))
       []
       [(not(retrans) and not(retranschk)) and (not(fin) and not(isempty(q)))]  ->
            (exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
                 (Succ(0) of Nat),nors,0 of Nat,0 of Bit))
       []
       [retrans or retranschk] ->
         ([fin or retranschk] ->
             (hide t in
               retransmit[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                    k,to,from,tmode)
             )
          []
          [not(fin) and not(retranschk)] ->
             (hide t in
             (t ! start
              ;retransmit[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                    k,to,from,tmode)
             ))
         )
      ))
 endproc  (* receiverr  *)

 process retransmit[phl,t] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                              reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                          exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                               Nat,Bool,Nat,frmrreason,Nat,Bit) :=
      retransmission[phl] (vs,vr,lu,vsp,q,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                             k,to,from,tmode)
      [> (receivernr[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                             k,to,from,tmode))

endproc (* retransmit *)
```

```
process sendrr[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                                reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
            exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                               Nat,Bool,Nat,frmrreason,Nat,Bit) :=
        [not(rbusy) and not(reject)] ->
         (((i;phl ! m(make_rr(flagging,to,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
             ;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,false,chkpt,retrans,reject,init,notwait,
                     0 of Nat,nors,0 of Nat,0 of Bit)
         )
        []
        (i;phl ! m(make_rr(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
         ;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,false,chkpt,retrans,reject,init,notwait,
                 0 of Nat,nors,0 of Nat,0 of Bit)
       )
      []
        (i;phl ! m(make_rr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
         ;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,false,true,retrans,reject,init,notwait,
                 0 of Nat,nors,0 of Nat,0 of Bit)
      ))
      []
        i;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
                 0 of Nat,nors,0 of Nat,0 of Bit))
endproc
process sendrej[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                                 retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                              exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                                   Nat,Bool,Nat,frmrreason,Nat,Bit) :=
    ([not(lbusy) and not(reject)] ->
         (((( phl ? f : eframe[(correct(f) and is_i(get_f(f))) and (((ns(f) ne (vr mod k))) and not((pf(f) eq 1 of Bit) and
                                  (eq(adrs(f),from))))]
             ;exit(nr(f),qdlr,f,q)
       |[phl]|
           validnrf[phl](vs,lu,k,q)
      )
    ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
       releasequeue[phl](k,nrcv,q)
       >> accept lu : Nat,q : pqueue in
       ((i;phl ! m(make_rej(flagging,to,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
        ;exit(lu,vr,q,qdls,qdlr,false)
       )
      []
       (i;phl ! m(make_rej(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
       ;exit(lu,vr,q,qdls,qdlr,false))
      []
      (i;phl ! m(make_rej(flagging,to,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
       ;exit(lu,vr,q,qdls,qdlr,false))
      []
       (i;phl ! m(make_rej(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
        ;exit(lu,vr,q,qdls,qdlr,true))
      ))
     []
      (phl ? f : eframe[(correct(f) and is_i(get_f(f))) and (((ns(f) ne (vr mod k))) and
                              ((pf(f) eq 1 of Bit)))]
      ;phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct, notless,noabort)
      ;exit(lu,vr,q,qdls,qdlr,false))
     )
    []
      [reject] ->
```

```
        (((( phl ? f : eframe[((correct(f) and is_i(get_f(f))) and ((ns(f) ne (vr mod k)))) and (pf(f) eq 0 of Bit)]
              ;exit(nr(f),qdlr,f,q)
     |[phl]|
         validnrf[phl](vs,lu,k,q)
    )
     >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
     releasequeue[phl](k,nrcv,q)
        )
        []
        ((phl ? f : eframe[((correct(f) and is_i(get_f(f))) and ((ns(f) ne (vr mod k)))) and (pf(f) eq 1 of Bit)]
          ;exit(nr(f),qdlr,f,q)
     |[phl]|
     validnrf[phl](vs,lu,k,q)
        )
      >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
        phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
        ;releasequeue[phl](k,nrcv,q)
       ))
     >> accept lu : Nat,q : pqueue in
        (exit(lu,vr,q,qdls,qdlr,chkpt)))
        )
  >> accept lu : Nat,vr : Nat,q : pqueue,qdls : pqueue,qdlr : pacqueue,chkpt : Bool in
       ([isempty(q)]  ->
           (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,0 of Nat,nors,
                      0 of Nat,0 of Bit))
        []
        [not(isempty(q))] ->
           exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,(Succ(0) of Nat),
                   nors,0 of Nat,0 of Bit))
endproc (* sendrej *)
process receiverej[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                                       reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                                      exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                                              Nat,Bool,Nat,frmrreason,Nat,Bit) :=
     ((((phl ? f : eframe[is_rej(get_f(f)) and ((pf(f) eq 0 of Bit))]
           ;exit(nr(f),qdlr,f,q)           (* no poll requested from peer entity *)
        |[phl]|
           validnrf[phl](vs,lu,k,q)
        ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
         exit(nrcv,qdlr,f,q,false,chkpt,init)
    )
    []
     ( (phl ? f : eframe[chkpt and (((pf(f) eq 1 of Bit) and (eq(adrs(f),to))) and (correct(f) and (is_rej(get_f(f)))))]
         ;exit(nr(f),qdlr,f,q) (* no poll requested *)
      |[phl]|
        validnrf[phl](vs,lu,k,q)
      ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
           exit(nrcv,qdlr,f,q,true,false,0 of Nat)
      )
    []
      ((phl ? f : eframe[is_rej(get_f(f)) and ((pf(f) eq 1 of Bit) and (eq(adrs(f),from)))]
        ;exit(nr(f),qdlr,f,q)
     |[phl]|
     validnrf[phl](vs,lu,k,q)
      ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe ,q : pqueue in
       ([not(lbusy)] ->      (* local entitty not busy *)
          ([not(reject)] ->
             ( phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
              ;exit(nrcv,qdlr,f,q,false,chkpt,init)         (* poll requested so rr with f=1 sent *)
```

```
              )
             []
             [reject] ->
              (( phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
              ;exit(nrcv,qdlr,f,q,false,chkpt,init)          (* poll requested so rr with f=1 sent *)
               )
               []
               ( phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
              ;exit(nrcv,qdlr,f,q,false,chkpt,init)
               )
             )
           )
          []
          [lbusy] ->             (* local entity busy send rnrn *)
                phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
               ;exit(nrcv,qdlr,f,q,false,chkpt,init)
          )
        ))
       >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue,retranschk : Bool,chk : Bool,init : Nat in
          releasequeue[phl](k,nrcv,q)
           >> accept lu : Nat,q : pqueue in
             (hide t in
              retransmit[phl,t] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                                       k,to,from,tmode)
          ))
      endproc
      process sendrnr[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                                    reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                               exit (Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                                     Nat,Bool,Nat,frmrreason,Nat,Bit) :=
          [(not(rbusy)) and not(reject)] ->
              (((i;phl ! m(make_rnr(flagging,to,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
                 ;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,true,chkpt,retrans,reject,init,notwait,
                        0 of Nat,nors,0 of Nat,0 of Bit)
              )
          []
          (i;phl ! m(make_rnr(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
            ;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,true,chkpt,retrans,reject,init,notwait,
                   0 of Nat,nors,0 of Nat,0 of Bit)
         )
        []
          i;phl ! m(make_rnr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          ;exit(vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,true,chkpt,retrans,reject,init,notwait,
                 0 of Nat,nors,0 of Nat,0 of Bit)
          ))
      endproc

      process receivernr[phl,t] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                                        retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                                   exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                                        Nat,Bool,Nat,frmrreason,Nat,Bit)  :=
          (((( (phl ? f : eframe[((pf(f) eq 0 of Bit) and (correct(f)) and (is_rnr(get_f(f))))]
                ;exit(nr(f),qdlr,f,q) (* no poll requested *)
           |[phl]|
               validnrf[phl](vs,lu,k,q)
             ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue   in
                exit(nrcv,qdlr,f,q,false,chkpt,init)
          )
        []
```

```
        ( (phl ? f : eframe[(chkpt and (((pf(f) eq 1 of Bit) and (eq(adrs(f),to))) and (correct(f) and (is_rnr(get_f(f))))))]
          ;exit(nr(f),qdlr,f,q) (* no poll requested *)
        |[phl]|
          validnrf[phl](vs,lu,k,q)
        ) >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
            exit(nrcv,qdlr,f,q,true,false,0 of Nat)
        )
       []
       (( (phl ? f : eframe[(((pf(f) eq 1 of Bit) and (eq(adrs(f),from))) and (((correct(f)) and ((is_rnr(get_f(f)))
      ))))]
          ;exit(nr(f),qdlr,f,q)
        |[phl]|
        validnrf[phl](vs,lu,k,q)
        )
        >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue in
          ([not(lbusy)] ->
            ([not(reject)] ->
              ( phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
                ;exit(nrcv,qdlr,f,q,false,chkpt,init)
              )
             []
             [reject] ->
              ((i;phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
                 ;exit(nrcv,qdlr,f,q,false,chkpt,init)
              )
              []
              (i;phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
                ;exit(nrcv,qdlr,f,q,false,chkpt,init)
              )
             )  )
           []
           [(lbusy)] ->
              phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
              ;exit(nrcv,qdlr,f,q,false,chkpt,init)
           ))
         )
        ))  >> accept nrcv : Nat,qdlr : pacqueue,f : eframe,q : pqueue,retranschk : Bool,chk : Bool,init : Nat in
        ([not(isempty(q)) and (nrcv ne lu)] ->
           (t ! reset
            ;releasequeue[phl](k,nrcv,q)
             >> accept lu : Nat,q : pqueue in
                exit(lu,q,qdls,true,chk)  (* timer stopped *)
          )
        []
        [isempty(q) or (nrcv eq lu)] ->
          exit(lu,q,qdls,false,chk)    (* timer not stopped *)
       )
       )
       >> accept lu : Nat,q : pqueue,qdls : pqueue,stoptimer : Bool,chkpt : Bool in
         ([stoptimer and (isempty(qdls) and isempty(q))] ->
           (exit(vs,vr,lu,vsp,q,qdls,qdlr,false,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
                  (Succ(Succ(0)) of Nat),nors,0 of Nat,0 of Bit))
          []
          [stoptimer and (not(isempty(qdls)) or not(isempty(q)))] ->
           (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
                  (Succ(Succ(Succ(0))) of Nat),nors,0 of Nat,0 of Bit))
          []
          [not(stoptimer) and (isempty(qdls) and isempty(q))] ->
           (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
```

