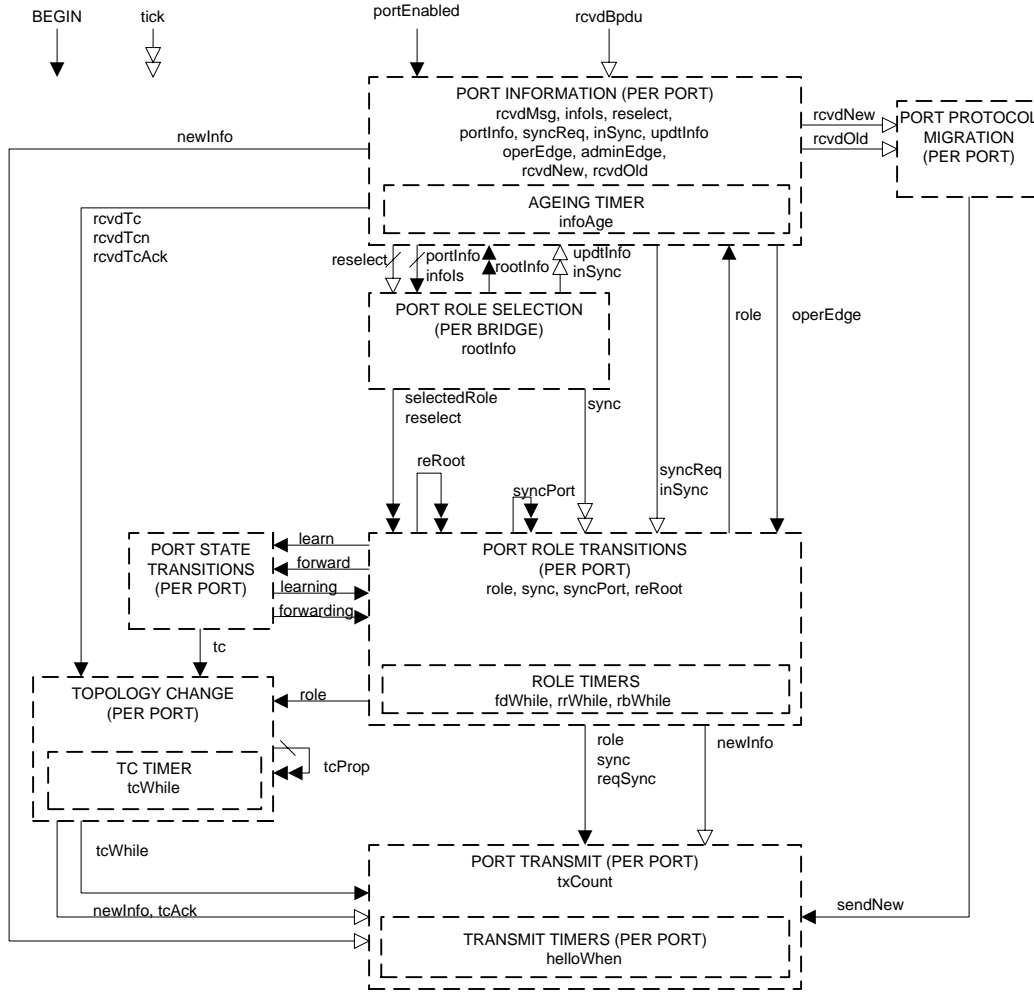


RSTP STATE MACHINES - AN OVERVIEW

NOTE: THIS OVERVIEW IS NOT ITSELF A STATE MACHINE BUT SERVES TO ILLUSTRATE THE PRINCIPAL VARIABLES THAT ARE USED TO COMMUNICATE BETWEEN THE INDIVIDUAL RSTP STATE MACHINES AND THE VARIABLES LOCAL TO EACH MACHINE

