1394c Connectivity and Low Power Management

Colin Whitby-Strevens
Apple
September 23rd 2003

Scope

This note makes a detailed proposal for the control of 1394c (aka S800T, aka T_mode) connectivity and low power management. It assumes an understanding of the 1394 services model, and in particular the use of this model in 1394b.

Data transfer at the lowest level is defined (elsewhere) by reference to the 802.3 Clause 35 GMII interface. Connectivity and low power management is defined in this note by a services model - the "Tport services" - which is intended to be complementary to the data transfer functions provided by the GMII.

Tport Services

The S800T Connectivity and Low Power management provides its services via a parameter to the PMD_CONTROL_request:-

Table 1: PMD Control requests

Service	Description
PMD_LINKPULSE_ON	Start sending a link pulse (as defined in 802.3)
PMD_LINKPULSE_OFF	Stop sending a link pulse (normally called 100ns after PMD_LINKPULSE_ON)
PMD_AUTONEGOTIATE	Carry out 1394c/802.3 Autonegotiation (as specified elsewhere)
PMD_SELECT_TPORT	Start up Tport synchronization and training (full power signaling)
PMD_UNSELECT_PORT	(Existing control request - extended function) Stop Tport full power data signaling if PMD_SELECT_TPORT had previously been called
PMD_DISCONNECT_TPORT	Stops 802.3 Clause 40 PHY operating as a Tport PHY. Clears operating mode status bits.
PMD_TPORT_OFF	All Tport circuitry disabled (including for 802.3 operation)

The Tport responds to a PMD_Status_request with one or more status flags, defined as:-

Table 2: PMD Status request flags

Service	Description
PMD_TPORT_SIGNAL_DETECT	Set if the connect detect logic detects incoming signaling energy
PMD_AUTONEGOTIATION_ ACTIVE	Set during the time that the 802.3 Clause 40 PHY is engaged in autonegotiation with the peer PHY.
PMD_TPORT_FW	Set when Autonegotiation has established a FireWire connection with its peer. Cleared when PMD_DISCONNECT_TPORT is issued.
PMD_TPORT_802	Set when Autonegotiation has established an 802.3 connection with its peer. Cleared when PMD_DISCONNECT_TPORT is issued (also may be cleared by Ethernet MAC command).
PMD_TPORT_OK	Set when the 802.3 Clause 40 PHY has established synchronization with its peer. GMII data can be transmitted and expected whenever this status flag is set.

Tport Services Controller State Machine

The 1394b state machine is updated as shown below.

Full details will be provided in the first draft specification.

P0: Disconnected

P11: Untested
untested_actions()

P2: Active

P0:P11 — connected && Beta_mode
connection_unreliable = FALSE;

P11:P2 — !loop_disabled
active = TRUE;

— P6:P11 —

P12: Loop Disabled
loop_disabled_actions()

P11:P12 — loop_disabled

P2:P11 — (Beta_mode && loop_to_detect && !bport_sync_OK) || force_disconnect
active = FALSE;
force_disconnect = TRUE;

P12:P0 — !connected
loop_disabled = FALSE;

P12:P11 — connected && (!loop_disabled || receive_ok)
loop_disabled = FALSE;

P1: Resuming
resume_actions()

P0:P1 — connected && !Beta_mode
connection_unreliable = FALSE;

P5: Suspended
suspended_actions()

P13: 802_mode

P1:P5 — resume_fault

P1:P2 — !resume_fault
active = TRUE;

P0:P13 — 802_mode

P5:P1 — connected && (resume ||
(!suspend_fault && receive_ok))
resume_fault = suspend_fault = FALSE;

P13:P0 — !802_mode

P5:P0 — !connected
suspend_fault = FALSE;
resume_fault = FALSE;

P3: Suspend Initiator
suspend_initiator_actions()

P3:P5

P5:P5 — !receive_ok && suspend_fault
suspend_fault = FALSE;

P2:P3 — ((!rx_ok && !suspend_request && !disable_request)||
(suspend_request && signaled)) &&
!(Beta_mode && loop_to_detect && !bport_sync_ok) &&
!force_disconnect
active = FALSE;

P4: Suspend Target
suspend_target_actions()

Power reset
port_power_reset()

P4:P5

P6: Disabled
disabled_actions()

P2:P4 — (rx_ok && ((portR_arb == SUSPEND) ||
(portR_arb == DISABLE_NOTIFY))) && !force_disconnect
active = FALSE;

P6:P5 — connected && !disabled &&
!Beta_mode && !802_mode

P6:P13 — 802_mode

P6:P11 — !disabled && Beta_mode &&
!802_mode
connected = TRUE;

All:P6 — (disabled && (!802_MODE || hard_disable)) ||
(disable_request && signaled)
active = FALSE;
resume_fault = suspend_fault =
standby_fault = loop_disabled = FALSE;