```
                0 of Nat,nors,0 of Nat,0 of Bit))
        []
        [not(stoptimer) and (not(isempty(qdls)) or not(isempty(q)))] ->
          (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
               0 of Nat,nors,0 of Nat,0 of Bit))
        )
    endproc
   process  timeout[phl,t] (n2,vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                               reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                          exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                               Nat,Bool,Nat,frmrreason,Nat,Bit) :=
      ([not(reject)] ->
       (t ! expired
        ;([init ne n2] ->
          (i;sendrrb[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                             k,to,from,tmode)
           []
           i;sendrnrb[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                             k,to,from,tmode)
          )
        []
        [init eq n2] ->
          (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
                Succ(Succ(Succ(Succ(0))) of Nat),make_reason(ctl_uframe(1),0 of Bit,0 of Nat,
                0 of Bit,0 of Nat,0 of Bit,0 of Bit, 0 of Bit,0 of Bit,0 of Bit),0 of Nat,1 of Bit ))
      ))
     []
     [reject] ->
       (t ! expired
       ;([init ne n2] ->
           (sendrejb[phl] (vs,vr,lu,vsp,q,qdls,qdlr,starttime,rbusy,lbusy,chkpt,retrans,reject,notwait,init,
                             k,to,from,tmode))
        []
        [init eq n2] ->
          (exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,lbusy,chkpt,retrans,reject,init,notwait,
                (Succ(Succ(Succ(Succ(0)))) of Nat),make_reason(ctl_uframe(1),0 of Bit,0 of Nat,
                0 of Bit,0 of Nat,0 of Bit,0 of Bit, 0 of Bit,0 of Bit,0 of Bit),0 of Nat,1 of Bit))
       ))
        )
     where
      process sendrrb[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                               reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                          exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                               Nat,Bool,Nat,frmrreason,Nat,Bit) :=
             phl ! m(make_rr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
             ;(exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,false,true,retrans,reject,init,notwait,
                  (Succ(Succ(0)) of Nat),nors,0 of Nat,0 of Bit))
      endproc  (* sendrrb *)

     process sendrnrb[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,
                               retrans,reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
                        exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                             Nat,Bool,Nat,frmrreason,Nat,Bit) :=
          phl ! m(make_rnr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          ;(exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,true,true,retrans,reject,init,notwait,(Succ(Succ(0)) of Nat),
                  nors,0 of Nat,0 of Bit))
      endproc  (* sendrrb *)
      process sendrejb[phl] (vs,vr,lu,vsp : Nat,q,qdls : pqueue,qdlr : pacqueue,starttime,rbusy,lbusy,chkpt,retrans,
                               reject,notwait : Bool,init,k : Nat,to,from : adress,tmode : mmode) :
```

```
                                        exit(Nat,Nat,Nat,Nat,pqueue,pqueue,pacqueue,Bool,Bool,Bool,Bool,Bool,Bool,
                                            Nat,Bool,Nat,frmrreason,Nat,Bit) :=
                phl ! m(make_rej(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
              ;(exit(vs,vr,lu,vsp,q,qdls,qdlr,true,rbusy,false,true,retrans,true,init,notwait,(Succ(Succ(0)) of Nat),
             nors,0 of Nat,0 of Bit))
        endproc  (* sendrejb *)
      endproc (*  timeoutbusy  *)
      endproc  (* infphase *)
      endproc (* inforphase *)
     endproc (* dataphase *)
    endproc (* lapb *)
     endspec
```

Appendix B: Trees Used in the Test Case Generation

Disconnected Tree

```
bh0  * 1 phl ?[sframe,sframe]eframe      (1)
          [or(and(correct(sframe),and(eq(pf(sframe),$1),eq(adrs(sframe),from))),is_disc(get_f(sframe)))]
          [not(or(and(eq(tmode,basic),is_sabm(get_f(sframe))),and(eq(tmode,extended),is_sabme(get_f(sframe)))))]  [619,622]
bh1  * | 1 i (enable: exit !pf(sframe):Bit) [620,623]
bh2  * | | 1 phl !m(make_dm(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort):eframe [624]
bh3  * | | | 1 i (enable: exit) [626] ==> again bh0
       * 2 phl ?[sframe,sframe]eframe      (2)
          [and(not(or(and(correct(sframe),and(eq(pf(sframe),$1),eq(adrs(sframe),from))),is_disc(get_f(sframe)))),
          not(and(is_dm(get_f(sframe)),eq(pf(sframe),$0))))]
          [not(or(and(eq(tmode,basic),is_sabm(get_f(sframe))),and(eq(tmode,extended),is_sabme(get_f(sframe)))))]  [629,633]
bh4  * | 1 i (enable: exit) [631,634] ==> again bh0
       * 3 dl ?p:primitive                 (3)
         [is_dl_est_req(p)]  [639]
bh5  * | 1 [eq(tmode,basic)] i (hiding: [lt(init,n2)] p !$1:Nat) [1096,1150]
bh6  * | | 1 phl !f:eframe [1097,1151]
bh7  * | | | 1 i (hiding: t !start:time) [1078,547] ==> continue
       * | 2 [eq(tmode,basic)] i (hiding: [eq(init,n2)] p !$0:Bit) [1108,1153]
bh8  * | | 1 exit !false:Bool !false:Bool ** EXIT SUCCEED ** [1109,1154]
       * | 3 [eq(tmode,extended)] i (hiding: [lt(init,n2)] p !$1:Nat) [1096,1150]
bh9   * | | 1 phl !f:eframe [1097,1151]
bh10* | | | 1 i (hiding: t !start:time) [1078,547] ==> continue
       * | 4 [eq(tmode,extended)] i (hiding: [eq(init,n2)] p !$0:Bit) [1108,1153] ==> again bh8
       * | 5 i (hiding: [lt(init,n2)] p !$1:Nat) [1096,1150]
bh11* | | 1 phl !f:eframe [1097,1151]
bh12* | | | 1 i (hiding: t !start:time) [1078,547] ==> continue
       * | 6 i (hiding: [eq(init,n2)] p !$0:Bit) [1128,1166] ==> again bh8
       * | 7 [senddmoption] i (hiding: [lt(init,n2)] p !$1:Nat) [682,719]
bh13* | | 1 phl !m(make_dm(flagging,to,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort):eframe [683,720]
bh14* | | | 1 i (hiding: t !start:time) [684,536] ==> continue
       * | 8 i (hiding: [eq(init,n2)] p !$0:Bit) [695,722] ==> again bh8
       * 4 phl ?sframe:eframe              (4)
        [and(or(and(is_sabm(get_f(sframe)),eq(tmode,basic)),and(is_sabme(get_f(sframe)),eq(tmode,extended))),
        correct(sframe))]  [596]
bh15* | 1 i [598]
        * | | 1 phl !m(make_ua(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort):eframe [598]
bh16* | | | 1 exit !true:Bool !false:Bool ** EXIT SUCCEED ** [598]
       * | 2 i [602]
       * | | 1 phl !m(make_dm(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort):eframe [602]
bh17* | | | 1 exit !false:Bool !false:Bool ** EXIT SUCCEED ** [603]
       * 5 phl ?sframe:eframe              (5)
          [and(correct(sframe),and(is_dm(get_f(sframe)),eq(pf(sframe),$0)))]  [607]
       * | 1 i [609]
bh18* | | 1 i (hiding: [lt(init,n2)] p !$1:Nat) [1096,1150]
bh19* | | | 1 phl !f:eframe [1097,1151] ==> again bh7
       * | | 1 i (hiding: [eq(init,n2)] p !$0:Bit) [1128,1166] ==> again bh8
       * | 2 i [613]
       * | | 1 phl !m(make_disc(flagging,to,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort):
            eframe [613] ==> again bh17
       * 6 phl ?f:eframe                   (6)
        [not(correct(f))]  [830] ==> again bh3
```

Linksetup Tree