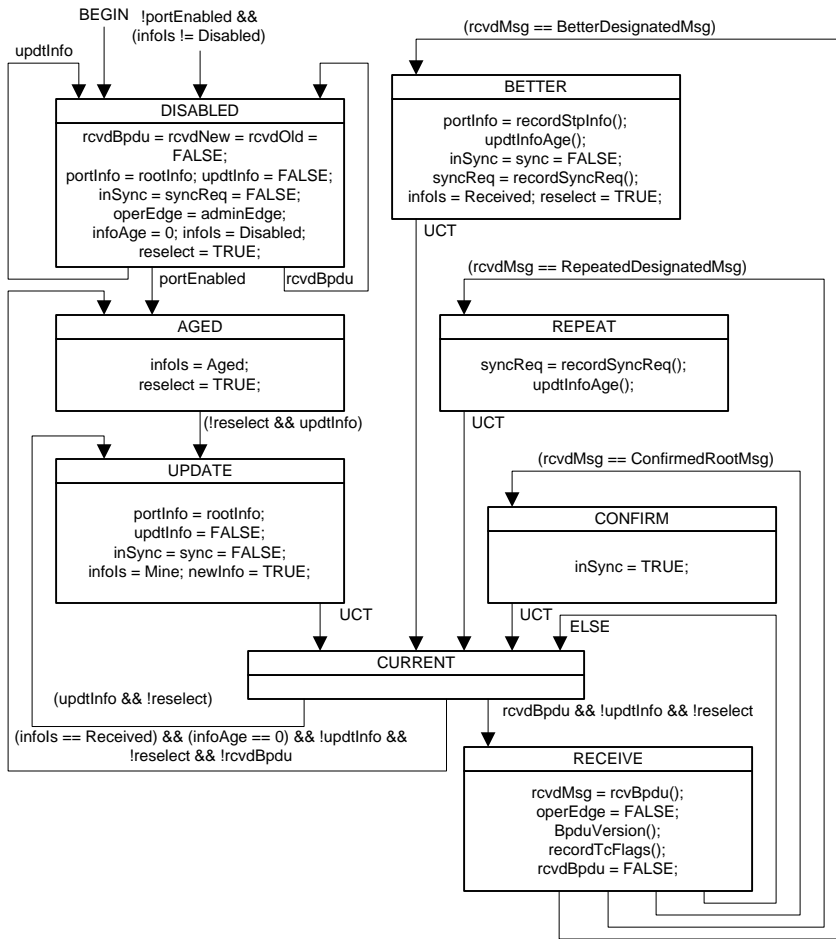


ABBREVIATIONS:

- PIM: Port Information Machine
- PRS: Port Role Selection Machine
- PRT: Port Role Transition Machine
- PST: Port State Transition Machine
- RCM: Role Confirmation Machine
- TCM: Topology Change Machine
- PPM: Port Protocol Migration Machine
- PRI: Port Role Information State Machine
- PTX: Port Transmit Machine
- PTI: Port Timers Machine

NOTE: For convenience all timers are collected together into one state machine.