P6:P0 — !connected && !disabled &&
!Beta_mode && !802_mode

P10: Restoring
restore_actions()

P10:P0 — !connected

P10:P2 — connected
active = TRUE;

P7: Standby Initiator
standby_initiator_actions()

P9: Standby
standby_actions()

P7:P9

P2:P7 — rx_ok && do_standby && signaled && !force_disconnect
active = FALSE;

P9:P10 — connected && (restore ||
(!standby_fault && receive_ok))
standby_fault = FALSE;

P8: Standby Target
standby_target_actions()

P8:P9

P9:P0 — !connected
standby_fault =
in_standby = FALSE;

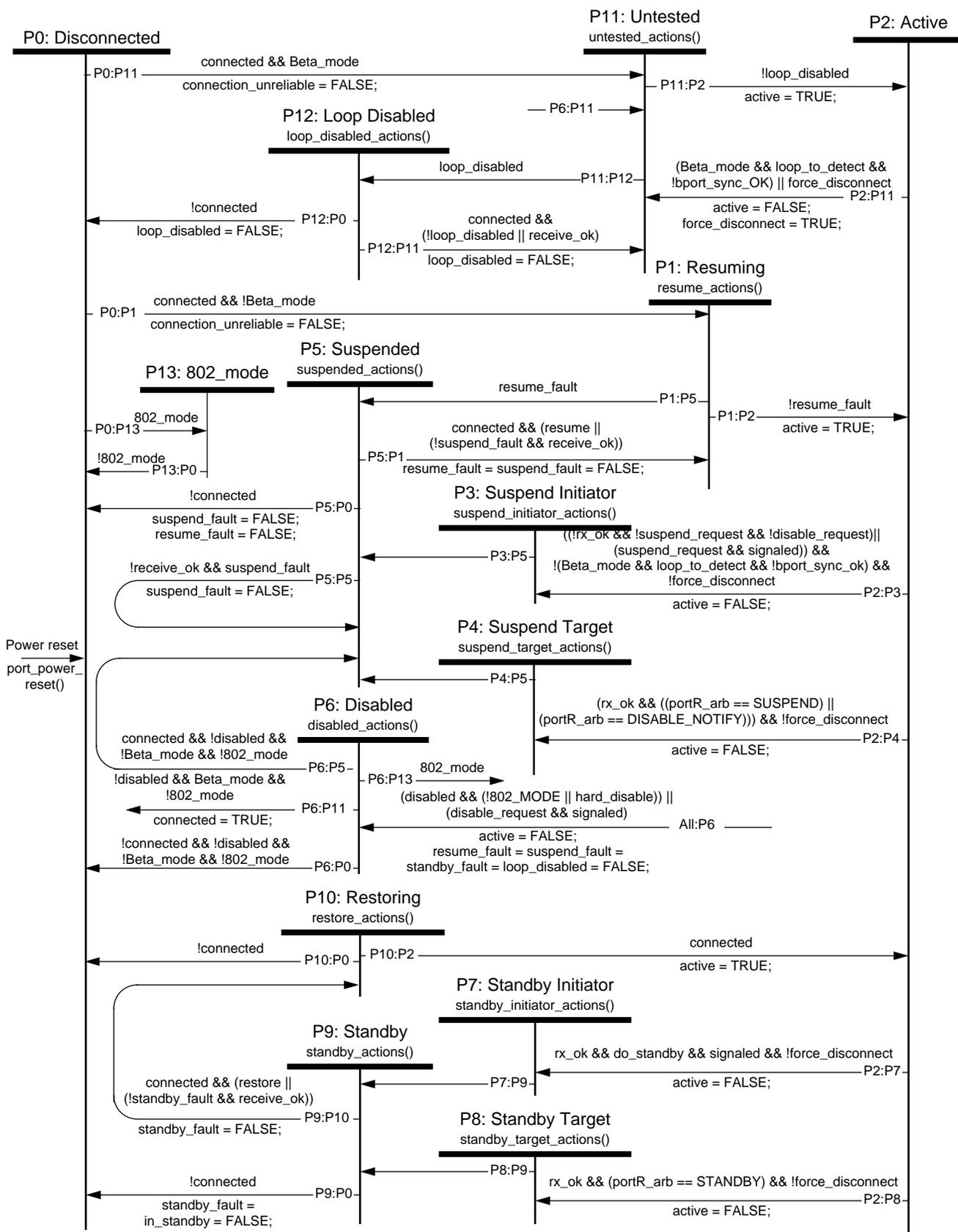P2:P8 — rx_ok && (portR_arb == STANDBY) && !force_disconnect
active = FALSE;

Figure 0-1—Port connection manager state machine