```
bh0  * 1 i (hiding: [lt(init,n2)] p !Succ(0):Nat) [1096,1150]
bh1  * | 1 phl !m(make_sabm(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort):eframe [1097,1151]
bh2  * | | 1 i (hiding: t !start:time) [1098,547]
bh3  * | | | 1 i (specified explicitly) [548]                              (1)
bh4  * | | | | 1 i (hiding: t !expired:time) [1100,549]
bh5 * | | | | | 1 i (enable: exit) [1100]
bh6  * | | | | | | 1 i (hiding: [lt(init,n2)] p !Succ(0):Nat) [1096,906,931,956,992,1027]
bh7  * | | | | | | | 1 phl !m(make_sabm(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort):
                    eframe [1097,907,932,957,993,1028] ==> again bh2
       * | | | | | | 2 i (hiding: [eq(init,n2)] p !0:Bit) [1108,1153]
bh8  * | | | | | | | 1 exit !false:Bool !false:Bool  !$f1  ** EXIT SUCCEED ** [1109,1154,550]
bh9  * | | | 2 phl ?[fram,sframe,f,f,f,f]eframe                          (2)
         [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
              eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
         [and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1)))]
         [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]
         [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
         [not(and(correct(f),is-disc(get-f(f))))]  [1127,882,921,938,979,1000]
        * | | | | | 1 i (hiding: pp !0:Nat) [1113,890,922,941,983,1003]
bh10* | | | | | | 1 i (hiding: t !reset:time) [1114,552]
bh11* | | | | | | | 1 exit !true:Bool !false:Bool !$f1:eframe  ** EXIT SUCCEED ** [1115,891,917,942,984,1004,553]
       * | | | 3 phl ?[fram,sframe,f,f,f,f]eframe                         (3)
            [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
             eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
            [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
            [and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1)))]
            [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
            [not(and(correct(f),is-disc(get-f(f))))]  [1127,882,913,945,979,1000]
        * | | | | | 1 i (hiding: pp !0:Nat) [1113,890,916,947,983,1003]
bh12* | | | | | | 1 i (hiding: t !reset:time) [1114,552]
bh13* | | | | | | | 1 exit !false:Bool !false:Bool !$f6:eframe  ** EXIT SUCCEED ** [1115,891,917,948,984,1004,553]
        * | | | 4 phl ?[fram,sframe,f,f,f,f]eframe                        (4)
            [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
            eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
            [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
            [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]
            [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
            [and(correct(f),is-disc(get-f(f)))]  [1127,882,913,938,979,1007]

       * | | | | 1 i (enable: exit !false:Bool !m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
                  eframe) [1008]
bh14* | | | | | 1 i (hiding: pp !$1:Nat) [1118,890,916,941,983,1011]
        * | | | | | | 1 i (hiding: t !reset:time) [1119,552]
        * | | | | | | | 1 i (hiding: p !$0:Nat) [1120,900,925,950,986,1012]
bh15* | | | | | | | | 1 phl !m(make_dm(flagging,a,ctl_uframe(1),fc,flagging),noinfor,correct,notless,noabort)
                    :eframe [1121,901,926,951,987,1013]
        * | | | | | | | | | 1 exit !false:Bool !false:Bool !$f8  ** EXIT SUCCEED ** [1122,902,927,952,988,1014,550]
        * | | | 5 phl ?[fram,sframe,f,f,f,f]eframe                        (5)
            [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
             eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
            [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
            [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]
            [and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f))))]
            [not(and(correct(f),is-disc(get-f(f))))]  [1127,882,913,938,964,1000]
       * | | | | 1 i (enable: exit !true:Bool  !m(make_sabm(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
```

```
            :eframe) [965]
        * | | | | | 1 i (hiding: [collis(m(make_sabm(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort),
                      m(make_sabm(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort))]
                      pp !$1:Nat) [1118,890,916,941,1035,1003] ==> continue
bh16* | | | | | | 1 i (hiding: t !reset:time) [1119,552]
bh17* | | | | | | | 1 i (hiding: p !$0:Nat) [1120,900,925,950,1036,1021]
bh18* | | | | | | | | 1 phl !m(make_ua(flagging,a,ctl_uframe(1),fc,flagging),noinfor,correct,notless,noabort) :
                        eframe [1121,901,926,951,1037,1022] ==> continue
        * | | | 6 phl ?[fram,f,f,sframe,f,f]eframe                    (6)
            [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
            [not(and(correct(f),is-disc(get-f(f))))]
            [not(or(and(correct(sframe),is-sabm(get-f(sframe))),or(is-disc(get-f(sframe)),or(and(is-ua(get-f(sframe)),
              eq(pf(sframe),1)),and(is-dm(get-f(sframe)),eq(pf(sframe),1))))))]
            [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
            [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]  [1127,895,913,938,979,1000] ==> again bh9
```

TSABM_resolve_collis_0   Tree

```
bh0 * 1 [eq(0,0)] i (hiding: t !start:time) [1052,536]
bh1 * | 1 i (specified explicitly) [537]
bh2 * | | 1 i (hiding: t !expired:time) [1055,538]
bh3 * | | | 1 i (hiding: [lt(0,$3)] p !$1:Nat) [1096,1150] ==> continue
       * | | 2 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
          [and(and(is_ua(get_f([fram])),eq(pf([fram]),$1)),correct([fram]))]  [874,874,874,874,1064,874]
bh4 * | | | 1 i (hiding: t !reset:time) [1065,541] ==> continue
       * | | | 2 i (hiding: t !expired:time) [1055,538] ==> again bh3
       * | | 3 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
         [or(incorrect([fram]),or(and(is_ua(get_f([fram])),eq(pf([fram]),$0)),not(is_ua(get_f([fram])))))]
            [874,874,874,874,1070,874] ==> again bh2
      * | 2 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
         [and(and(is_ua(get_f([fram])),eq(pf([fram]),$1)),correct([fram]))]  [874,874,874,874,1064,874]
bh5 * | | 1 i (specified explicitly) [537] ==> again bh4
       * | | 2 i (hiding: t !reset:time) [1065,541]
bh6 * | | | 1 exit !true:Bool !false:Bool !m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor ,correct,notless,noabort):
              eframe ** EXIT SUCCEED ** [875,875,875,875,976,875,553]
       * | 3 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
           [or(incorrect([fram]),or(and(is_ua(get_f([fram])),eq(pf([fram]),$0)),not(is_ua(get_f([fram])))))]
            [874,874,874,874,1070,874] ==> again bh1
```

TSABM_resolve_collis_1   Tree

```
bh0  * 1 [eq(Succ(0),Succ(0))] i (hiding: t !start:time) [1052,536]
bh1  * | 1 i (specified explicitly) [537]
bh2  * | | 1 i (hiding: t !expired:time) [1055,538]
bh3  * | | | 1 exit !true:Bool !false:Bool !m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor ,correct,notless,noabort):
              eframe ** EXIT SUCCEED ** [875,875,875,875,976,875,553]
        * | | 2 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
           [and(and(is_ua(get_f([fram])),eq(pf([fram]),$1)),correct([fram]))]  [874,874,874,874,1064,874]
bh4 * | | | 1 i (hiding: t !reset:time) [1065,541] ==> continue
       * | | | 2 i (hiding: t !expired:time) [1055,538] ==> again bh3
       * | | 3 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
          [or(incorrect([fram]),or(and(is_ua(get_f([fram])),eq(pf([fram]),$0)),not(is_ua(get_f([fram])))))]
           [874,874,874,874,1070,874] ==> again bh2
   * | 2 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
         [and(and(is_ua(get_f([fram])),eq(pf([fram]),$1)),correct([fram]))]  [874,874,874,874,1064,874]
bh5 * | | 1 i (specified explicitly) [537] ==> again bh4
       * | | 2 i (hiding: t !reset:time) [1065,541]
bh6 * | | | 1 exit !true:Bool !false:Bool !m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor ,correct,notless,noabort):
```

eframe ** EXIT SUCCEED ** [875,875,875,875,976,875,553]
* | 3 phl ?[fram,sframe,fram,fram,fram,fram]:eframe
[or(incorrect([fram]),or(and(is_ua(get_f([fram])),eq(pf([fram]),$0)),not(is_ua(get_f([fram])))))]
[874,874,874,874,1070,874] ==> again bh2

TSABM_resolve_collis_2 Tree

bh0 * 1 [eq(Succ(Succ(0)),Succ(Succ(0)))] i (enable: exit !true:Bool !false:Bool [1090]
* | 1 exit !true:Bool !false:Bool !m(make_sabm(flagging,to,ctl_uframe(1),fc,flagging),noinfor ,correct,notless,noabort):
eframe ** EXIT SUCCEED ** [875,875,875,875,976,875]

Dataphase Tree

bh0 * 1 [ne(Succ(vsp)mod k,lu)] dl ?p:primitive
[is_dl_data_req(p)] [1664] ==> continue
* 2 [ne(Succ(vsp)mod k,lu)] [and(not(chkpt),not(notwait))] phl ?f:eframe **(2)**
[and(correct(f),and(not(is_sabm(get_f(f))),and(not(is_sabme(get_f(f))),and(not(is_dm(get_f(f))),and(not(is_disc
(get_f(f))),and(not(is_ua(get_f(f))),and(not(is_frmr(get_f(f))),and(not(is_i(get_f(f))),and(eq(adrs(f),to),
eq(pf(f),$1))))))))))] [1588]
* | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1233] ==> continue
* | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* 3 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(correct(f),or(is_sabm(get_f(f)),is_sabme(get_f(f))))] [1596]
* | 1 [eq(n,$2*$2)] phl !m(make_ua(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1253] ==> continue
* | 2 [eq(n,$2*$2)] phl !m(make_dm(flagging,to,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1258] ==> continue
* 4 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(correct(f),or(is_ua(get_f(f)),is_dm(get_f(f))))] [1603]
* | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1233] ==> continue
* | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* 5 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(and(is_valid(f),not_abort(f)),or(and(not(is_i(get_f(f))),is_inforr(f)),not(corrl(f))))] [1407]
* | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort) :eframe [1268]
==> continue
* 6 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(and(is_valid(f),not_abort(f)),is_undef_ctl(get_ctl(get_f(f))))] [1434]
* | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort):eframe [1268]
==> continue
* 7 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe **(7)**
[and(and(is_valid(f),not_abort(f)),notvalid_nrf(f,vs,lu,k))] [1412]
* | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort):eframe [1268]
==> continue
* 8 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(and(is_valid(f),not_abort(f)),and(is_i(get_f(f)),not(corrl(f))))] [1416]
* | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort):eframe [1268]
==> continue
* 9 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(correct(f),is_frmr(get_f(f)))] [1420]
* | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)