PIM: PORT INFORMATION MACHINE (PER PORT)



```

recordStpInfo()
/* Copies spanning tree information, i.e. Designated Root thru Designated Port, from a ConfigBPDU or an RstpBpdu with a Designated Port Role to the vector portInfo. */
}

recordTcFlags()
/* Sets rcvdTc, rcvdTcn, and rcvdTcAck, if the Topology Change, Topology Change Notification, or Topology Change Acknowledgment flags respectively are set in a ConfigBPDU or RstpBpdu . Sets rcvdTcn if the BPDU is a TcnBpdu. */
}

recordSyncReq()
/* Sets syncReq if the BPDU is an RstpBpdu with a Designated Port Role, and the syncReq flag is set, and the attached LAN is a point to point link. NOTE: the syncReq flag is a new proposal! */
}

updtInfoAge()
/* Updates infoAge from a ConfigBPDU or an RstpBpdu with a Designated Port Role. Copies all timer parameters (Message Age, Max Age, Hello Time, Forward Delay) from the BPDU. */
}

RcvdMsg rcvBpdu()
/*
Returns BetterDesignatedMsg if the received PDU is an RstpBpdu with a Designated Port Role, or a Config BPDU, and either the spanning tree information in the received PDU is strictly better than that already held for the port (comparing Designated Root, Root Path Cost, Designated Bridge, and Designated Port parameters) or the Designated Bridge in the received PDU information is the same as that already held for the port and the spanning tree information or any of the timer parameters received with the BPDU differ from that already held.

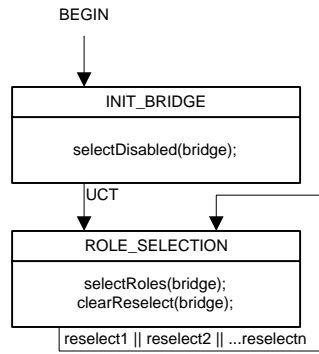
Returns RepeatedDesignatedMsg if the received PDU is an RstpBpdu with a Designated Port Role, or a Config BPDU, and the spanning tree information in the received PDU is the same as that already held for the port.

Returns ConfirmedRootMsg if the received PDU was received on a point to point link, and is an RstpBpdu with a Root Port Role, the receiving port has a Designated Port Role, and the Spanning Tree information in the received BPDU has the same Root and a worse (i.e. higher) Ro of Path Cost than the receiving port.

Otherwise, the received BPDU contains inferior information, or is a TCN BPDU. */
}

BpduVersion()
/* Sets rcvdOld if the BPDU received is a version 0 or version 1 PDU, either a TCN or a Config BPDU. Sets rcvdNew if the received BPDU is an RSTP BPDU. */
}
    
```

PRS: PORT ROLE SELECTION MACHINE (PER BRIDGE)

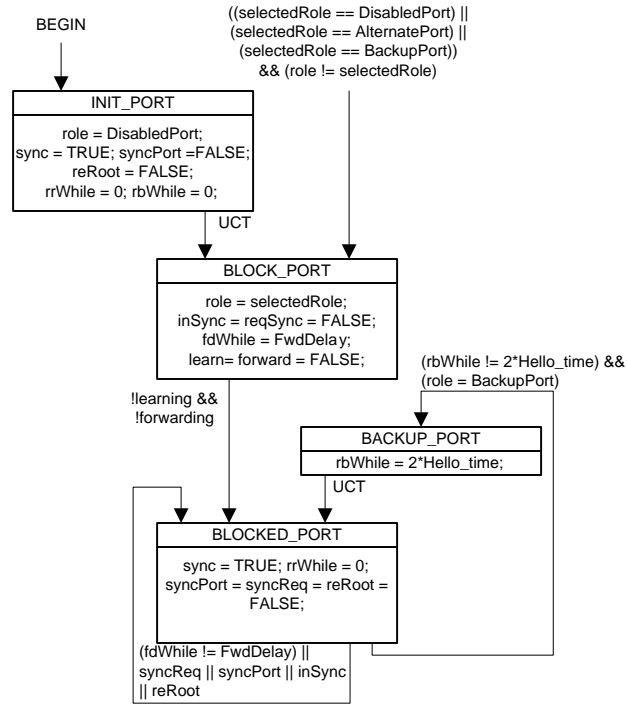


```

selectDisabled(bridge)
{ /* Sets selectedRole to DisabledPort for all bridge ports. */
}
selectRoles(bridge)
{ /* Assigns a role, i.e. sets selectedRole to one of DisabledPort, RootPort, DesignatedPort, AlternatePort, or BackupPort for each port of the bridge.
First selects the RootPort from ports with (portIs == Received), choosing the port with the best Spanning Tree Information after the port's Path Cost
has been added. If this Spanning Tree information differs from that held as rootInfo for the bridge, updates the latter and clears inSync and sync for
all bridge ports.
Then sets updtInfo for all other ports for which:
  a) (portIs == Received) and the Spanning Tree Information, without the port's own Path Cost added, is worse than the RootPort 's
with its Path Cost added
  b) (portIs == Aged)
  c) (portIs == Mine) and the Spanning Tree Information for the port or the associated timer parameters differ from those with the for
the RootPort with its Path Cost added).
*/
}
clearReselect(bridge)
{ /* Sets reselect = FALSE for all ports */
}
  
