PORT SUSPEND/RESUME ACTIONS
Draft 97-0053R2 Oct. 11, 1997

```
void active_actions () {
        active = TRUE;
        initiate_resume = resume_target = FALSE;
}

void suspend_initiator_actions ()   {
        connect_timer = 0;
        suspend_initiator = FALSE;              // clear the suspend request
        wait((connect_timer >= NOTIFY_HOLD) || (!port_status));
        if (!port_status)                       //TpBias driven low by target
                TpGen = LOW;                            // drive TpBias
}


void suspend_target_actions ()  {
        suspend_target = FALSE;
        connect_timer = 0;                              // start the timer
        TpGen = LOW;                            // drive TpBias low to Initiator
        wait ((connect_timer >= BIAS_HOLD) || (!port_status));
                                                        // wait for initiator to drive TpBias
}

void suspend_failed_actions ()  {               // legacy target

        if (chg_int_en && link_active)
                PH_EVENT.ind (INTERRUPT); // notify higher layers of  fault
        else if (resume && !link_active)                // or wake up the LINK
                PH_EVENT.ind (LINK_ON);
        port_power = FALSE;
}


void disconnected_actions ()  {
        port_power = TRUE;                      // Power up the port
        TpGen = HiZ;                            // But don't drive TpBias
        wait (port_functional);                 // Clock is running
        active_en = FALSE;                              // prevents port from being active until bus reset
        connection_in_progress = FALSE;
        if (!debouncing) {                      // Don't reset if debouncing a new connection
                if (child)                              // Parent still connected?
                        isbr = TRUE;                    // Yes, arbitrate for short reset
                else
                        ibr = TRUE;                     // No, core transits to R0.
        }
        if (connected || fault)
                if (link_active && chg_int_en)
                        PH_EVENT.ind (INTERRUPT);       // notify higher layers of disconnect or !fault
                else if (!link_active && resume)        // wake up the link
                        PH_EVENT.ind (LINK_ON);
        connected = FALSE;                              // Clear STATUS_A.con
        fault = FALSE;                          // Clear CONTROL_SET.fault
        debouncing = FALSE;
        initiate_resume = resume_target = FALSE;
        port_power = FALSE;                     // Shutdown the port.
}
```

```
void suspended_actions () {
        port_power = TRUE;                      // Power up the port
        wait (port_functional);                 // Clock is running
        if (link_active && chg_int_en)
                PH_EVENT.ind (INTERRUPT);               // notify higher layers of !disable or !fault or suspend
        else if (!link_active && resume)        // wake up the link
                PH_EVENT.ind (LINK_ON);
        initiate_resume = resume_target = FALSE;
        fault = FALSE;
        TpGen = LOW;                            // drive TpBias low and then release
        wait (con_status == TRUE);                      // wait to eliminate false disconnect
        port_power = FALSE;                     // turn off bias generator
}


void resume _actions ()  {
        port_power = TRUE;                      // Power up the port
        wait (port_functional);                 // Clock is running
        if (port_status)
                resume_target = resume_all = TRUE;          // This port is a resume target and resume all suspended ports
        TpGen = HIGH;                                       // drive TpBias to resume target or resume initiator
        connect_timer = 0;                                          // start timer
        wait ( (connect_timer >= DETECT_MIN) || (port_status == TRUE));   // wait for target to drive TpBias
        if (port_status) {
                fault = FALSE;
                if (chg_int_en && link_active)
                        PH_EVENT.ind (INTERRUPT);       // notify higher layers of  port resuming
                else if (!link_active && resume)        // or wake up the link
                        PH_EVENT.ind (LINK_ON);
        }
}



void resume_failed_actions ()  {
        if (!fault) {
                fault = TRUE;
                if (chg_int_en && link_active)
                        PH_EVENT.ind (INTERRUPT);               // notify higher layers of  fault
                else if (!link_active && resume)
                        PH_EVENT.ind (LINK_ON);                 // or wake up the LINK
        }
        TpGen = LOW;                            // drive TpBias low and then release
        wait (con_status == TRUE);                      // wait to eliminate false disconnect
        port_power = FALSE;                     // return to suspend state
}


void disabled_actions ()  {
        port_power = TRUE;                      // Power up the port
        wait (port_functional);                 // Clock is running
        if (chg_int_en && link_request)
                PH_EVENT.ind (INTERRUPT);               // notify higher layers of  disable
        else if (!link_active && resume)
                PH_EVENT.ind (LINK_ON);
        port_power = FALSE;                     // Stop driving TpBias
}
```

```
void connect_debounce_actions ()  {
        port_power = TRUE;                      // Power up the port
        TpGen = HiZ;                            // But don't drive TpBias
        wait (port_functional);                 // Clock is running
        connection_in_progress = TRUE;
        connect_timer = 0;                      // Start the timer
        wait (!con_status || connected || connect_timer >= (isolated_node()) ? 2 * CONNECT_TIMEOUT: CONNECT_TIMEOUT);
                // Debounce con_status
                // connected == TRUE when reset_detected() == TRUE
        if (con_status)  {              // Still connected?
                connected = TRUE;               // Yes, set STATUS_A.con
                if (isolated_node())            // Can we arbitrate?
                        ibr = TRUE;             // No, core transits to R0.
                else
                        isbr = TRUE;            // Yes, arbitrate for short reset.
        }
        connection_in_progress = FALSE;
}
```