```
    :eframe [1233]
*   | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1097,1151] ==> continue
*   | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1097,1151] ==> continue
* 10  phl ?[f,f]eframe
   [and(eq(pf(f),$0),and(correct(f),is_rnr(get_f(f))))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),
    ge(vs mod k,lu)))]  [2231,1546] ==> continue
* 11  phl ?[f,f]eframe
  [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rnr(get_f(f)))))]
  [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),
   ge(vs mod k,lu)))]  [2239,1546] ==> continue
* 12  phl ?[f,f]eframe
  [and(and(eq(pf(f),$1),eq(adrs(f),from)),and(correct(f),is_rnr(get_f(f))))]
  [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),
   ge(vs mod k,lu)))]  [2247,1546]
*   | 1 [not(lbusy)] phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
    :eframe  [2256] ==> continue
* 13    phl  !  m(make_iframe(flagging,to,ctl_iframe(vs,$0,vrmod k),packe(firstone(qdls)),fc,flagging),
       noinfor,correct,notless,noabort) :eframe
     [and(not(isempty(qdls)),not(rbusy))]  [1679]
* 14  phl !m(make_iframe(flagging,to,ctl_iframe(vs,$1,vr mod k),packe(firstone(qdls)),fc,flagging),
     infor,correct,notless,noabort):eframe
    [and(not(isempty(qdls)),not(rbusy))]  [1697]
* 15  phl ?[f,f]eframe
    [and(is_rej(get_f(f)),eq(pf(f),$0))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
   [2155,1746] ==> continue
* 16  phl ?[f,f]eframe
   [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rej(get_f(f)))))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
   [2163,1746] ==> continue
 * 17  phl ?[f,f]eframe
   [and(is_rej(get_f(f)),and(eq(pf(f),$1),eq(adrs(f),from)))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
      [2171,1746]
 * | 1 phl ! m(make_rr(flagging,from,ctl_sframe(1,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
    [2178] ==> continue
 * 18  [not(lbusy)] phl ?[f,f]eframe                         (18)
   [and(eq(pf(f),$0),and(correct(f),and(is_i(get_f(f)),eq(ns(f),vr mod k))))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [1756,1746]
*   | 1 phl ! m(make_iframe(flagging,to,ctl_iframe(vs,0,(vr mod k)),packe(firstone(qdls)),fc,flagging),
   noinfor,correct,notless,noabort)  [not(isempty(qdls)) and not(rbusy)] [1679] ==> continue
*   | 2 phl ! m(make_rr(flagging,from,ctl_sframe(0,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
    [2068] ==> continue
*   | 3 phl ! m(make_rnr(flagging,from,ctl_sframe(0,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
    [2211] ==> continue
* 19  [not(lbusy)] phl ?[f,f]eframe
   [and(eq(pf(f),$1),and(correct(f),and(is_i(get_f(f)),eq(ns(f),vr mod k))))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [1764,1746]
*   | 1 phl ! m(make_rr(flagging,from,ctl_sframe(1,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
    [1769] ==> continue
*   | 2 phl ! m(make_rnr(flagging,from,ctl_sframe(1,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
    [1774] ==> continue
* 20  phl ?[f,f]eframe
```

```
      [and(is_rr(get_f(f)),eq(pf(f),$0))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [1968,1746]
*    | 1 phl ! m(make_rr(flagging,to,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
        [2068] ==> continue
*    | 2 phl ! m(make_rr(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
       [2073] ==> continue
*    | 3 m(make_rr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort) [2078] ==> continue
*    | 4 i (enable: exit !0:Nat !0:Nat !0:Nat !0:Nat !empty:pqueue !empty:pqueue !empty:pacqueue !false:Bool
             !false:Bool !false:Bool !true:Bool !false:Bool !false:Bool !0:Nat !false:Bool !$0:Nat !nors:frmrreason !0:Nat
             !0:Bit) [2083] ==> continue
*    | 5 phl ! m(make_rnr(flagging,to,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          :eframe [2211] ==> continue
*    | 6 phl ! m(make_rnr(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          :eframe [2216] ==> continue
*    | 7 phl ! m(make_rnr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
         :eframe [2221] ==> continue
* 21  phl ?[f,f]eframe
     [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rr(get_f(f)))))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [1976,1746] ==> continue
 * 22  phl ?[f,f]eframe
      [and(is_rr(get_f(f)),and(eq(pf(f),$1),eq(adrs(f),from)))]
      [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),
       ge(vs mod k,lu)))]  [1984,1746]
   *   | 1 [not(lbusy)] phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          :eframe [1991] ==> continue
  * 23  phl ?f:eframe
      [not(is_valid(f))]  [1193] ==> continue
  * 24  [not(rbusy)] phl !m(make_rr(flagging,to,ctl_sframe($0,vr mod k),fc,flagging),
            noinfor,correct,notless,noabort):eframe [2068] ==> continue
  * 25  [not(rbusy)] phl !m(make_rr(flagging,from,ctl_sframe($0,vr mod k),
           fc,flagging),noinfor,correct,notless,noabort):eframe [2073] ==> continue
  * 26  phl !m(make_rr(flagging,to,ctl_sframe($1,vr mod k),fc,flagging),noinfor,correct,notless,noabort)
           :eframe [2078] ==> continue
  * 27  [and(not(lbusy),not(reject))] phl ?[f,f]eframe        (27)
        [and(and(correct(f),is_i(get_f(f))),(ne(ns(f),vr mod k))] [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),
      lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]  [2091,1746]
  *   | 1 phl ! m(make_rej(flagging,to,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
        :eframe  [2100] ==> continue
  *   | 2 phl ! m(make_rej(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
        :eframe  [2104] ==> continue
  *   | 3 phl ! m(make_rej(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
        :eframe [2110] ==> continue
 * 28  [and(not(lbusy),not(reject))] phl ? f : eframe[(correct(f) and is_i(get_f(f))) and (((ns(f) ne (vr mod k))) and
       ((pf(f) eq 1 of Bit)))]
   | 1 phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct, notless,noabort)
        :eframe  [2116] ==> continue
   * 29  [not(rbusy)] phl !m(make_rnr(flagging,to,ctl_sframe($0,vr mod k),fc,flagging),noinfor,correct,notless,noabort)
         :eframe [2211] ==> continue
   * 30  [not(rbusy)] phl !m(make_rnr(flagging,from,ctl_sframe($0,vr mod k),fc,flagging),noinfor,correct,notless,
       noabort) :eframe [216] ==> continue
   * 31  [not(rbusy)] phl !m(make_rnr(flagging,to,ctl_sframe($1,vr mod k),fc,flagging),noinfor,correct,notless,
            noabort) :eframe [2221] ==> continue
  * 32  i (hiding: [not(reject)] t !expired:time) [2310,538]
  *   | phl ! m(make_rr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
          :eframe [2342] ==> continue
  *   | phl ! m(make_rnr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
```

```
           :eframe  [2351] ==> continue
      * 33  dl !dl_dis_req:primitive [847]
      * | 1 phl !m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
              :eframe [1097,115] ==> continue
      * 34  phl ?sframe:eframe
        [and(correct(sframe),is_disc(get_f(sframe)))]  [856]
     *  | 1 m(make_ua(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort) [857] ==> continue
```

FRMR Tree