```

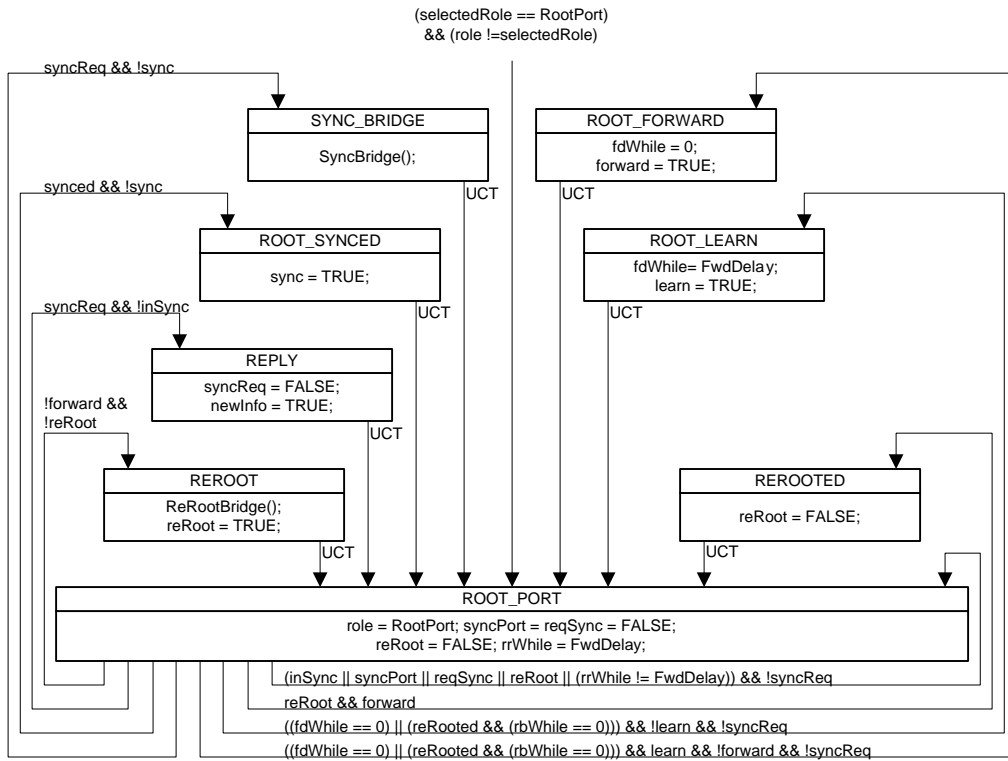
PRT: PORT ROLE TRANSITIONS STATE MACHINE (PER PORT)

PAGE 1 OF 3 : SHOWING INITIALIZATION OF THE PORT ROLE TRANSITIONS STATE MACHINE AND THE DISABLED_PORT, ALTERNATE_PORT, BACKUP_PORT AND IMMEDIATELY ASSOCIATED STATES



PRT: PORT ROLE TRANSITIONS STATE MACHINE (PER PORT)

PAGE 2 OF 3 : SHOWING THE ROOT PORT STATES



NOTE: All transtions, except UCT, qualified by "&& !reselect".

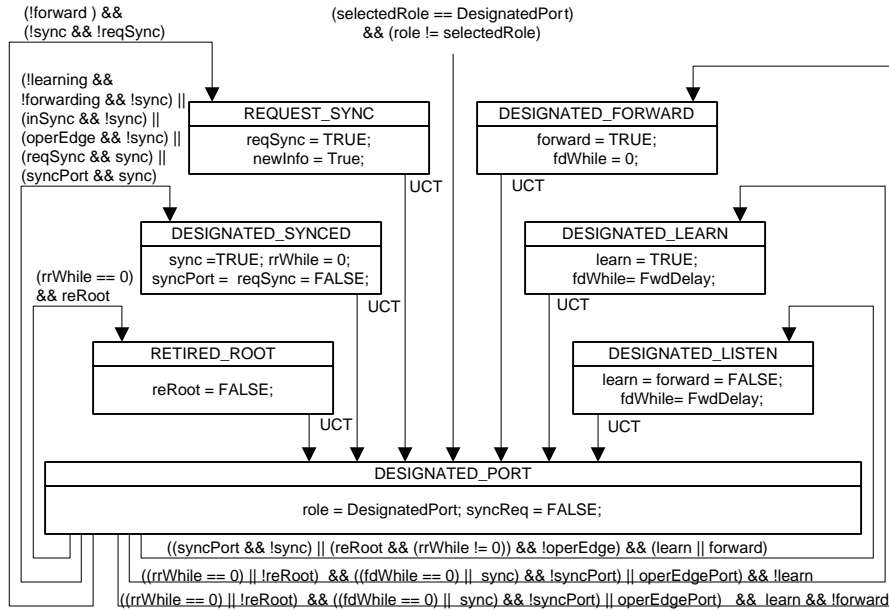
reRooted = ((rrWhile1 == 0) && (rrWhile2 == 0) && ... (rrWhilen == 0)) for all ports except this Root Port
synced = (sync1 && sync2 && ... syncn) for all ports except this Root Port

```

syncBridge()
/* Sets syncPort for all other ports. */
}
reRootBridge()
/* Sets reRoot for all other ports. */
}
    
```

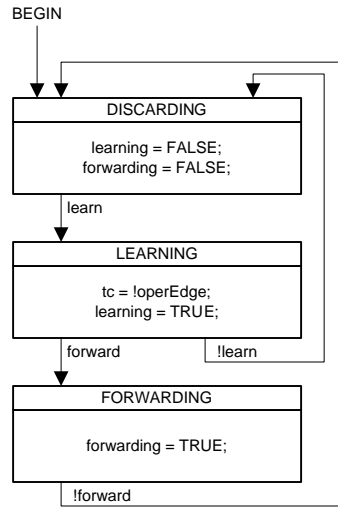
PRT: PORT ROLE TRANSITIONS STATE MACHINE (PER PORT)

PAGE 3 OF 3 : SHOWING THE DESIGNATED PORT STATES



NOTE: All transtions, except UCT, qualified by "&& !reselect".

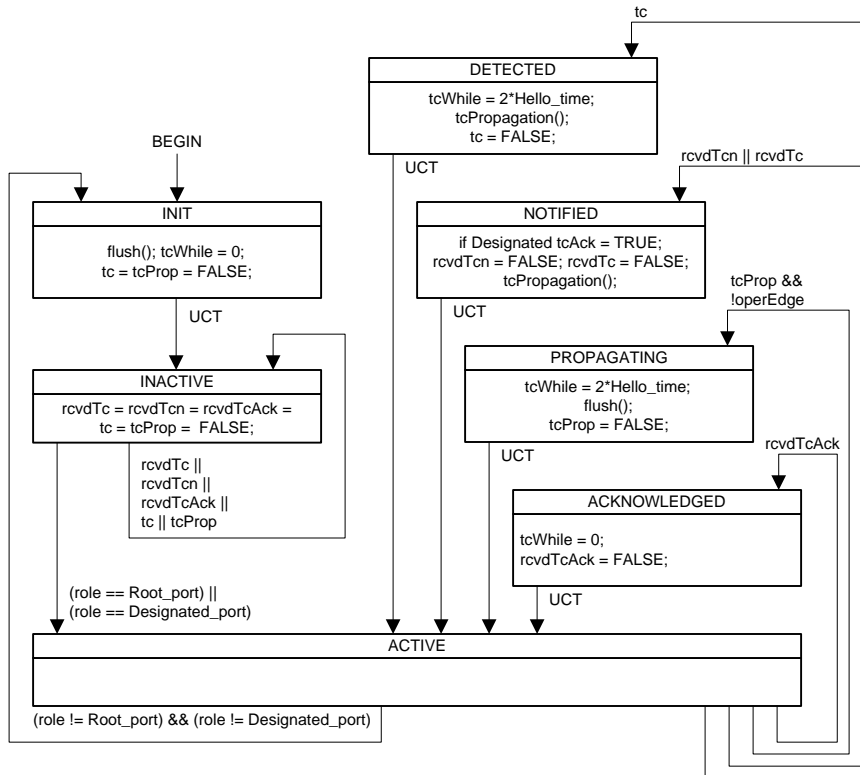
PST: PORT STATE TRANSITION MACHINE (PER PORT)



NOTE: A small system dependent delay may occur on each of the transitions shown.