```
bh0 * 1 phl ?f:eframe
      [not(is_valid(f))]  [1185]
bh1 * | 1 i (enable: exit) [1186] ==> again bh0
      * 2 phl ?f:eframe
      [and(and(is_valid(f),not_abort(f)),or(and(not(is_i(get_f(f))),is_inforr(f)),not(corrl(f))))]  [1407]
bh2 * | 1 i (enable: exit !f:eframe) [1409]
bh3 * | | | 1 phl !m(make_frmr(flagging,var,ctl_uframe($1),var,fc,flagging),noinfor,correct,notless,noabort)
            :eframe [1301] ==> again bh0
      * 3 phl ?f:eframe
       [and(and(is_valid(f),not_abort(f)),is_undef_ctl(get_ctl(get_f(f))))]  [1412]
bh4 * | 1 i (enable: exit !f:eframe) [1413]
bh5 * | | 1 i (enable: exit !$1:Nat !make_reason(ctrl(f),$0,var,$0,var,$1,$1,$1,$1,$0):frmrreason
              !var:pqueue !var:pqueue !var:pacqueue !$1:Bit) [1301] ==> again bh3
       * 4 phl ?f:eframe
      [and(and(is_valid(f),not_abort(f)),notvalid_nrf(f,vs,lu,k))]  [1416]
bh6 * | 1 i (enable: exit !f:eframe) [1417]
bh7 * | | 1 i (enable: exit !$1:Nat !make_reason(ctrl(f),$0,var,$0,var,$1,$1,$1,$1,$0):frmrreason
            !var:pqueue !var:pqueue !var:pacqueue !$1:Bit) [1301] ==> again bh3
       * 5 phl ?f:eframe
      [and(and(is_valid(f),not_abort(f)),and(is_i(get_f(f)),not(corrl(f))))]  [1420]
bh8 * | 1 i (enable: exit !f:eframe) [1421]
bh9 * | | 1 i (enable: exit !$1:Nat !make_reason(ctrl(f),$0,var,$0,var,$1,$1,$1,$1,$0):frmrreason
                   !var:pqueue !var:pqueue !var:pacqueue !$1:Bit) [1301] ==> again bh3
      * 6 phl ?[f,f]:eframe
      [and(and(correct(f),eq(adrs(f),from)),eq(pf(f),$1))]
      [not(or(or(or(is_sabm(get_f(f)),is_sabme(get_f(f))),or(is_frmr(get_f(f)),is_dm(get_f(f)))),
         is_disc(get_f(f))))]  [1305,1398]
bh10* | 1 i (enable: exit) [1305,1399]
bh11* | | 1 phl !m(make_frmr(flagging,var,ctl_uframe($1),var,fc,flagging),noinfor,correct,notless,noabort)
               :eframe [1309] ==> again bh0
       * 7 phl ?[f,f]:eframe
      [and(correct(f),or(eq(adrs(f),from),and(eq(adrs(f),to),eq(pf(f),$0))))]
      [not(or(or(or(is_sabm(get_f(f)),is_sabme(get_f(f))),or(is_frmr(get_f(f)),is_dm(get_f(f)))),
          is_disc(get_f(f))))]  [1313,1398]
bh12* | 1 i (enable: exit) [1313,1398] ==> again bh0
       * 8 phl ?[f,f]:eframe
      [correct(f)]
      [or(or(is_sabm(get_f(f)),is_sabme(get_f(f))),or(is_frmr(get_f(f)),is_dm(get_f(f))))]  [1319,1398]
bh13* | 1 i (enable: exit !f:eframe) [1319,1399]
bh14* | | 1 [eq(call,0)] t !reset:time [1323]
bh15* | | | 1 i (enable: exit) [1324]
bh16* | | | | 1 [or(is_sabm(get_f(f)),is_sabme(get_f(f)))] i (specified explicitly) [1329]
bh17* | | | | | 1 phl !m(make_ua(flagging,var,ctl_uframe(pf(f)),fc,flagging),noinfor,correct,notless,noabort)
                  :eframe [1330] ==> continue
       * | | | | 2 [or(is_sabm(get_f(f)),is_sabme(get_f(f)))] i (specified explicitly) [1329]
bh18* | | | | | 1 phl !m(make_dm(flagging,var,ctl_uframe(pf(f)),fc,flagging),noinfor,correct,notless,noabort):eframe [1334]
bh19* | | | | | | 1 pp !termine:term [1335] ==> continue
       * | | | | 3 [and(is_dm(get_f(f)),eq(pf(f),$1))] [1341] ==> bh0
```

* | | | | 4 [and(is_dm(get_f(f)),eq(pf(f),$0))] i (specified explicitly) [1345]
bh20* | | | | | 1 phl !m(make_disc(flagging,var,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1346] ==> again bh101
* | | | | 5 [and(is_dm(get_f(f)),eq(pf(f),$0))] i (specified explicitly) [1345]
bh21* | | | | | 1 i (hiding: [lt(0,n2)] p !$1:Nat) [1096,1150]
bh22* | | | | | | 1 phl !m(make_sabm(flagging,var,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* | | | | | 2 i (hiding: [eq(0,n2)] p !$0:Bit) [1108,1153]
bh23* | | | | | | 1 i (enable: exit !false:Bool !false:Bool !$f1:eframe) [1109,1154] ==> continue
* | | | | 13 [is_frmr(get_f(f))] i (specified explicitly) [1372]
bh24* | | | | | 1 phl !m(make_dm(flagging,var,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort):eframe [1374]
bh25* | | | | | | 1 pp !termine:term [1375] ==> continue
* | | | | 14 [is_frmr(get_f(f))] i (specified explicitly) [1372]
bh26* | | | | | 1 i (hiding: [lt(0,n2)] p !$1:Nat) [1096,1150]
bh27* | | | | | | 1 phl !m(make_sabm(flagging,var,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* | | | | | 2 i (hiding: [eq(0,n2)] p !$0:Bit) [1108,1153]
bh28* | | | | | | 1 i (enable: exit !false:Bool !false:Bool !$f1:eframe) [1109,1154] ==> continue
* | | 2 [eq(call,$1)] i (enable: exit) [1327] ==> again bh97
* 9 [ok] phl ?sframe:eframe
[and(correct(sframe),is_disc(get_f(sframe)))] [856]
* | 1 m(make_ua(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort)
:eframe [857] ==> continue
* 10 i (hiding: t !expired:time) [1272,549]
* | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort) [1268]
* | | 1 i (hiding: t !start:time) [1269,547] ==> again bh0

REJ Tree

bh0 * 1 [ne(Succ(vsp)mod k,lu)] dl ?p:primitive
[is_dl_data_req(p)] [1664] ==> continue
* 2 [ne(Succ(vsp)mod k,lu)] [and(not(chkpt),not(notwait))] phl ?f:eframe
[and(correct(f),and(not(is_sabm(get_f(f))),and(not(is_sabme(get_f(f))),and(not(is_dm(get_f(f))),
and(not(is_disc(get_f(f))),and(not(is_ua(get_f(f))),and(not(is_frmr(get_f(f))),and(not(is_i(get_f(f))),
and(eq(adrs(f),to),eq(pf(f),$1))))))))))] [1588]
* | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1233] ==> continue
* | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* 3 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(correct(f),or(is_sabm(get_f(f)),is_sabme(get_f(f))))] [1596]
* | 1 [eq(n,$2*$2)] phl !m(make_ua(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1253] ==> continue
* | 2 [eq(n,$2*$2)] phl !m(make_dm(flagging,to,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1258] ==> continue
* 4 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(correct(f),or(is_ua(get_f(f)),is_dm(get_f(f))))] [1603]
* | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1233] ==>
* | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
:eframe [1097,1151] ==> continue
* 5 [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
[and(and(is_valid(f),not_abort(f)),or(and(not(is_i(get_f(f))),is_inforr(f)),not(corrl(f))))] [1407]

```
* | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
     :eframe  [1268] ==> continue
 * 6  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
    [and(and(is_valid(f),not_abort(f)),is_undef_ctl(get_ctl(get_f(f))))]  [1434]
 *  | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
       :eframe [1268] ==> continue
 * 7  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
     [and(and(is_valid(f),not_abort(f)),notvalid_nrf(f,vs,lu,k))]  [1412]
 *  | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
     :eframe [1268] ==> continue
 * 8  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
    [and(and(is_valid(f),not_abort(f)),and(is_i(get_f(f)),not(corrl(f))))]  [1416]
 *   | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
   :eframe [1268] ==> continue
 * 9  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
    [and(correct(f),is_frmr(get_f(f)))]  [1420]
 *   | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1233]
 *   | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1097,1151] ==> continue
 *   | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1097,1151] ==> continue
 * 10   phl ?[f,f]eframe
     [and(eq(pf(f),$0),and(correct(f),is_rnr(get_f(f))))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [2231,1546] ==> continue
 * 11  phl ?[f,f]eframe
     [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rnr(get_f(f)))))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [2239,1546] ==> continue
 * 12  phl ?[f,f]eframe
    [and(and(eq(pf(f),$1),eq(adrs(f),from)),and(correct(f),is_rnr(get_f(f))))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2247,1546]
 *   | 1 phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
     : eframe  [2261] ==> continue
 *   | 2 phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
     :eframe  [2265] ==> continue
  *  13   phl ! m(make_iframe(flagging,to,ctl_iframe(vs,$0,vrmod k),packe(firstone(qdls)),fc,flagging),
         noinfor,correct,notless,noabort) :eframe
        [and(not(isempty(qdls)),not(rbusy))] **false** [1679]
  * 14  phl !m(make_iframe(flagging,to,ctl_iframe(vs,$1,vr mod k),packe(firstone(qdls)),fc,flagging),
        infor,correct,notless,noabort):eframe
         [and(not(isempty(qdls)),not(rbusy))] **false** [1697]
   * 15  phl ?[f,f]eframe
     [and(is_rej(get_f(f)),eq(pf(f),$0))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [2155,1746] ==> continue
   * 16  phl ?[f,f]eframe
     [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rej(get_f(f)))))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2163,1746] ==> continue
  * 17  phl ?[f,f]eframe
    [and(is_rej(get_f(f)),and(eq(pf(f),$1),eq(adrs(f),from)))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2171,1746]
   *   | 1 phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
    :eframe  [2183] ==> continue
   *   | 2 phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
```