TCM: TOPOLOGY CHANGE MACHINE (PER PORT)

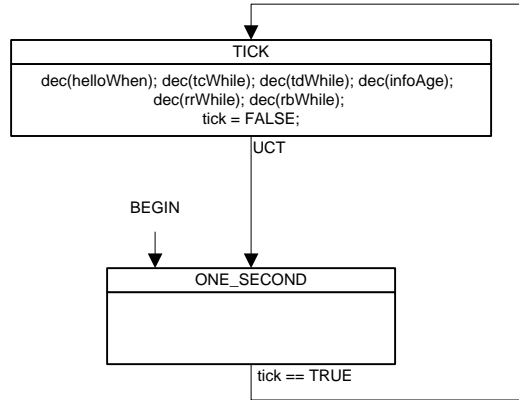
TOPOLOGY CHANGE DETECTION, NOTIFICATION, PROPAGATION AND FILTERING DATABASE FLUSHING



```

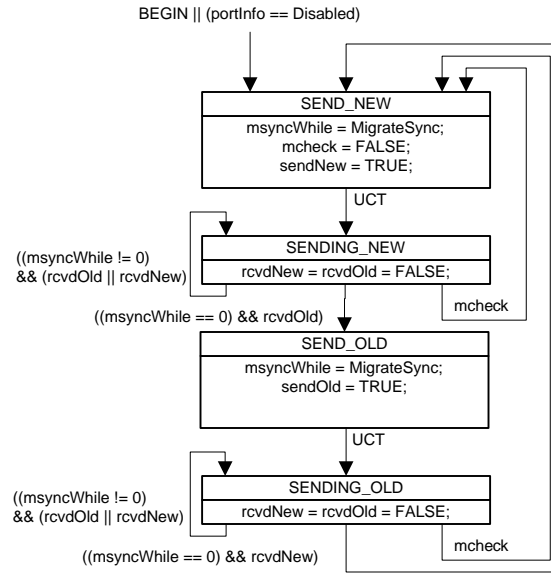
tcPropagation()
{ /* Sets tcprop for all other ports. */
}
flush()
{ /* Flushes the filtering database for this port. !!! Unless an edge port!!! */
}
    
```


PORT TIMERS STATE MACHINE (PER PORT)



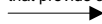
```
dec(x)  
{ if (x !=0) x= x-1;  
}
```

PPM: PORT PROTOCOL MIGRATION (PER PORT)



NOTATION:

In the Overview variables are shown both within the machine where they are principally used and between machines where they are use to communicate information. In the latter case they are shown with a variety of arrow styles, running from one machine to another, that provide an overview of how the variables are typically used:



Not changed by the target machine. Where the state machines are both per port, this variable communicates between machine instances for the same port.



Set (or cleared) by the originating machine, clear (or set) by the target machine. Where the state machines are both per port, this variable communicates between machine instances for the same port.



As above except that the originating per port machine instance communicates with multiple port machine instances (by setting or clearing variables owned by those ports).



As above except that multiple per port instances communicate with (an)other instance(s) (by setting or clearing variables owned by the originating ports).

```
typedef enum {BetterDesignatedMsg, RepeatedDesignatedMsg, OtherMsg} RcvdMsg;
typedef enum {Disabled, Mine, Received, Aged} InfoIs;
typedef struct /* StpInfo */
{
    Priority      root_pri;
    Bridge_id    root_id;
    Stp_cost     root_cost;
    Priority      bridge_pri;
    Bridge_id    bridge_id;
    Priority      port_pri;
    Port_id      port_id;
    Centisecs    message_age;
    Centisecs    max_age;
    Centisecs    forward_delay;
    Centisecs    hello_time;
}
```