```
    :eframe  [2187] ==> continue
 * 18  [not(lbusy)] phl ?[f,f]eframe
      [and(eq(pf(f),$0),and(correct(f),and(is_i(get_f(f)),eq(ns(f),vr mod k))))]
      [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
      [1756,1746]
 *   | 1 phl ! m(make_iframe(flagging,to,ctl_iframe(vs,0,(vr mod k)),packe(firstone(qdls)),fc,flagging),
        noinfor,correct,notless,noabort)  [not(isempty(qdls)) and not(rbusy)] [1679] ==> continue
 *   | 2 phl ! m(make_rr(flagging,from,ctl_sframe(0,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
       [2068] ==> continue
 *   | 3 phl ! m(make_rnr(flagging,from,ctl_sframe(0,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
      [2211] ==> continue
 * 19  [not(lbusy)] phl ?[f,f]eframe
     [and(eq(pf(f),$1),and(correct(f),and(is_i(get_f(f)),eq(ns(f),vr mod k))))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [1764,1746]
 *   | 1 phl ! m(make_rr(flagging,from,ctl_sframe(1,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
     [1769] ==> continue
 *   | 2 phl ! m(make_rnr(flagging,from,ctl_sframe(1,((vr + Succ(0)) mod k)),fc,flagging),noinfor,correct,notless,noabort)
      :eframe [1774] ==> continue
 * 20  phl ?[f,f]eframe
     [and(is_rr(get_f(f)),eq(pf(f),$0))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [1968,1746] ==> continue
 * 21   phl ?[f,f]eframe
     [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rr(get_f(f)))))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [1976,1746] ==> continue
  * 22   phl ?[f,f]eframe
     [and(is_rr(get_f(f)),and(eq(pf(f),$1),eq(adrs(f),from)))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [1984,1746]
  *   | 1 phl ! m(make_rr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
      :eframe  [1996] ==> continue
  *   | 2 phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
      :eframe  [2000] ==> continue
  * 23   phl ?f:eframe
     [not(is_valid(f))]  [1193] ==> continue
  * 24   [reject] phl ?[f,f]eframe
     [and(and(and(correct(f),is_i(get_f(f))),(ne(ns(f),vr mod k))),eq(pf(f),0))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [2121,1746] ==> continue
  * 25   [reject] phl ?[f,f]eframe
     [and(and(and(correct(f),is_i(get_f(f))),(ne(ns(f),vr mod k))),eq(pf(f),1))]
     [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
     [2130,1746]
  *   | 1 phl ! m(make_rej(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
  * 26   i (hiding: [(reject)] t !expired:time) [2310,538]
  * | 1 phl ! m(make_rej(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
      :eframe  [2359] ==> continue
  * 27   dl !dl_dis_req:primitive [847]
  * | 1 phl !m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
      :eframe [1097,115] ==> continue
  * 28   phl ?sframe:eframe
     [and(correct(sframe),is_disc(get_f(sframe)))] [856]
*    | 1 m(make_ua(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort)
        :eframe  [857] ==> continue
```

RNR Tree

```
* 1  [ne(Succ(vsp)mod k,lu)] [and(not(chkpt),not(notwait))] phl ?f:eframe
 [and(correct(f),and(not(is_sabm(get_f(f))),and(not(is_sabme(get_f(f))),and(not(is_dm(get_f(f))),and(not(
  is_disc(get_f(f))),and(not(is_ua(get_f(f))),and(not(is_frmr(get_f(f))),and(not(is_i(get_f(f))),and(eq(adrs(f),to),
  eq(pf(f),$1))))))))))]  [1588]
*   | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1233] ==> continue
*   | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1097,1151] ==> continue
*   | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
   :eframe [1097,1151] ==> continue
* 2  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
  [and(correct(f),or(is_sabm(get_f(f)),is_sabme(get_f(f))))]  [1596]
*   | 1 [eq(n,$2*$2)] phl !m(make_ua(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
   :eframe [1253] ==> continue
*   | 2 [eq(n,$2*$2)] phl !m(make_dm(flagging,to,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
   :eframe [1258] ==> continue
* 3  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
   [and(correct(f),or(is_ua(get_f(f)),is_dm(get_f(f))))]  [1603] ==> continue
*   | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1233] ==> continue
*   | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1097,1151] ==> continue
*   | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
    :eframe [1097,1151] ==> continue
 * 4  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
    [and(and(is_valid(f),not_abort(f)),or(and(not(is_i(get_f(f))),is_inforr(f)),not(corrl(f))))]  [1407]
 *  | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
       :eframe [1268] ==> continue
 * 5  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
     [and(and(is_valid(f),not_abort(f)),is_undef_ctl(get_ctl(get_f(f))))]  [1434]
 *  | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
     :eframe  [1268] ==> continue
 * 6  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
     [and(and(is_valid(f),not_abort(f)),notvalid_nrf(f,vs,lu,k))]  [1412]
  *  | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
     :eframe  [1268] ==> continue
  * 7  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
     [and(and(is_valid(f),not_abort(f)),and(is_i(get_f(f)),not(corrl(f))))]  [1416] ==> continue
  *   | 1 phl ! m(make_frmr(flagging,to,ctl_uframe(1),reasonfrmr,fc,flagging),noinfor,correct,notless,noabort)
      :eframe  [1268] ==> continue
  * 8  [ne(Succ(vsp)mod k,lu)] phl ?f:eframe
       [and(correct(f),is_frmr(get_f(f)))]  [1420]
  *   | 1 [or(eq(n,$2),eq(n,$3))] phl !m(make_dm(flagging,from,ctl_uframe($0),fc,flagging),noinfor,correct,notless,noabort)
     :eframe [1233] continue
  *   | 2 [or(eq(n,$2),eq(n,$3))] phl !m(make_sabm(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
     :eframe [1097,1151] ==> continue
  *   | 3 [or(eq(n,$2),eq(n,$3))] phl !m(make_disc(flagging,from,ctl_uframe(b),fc,flagging),noinfor,correct,notless,noabort)
     :eframe [1097,1151] ==> continue
  * 9  phl ?[f,f]eframe
    [and(eq(pf(f),$0),and(correct(f),is_rnr(get_f(f))))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2231,1546] ==> continue
   * 10  phl ?[f,f]eframe
    [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rnr(get_f(f)))))]
  [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
   [2239,1546] ==> continue
```

```
* 11 [ok] phl ?[f,f]eframe
    [and(and(eq(pf(f),$1),eq(adrs(f),from)),and(correct(f),is_rnr(get_f(f))))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2247,1546]
 *  | 1 [(lbusy)] phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
      :eframe   [2271] ==> continue
 * 12  phl !m(make_iframe(flagging,to,ctl_iframe(vs,$0,vr mod k),packe(firstone(qdls)),fc,flagging),
       noinfor,correct,notless,noabort):eframe  [and(not(isempty(qdls)),not(rbusy))] **false** [1679]
 * 13  phl !m(make_iframe(flagging,to,ctl_iframe(vs,$1,vr mod k),packe(firstone(qdls)),fc,flagging),
        infor,correct,notless,noabort):eframe  [and(not(isempty(qdls)),not(rbusy))] **false** [1697]
 * 14   phl ?[f,f]eframe
    [and(is_rej(get_f(f)),eq(pf(f),$0))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2155,1746] ==> continue
 * 15  phl ?[f,f]eframe
    [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rej(get_f(f)))))]
    [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2163,1746] ==> continue
 * 16  phl ?[f,f]eframe
   [and(is_rej(get_f(f)),and(eq(pf(f),$1),eq(adrs(f),from)))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [2171,1746]
* | 1 phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
    :eframe  [2194] ==> continue
* 17  phl ?[f,f]eframe
   [(correct(f) and is_i(get_f(f))) and  ((pf(f) ne 1 of Bit) and (ns(f) eq (vr mod k)))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
   [1918,1746] ==> continue
* 17   phl ?[f,f]eframe
   [(correct(f) and is_i(get_f(f))) and  ((pf(f) eq 1 of Bit) and (not(ns(f) eq (vr mod k))))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
   [1924,1746]]
* | 1 phl ! m(make_rnr(flagging,from,ctl_sframe(0,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort):
      :eframe   [1929] ==> continue
* 18  phl ?[f,f]eframe
   [(correct(f) and is_i(get_f(f))) and  ((pf(f) eq 1 of Bit) and (ns(f) eq (vr mod k))) [1933,1746]
* | 1 phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
     :eframe  [1938] ==> continue
* 19   phl ?[f,f]eframe
   [and(is_rr(get_f(f)),eq(pf(f),$0))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
    [1968,1746] ==> continue
* 20   phl ?[f,f]eframe
   [and(chkpt,and(and(eq(pf(f),$1),eq(adrs(f),to)),and(correct(f),is_rr(get_f(f)))))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
   [1976,1746] ==> continue
* 21   phl ?[f,f]eframe
   [and(is_rr(get_f(f)),and(eq(pf(f),$1),eq(adrs(f),from)))]
   [or(and(or(ge(nr(f),lu),le(nr(f),vs mod k)),lt(vs mod k,lu)),and(and(ge(nr(f),lu),le(nr(f),vs mod k)),ge(vs mod k,lu)))]
   [1984,1746]
 *  | 1 phl ! m(make_rnr(flagging,from,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
     :eframe   [2006] ==> continue
* 22  phl ?f:eframe
   [not(is_valid(f))]  [1193] ==> continue
* 23  i (hiding: [not(reject)] t !expired:time) [2310,538]
* | 1 phl ! m(make_rr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
      :eframe   [2342] ==> continue
*   | phl ! m(make_rnr(flagging,to,ctl_sframe(1,(vr mod k)),fc,flagging),noinfor,correct,notless,noabort)
      :eframe [2351] ==> continue
```

```
      * 24   dl !dl_dis_req:primitive [847]
      * | 1 phl !m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
            :eframe [1097,115] ==> continue
      * 25  phl ?sframe:eframe
         [and(correct(sframe),is_disc(get_f(sframe)))]  [856]
      *   | 1 m(make_ua(flagging,from,ctl_uframe(pf(sframe)),fc,flagging),noinfor,correct,notless,noabort)
             :eframe   [857] ==> continue
~
```

LINKDISC Tree

```
bh0 * 1 i (hiding: [lt(init,n2)] p !Succ(0):Nat) [1096,1150]
bh1 * | 1 phl !m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort):eframe [1097,1151]
bh2 * | | 1 i (hiding: t !start:time) [1098,547]
bh3 * | | | 1 i (specified explicitly) [548]
bh4 * | | | | 1 i (hiding: t !expired:time) [1100,549]
bh5 * | | | | | 1 i (enable: exit) [1100]
bh6 * | | | | | | 1 i (hiding: [lt(init,n2)] p !Succ(0):Nat) [1096,906,931,956,992,1027]
bh7 * | | | | | | | 1 phl !m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
                         :eframe [1097,907,932,957,993,1028] ==> again bh2
       * | | | | | | 2 i (hiding: [eq(init,n2)] p !0:Bit) [1108,1153]
bh8 * | | | | | | | 1 exit !false:Bool !false:Bool  $f1  ** EXIT SUCCEED ** [1109,1154,550]
bh9 * | | | 2 phl ?[fram,sframe,f,f,f,f]eframe
             [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
               eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
             [and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1)))]
             [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]
             [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
             [not(and(correct(f),is-disc(get-f(f))))]  [1127,882,921,938,979,1000]
       * | | | | | 1 i (hiding: pp !0:Nat) [1113,890,922,941,983,1003]
bh10* | | | | | | 1 i (hiding: t !reset:time) [1114,552]
bh11* | | | | | | | 1 exit !true:Bool !false:Bool !$f1:eframe  ** EXIT SUCCEED ** [1115,891,917,942,984,1004,553]
        * | | | 3 phl ?[fram,sframe,f,f,f,f]eframe
             [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
               eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
             [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
             [and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1)))]
             [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
             [not(and(correct(f),is-disc(get-f(f))))]  [1127,882,913,945,979,1000]
        * | | | | | 1 i (hiding: pp !0:Nat) [1113,890,916,947,983,1003]
bh12* | | | | | | 1 i (hiding: t !reset:time) [1114,552]
bh13* | | | | | | | 1 exit !true:Bool !false:Bool !$f6:eframe  ** EXIT SUCCEED ** [1115,891,917,948,984,1004,553]
        * | | | 4 phl ?[fram,sframe,f,f,f,f]eframe
             [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
              eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
             [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
             [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]
             [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
             [and(correct(f),is-disc(get-f(f)))]  [1127,882,913,938,979,1007]
        * | | | | 1 i (enable: exit !true:Bool !m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort)
                   :eframe)  [1008]
        * | | | | | 1 i (hiding: [collis(m(make_disc(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort),
                     m(make_sabm(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort))]
                     pp !$1:Nat) [1118,890,916,941,983,1035]
 bh14* | | | | | | 1 i (hiding: t !reset:time) [1119,552]
bh15* | | | | | | | 1 i (hiding: p !$0:Nat) [1120,900,925,950,986,1036]
bh16* | | | | | | | | 1 phl !m(make_ua(flagging,a,ctl_uframe(1),fc,flagging),noinfor,correct,notless,noabort)
                         :eframe [1121,901,926,951,987,1037] ==> continue
```

```
      * | | | 5 phl ?[fram,sframe,f,f,f,f]eframe
            [and(correct(sframe),or(is-sabm(get-f(sframe)),or(or(is-disc(get-f(sframe)),and(is-ua(get-f(sframe)),
                eq(pf(sframe),1))),and(is-dm(get-f(sframe)),eq(pf(sframe),1)))))]
            [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
            [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]
            [and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f))))]
            [not(and(correct(f),is-disc(get-f(f))))]  [1127,882,913,938,964,1000]
      * | | | | 1 i (enable: exit !false:Bool !m(make_sabm(flagging,b,ctl_uframe($1),fc,flagging),noinfor,correct,notless,noabort):
                eframe) [965]
bh14* | | | | | 1 i (hiding: pp !$1:Nat) [1118,890,916,941,968,1003]
      * | | | | | | 1 i (hiding: t !reset:time) [1119,552]
      * | | | | | | | 1 i (hiding: p !$0:Nat) [1120,900,925,950,969,1021]
bh15* | | | | | | | | 1 phl !m(make_dm(flagging,a,ctl_uframe(1),fc,flagging),noinfor,correct,notless,noabort) :
                     eframe [1121,901,926,951,970,1022]
      * | | | | | | | | | 1 exit !true:Bool !false:Bool !$f8  ** EXIT SUCCEED ** [1122,902,927,952,971,1023,550]
      * | | | 6 phl ?[fram,f,f,sframe,f,f]eframe
            [not(and(correct(f),or(is-sabm(get-f(f)),is-sabme(get-f(f)))))]
            [not(and(correct(f),is-disc(get-f(f))))]
            [not(or(and(correct(sframe),is-sabm(get-f(sframe))),or(is-disc(get-f(sframe)),or(and(is-ua(get-f(sframe)),
              eq(pf(sframe),1)),and(is-dm(get-f(sframe)),eq(pf(sframe),1))))))]
            [not(and(correct(f),and(is-ua(get-f(f)),eq(pf(f),1))))]
            [not(and(correct(f),and(is-dm(get-f(f)),eq(pf(f),1))))]  [1127,895,913,938,979,1000] ==> again bh9
```

References

[ADLU] A.V. Ahon, A.T. Dahbura, D. Lee, M.U Uyan, "An optimized technique for protocol conformance test generation based on UIO sequences and rural chinese postman tours", in: S. Aggarwal and K. Sabnani (eds) Protocol Specification, Testing, and Verification VIII, North-Holland, pp.75-86, 1988.

[BB] Ed. Brinksma, T. Bolognesi, "Introduction to the ISO specification language LOTOS", North-Holland, Computer Networks and ISDN systems 14, pp.25-59, 1987.

[BDS] G.V Bochmann, R. Dssouli, R. Sarikaya, "Methodes de tests de protocoles: architecture et selection de tests", Ingenierie des Protocoles, CFIP, pp.337-363, 1988.

[BGPU] T. Boyce, T. Grenier, R. L. Probert, H. Ural, "Formalization of ISDN LAPD for Conformance Testing", INFOCOMM 89, Ottawa, Ontario, Canada, April 1989.

[BK] E. Brinksma, G. Karjoth, "A Specification of the OSI Transport Service in LOTOS", in Y. Yemini, R. Strom, and S. Yemini (eds) Protocol Specification, Testing, and Verification IV, North-Holland, pp.227-251, 1985.

[BV] G.V. Bochmann, J. Vaucher, "Adding performance aspects to specification languages", in: S. Aggarwal and K. Sabnani (eds) Protocol Specification, Testing, and Verification VIII, North-Holland, pp.19-31, 1988.

[Boch] G.V. Bochmann, "Specifications of a Simplified Transport Protocol using Different Description Techniques", University of Montreal, Technical Report, April 1987.

[Brink1] Ed. Brinksma, "A theory for the derivation of tests", in: S. Aggarwal and K. Sabnani (eds) Protocol Specification, Testing, and Verification VIII, North-Holland, pp.19-31, 1988.

[Brink2] Ed. Brinksma, "On the existence of canonical testers", Technical report, University of Twente, January 1987.

[Britel] British Telecommunications, "Fomal methods applied to protocols (FORMA), A framework for the underlying concepts of communications protocols", February 1987.

[Brui] J. Bruijning, "Evaluation and Integration of Specification Languages", North-Holland, Computer Networks and ISDN systems 13, 1987.

[EB] E.H. Eirtink, Ed. Brinksma, "The implementation of a test derivation algorithm", Technical report, University of Twenty.

[FLP] J.P. Favreau, R.J. Linn Jr.,P. Gaudette, "Application of Formal Description

Techniques to Conformance Evaluation.", in: K.J Turner (ed) Proccedings of the first International Conference on Formal Description Techniques (FORTE), North-Holland, pp.295-309, 1988.

[FLS] M. Faci, L. Logrippo, B. Steipien, "Formal specifications of telephone systems in LOTOS", University of Ottawa, Department of Computer Science, Technical Report, University of Ottawa 89-07,February 1989.

[GHL] R. Guillemont, M. Haj-hussein, L. Logrippo, "Executing large LOTOS specifications", in: S. Aggarwal and K. Sabnani (eds) Protocol Specification, Testing, and Verification VIII, North-Holland, pp.399-410, 1988.

[Haj] M. Haj-Hussein, "An Interactive System for LOTOS Applications (ISLA)", MCS Thesis, University of Ottawa, November 1988.

[Hoa] C.A.R. Hoare, "Communicating Sequential Processes", Prentice-Hall International, London, 1985.

[ISO1] International Organization for Standardization, Information Processing Systems, Open Systems Interconnection, "LOTOS- A Formal Description Technique based on Temporal Ordering of Observational Behaviour" (ISO DIS 8807), 1987.

[ISO2] International Organization for Standardization, Information Processing Systems, Open Systems Interconnection, "Formal Description of the OSI connection-oriented transport Protocol in LOTOS" (SEDOS/119.7), 1987.

[ISO3] International Organization for Standardization, Information Processing Systems, Open Systems Interconnection, "Formal Description of the OSI Transport Service in LOTOS" (SEDOS/119.6), 1987.

[ISO4] International Organization for Standardization, Telecommunications and Information Exchange Between Systems, "X.25-DTE Conformance Testing: Revised Text for Data Link Layer Test Suite for DP 8882 part 2", ISO/JTC1/SC6/WG1, April 1988.

[ISO5] International Organization for Standardization, Telecommunications and Information Exchange Between Systems, "X.25-DTE Conformance Testing: Revised Text for Data Link Layer Test Suite for DP 8882 part 2", ISO/IEC TC1/SC 6/WG 1, January 1989.

[ISO6] International Organization for Standardization, Information Retrieval, Transfer and Management for OSI, "The Tree and Tabular Combined Notation (TTCN)" (DP9646-3), ISO/IEC JTC 1/SC 21 N3077 Sydney Meeting, February 1989.

[ISO7] International Organization for Standardization, Information Retrieval, Transfer and Management for OSI, "The Tree and Tabular Combined Notation (TTCN)", ISO/

IEC JTC 1/SC 21 N Teddington/Stockholm output, September 1988.

[ISO8] International Organization for Standardization, "Description of the 1984 X.25 LAPB-Compatible DTE Data Link Procedures" (ISO 7776), ISO TC 97/ SC6, 1984.

[ISO9] International Organization for Standardization, Information Retrieval, Transfer and Management for OSI, "OSI Conformance Testing Methodology and Framework. Part 1: General Concepts." (DP9646-1), ISO TC 97/SC 21 N, December 1987.

[ISO10] International Organization for Standardization, Information Retrieval, Transfer and Management for OSI, "OSI Conformance Testing Methodology and Framework. Part 2: Abstract Test Suite Specification." (DP9646-2), ISO TC 97/SC 21 N, December 1987.

[ISO11] International Organization for Standardization, Information Retrieval, Transfer and Management for OSI, "Working Draft for the guidelines for the application of ESTELLE, LOTOS and SDL.", ISO/IEC JTC1/SC21, March 1988.

[KLPU] B. Kanungo, L. Lamont, R.L. Probert, H. Ural, "A Useful FSM Representation for Test Suite Design and Development", Proc. of 6th IFIP/WG6.1 Int. Workshop on Protocol Specification Verification and Testing Canada, pp.163-176, June 1986.

[Kan] B. Kanungo, "Specification and Design of an OSI Standard Test Suite for X.25 DTEs", MCS Thesis, University of Ottawa, 1987.

[LG] L. Logrippo, R. Guillemont, "Derivation of Useful Execution Trees From LOTOS Specifications by Using an Interpreter", in: K.J Turner (ed) Proccedings of the first International Conference on Formal Description Techniques (FORTE), North-Holland, pp.311-327, 1988.

[LO] L. Logrippo, A. Obaid, "Outils logiciels pour le langage LOTOS", Ingenierie des Protocoles, CFIP, pp.255-266, 1988.

[LS] J. Van de Langemaat, G. Scollo, "On the Use of LOTOS for the Formal Description of a Transport Protocol", in: K.J Turner (ed) Proceedings of the first International Conference on Formal Description Techniques (FORTE), North-Holland, pp.247-261, 1988.

[Linn] R.J. Linn, jr, "Conformance evaluation methodology and protocol testing", National bureau of standards, Institute of computer sciences and technology, USA.

[MA] J. de Meer, R. Ahooja, "Glasnost-oriented style for specifications of OSI systems", to appear in 9th International Symposium on Protocol, Specification, Testing, and Verification, June 1989.

[MM] J. A. Manas, T. de Miguel, "From LOTOS to C", in: K.J Turner (ed) Proccedings of the first International Conference on Formal Description Techniques (FORTE), North-Holland, pp.79-84, 1988.

[Man] J. A. Manas, "A Tutorial on ADT semantics for LOTOS users", Part one and two, Technical report, University of Madrid, May 1988.

[Myer] G.J. Myers, "The Art of Software Testing", John Wiley and Sons, 1979.

[PA] R.L. Probert, S. Akhavan, "Abstract Conformance Test Results for Open Systems", University of Ottawa, Department of Computer Science, Technical Report, University of Ottawa 88-28, October 1988.

[PUH] R.L Probert, H. Ural, M.W.A Hornbeek, "An integrated software environment for developing standardized conformance tests", in: S. Aggarwal and K. Sabnani (eds) Protocol Specification, Testing, and Verification VIII, North-Holland, pp.87-98, 1988.

[QFVM] J. Quemada, A. Fernandez, L. Vidaller, C. Miguel, "Transforming LOTOS specifications with LOLA. The parameterized expansion", in: K.J Turner (ed) Proccedings of the first International Conference on Formal Description Techniques (FORTE), North-Holland, pp.45-54, 1988.

[Raf] O. Rafiq, "L'evolution du test de l'OSI", Ingenierie des Protocoles, CFIP, pp.377-389, 1988.

[Ray] D. Rayner, "OSI Conformance Testing", Computer Networks and ISDN systems, Vol. 14, No. 1, pp.79-98, 1987.

[SAC] M.V. Sinderen, I. Ajubi, F. Caneschi, "The Application of LOTOS for the Formal Description of the ISO Session Layer", in: K.J Turner (ed) Proceedings of the first International Conference on Formal Description Techniques (FORTE), North-Holland, pp.263-278, 1988.

[SB] B. Sarikaya, G.V. Bochmann, "Synchronization and Specification Issues in Protocol Testing", IEEE Transactions on Communications, Vol. 32, No. 4, pp.389-365, 1984.

[SD] K. Sabnani, A.T. Dahbura, "A new technique for generating protocol tests", ACM transactions on Computer Communications, 1986.

[SEDOS1] Software Environment for the Design of Distributed Open Systems, "OSI Connection-Oriented Data Link Service: a Constraint-Oriented Specification in LOTOS", Public Report SEDOS/119.2, October 1987.

[SEDOS2] Software Environment for the Design of Distributed Open Systems, "OSI Connection-less Data Link Service: a Constraint-Oriented Specification in LOTOS", Public Report SEDOS/119.3, October 1987.

[SG] B. Sarikaya, Q. Gao, "Translation of test specifications in TTCN to LOTOS", in: S. Aggarwal and K. Sabnani (eds) Protocol Specification, Testing, and Verification VIII, North-Holland, pp.219-230, 1988.

[SHW] M.H. Sherif, G.L. Hoover, R.P. Wiederhold, "X.25 Conformance Testing- a Tutorial", IEEE Communications, Vol. 24, No 1, pp.16-27, January 1986.

[SL] D. Sidhu, T.K. Leung, "Formal Methods for Protocols Testing: A Detailed Study", Department of Computer Science, Iowa State University, December 1986 .

[SS] G. Scollo, M.V. Sinderen, "On the Architectural Design of the Formal Specification of the Session Standards in LOTOS", in B. Sarikaya, and G.V. Bochmann (eds) Protocol Specification, Testing, and Verification VI, North-Holland, pp.3-14, 1987.

[SVS] G. Scollo, C.A. Vissers, A. di Stefano, "LOTOS in practice", Proc. IFIP 86, 10th World Congress, Dublin, Ireland (North-Holland), pp.869-875, 1986.

[Steen] C. Steenbergen. "Conformance Testing of OSI systems", MCS Thesis, Universite of Twente, August 1986.

[Tur1] K. J. Turner, "The formal specification language LOTOS: a course for users", University of Stirling, 1987.

[Tur2] K. Turner, "Constraint-oriented style in LOTOS".

[UP] H. Ural, R.L. Probert, "Stepwise Validation of Communication Protocols and Services", North-Holland, Computer Networks and ISDN systems.

[US] H. Ural, R. Short, "An Interactive Test Sequence Generator", in: Communications Architectures and Protocols, SIGCOMM '86 Symposium, pp.241-250.

[Ural] H. Ural, "A Test Derivation Method for Protocol Conformance Testing", in: H. Rudin and C.H. West (eds) Protocol Specification, Testing, and Verification VII, North-Holland, pp.347-358, 1987.

[VSS] C.A. Vissers, G. Scollo, M.V. Sinderen, "Architecture and specification styles in formal descriptions of distributed systems" in: S. Aggarwal and K. Sabnani (eds) Protocol Specification, Testing, and Verification VIII, North-Holland, pp.189-204, 1988.

[Wez] C.D. Wezeman, "The 'CO-OP method': a method for compositional derivation of canonical testers" Technical report, University of Twenty, September 1988.

[Zim] H. Zimmerman, "OSI Reference Model. The ISO Model of Architecture for Open Systems Connection", IEEE Transactions on Communications, Vol Com-28, No. 4, April 1980.