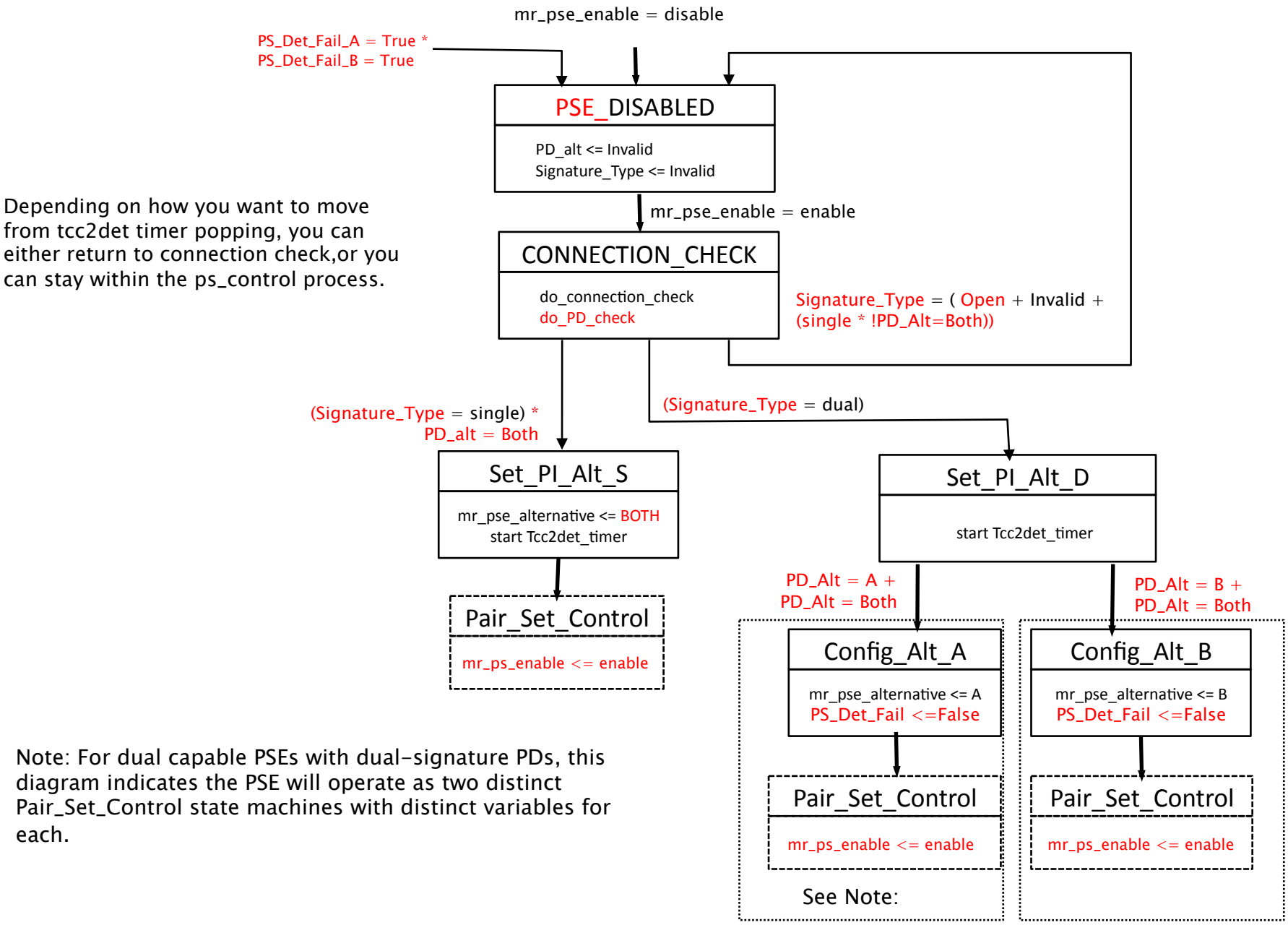
IEEE P802.3bt
PSE State Diagram Proposal

Dan Dove, DNS for LTC

September 11, 2015

Objectives of This Presentation

- Solidify a proposal to create a higher-level heirarchy that performs necessary functions to determine if PSE should apply power to single-signature PD or dual-signature PD, and then launch either one or two instances of a pair-set controller.
  - Leaves Pairset State Diagram relatively un-touched with only minor changes from Type 1 / Type 2 behavior
  - Requires some language/discussion about creation of local variables for each pairset controller
  - Requires some language/discussion about conditions within the pairset controller that would lead the PSE to resume control at the higher level.



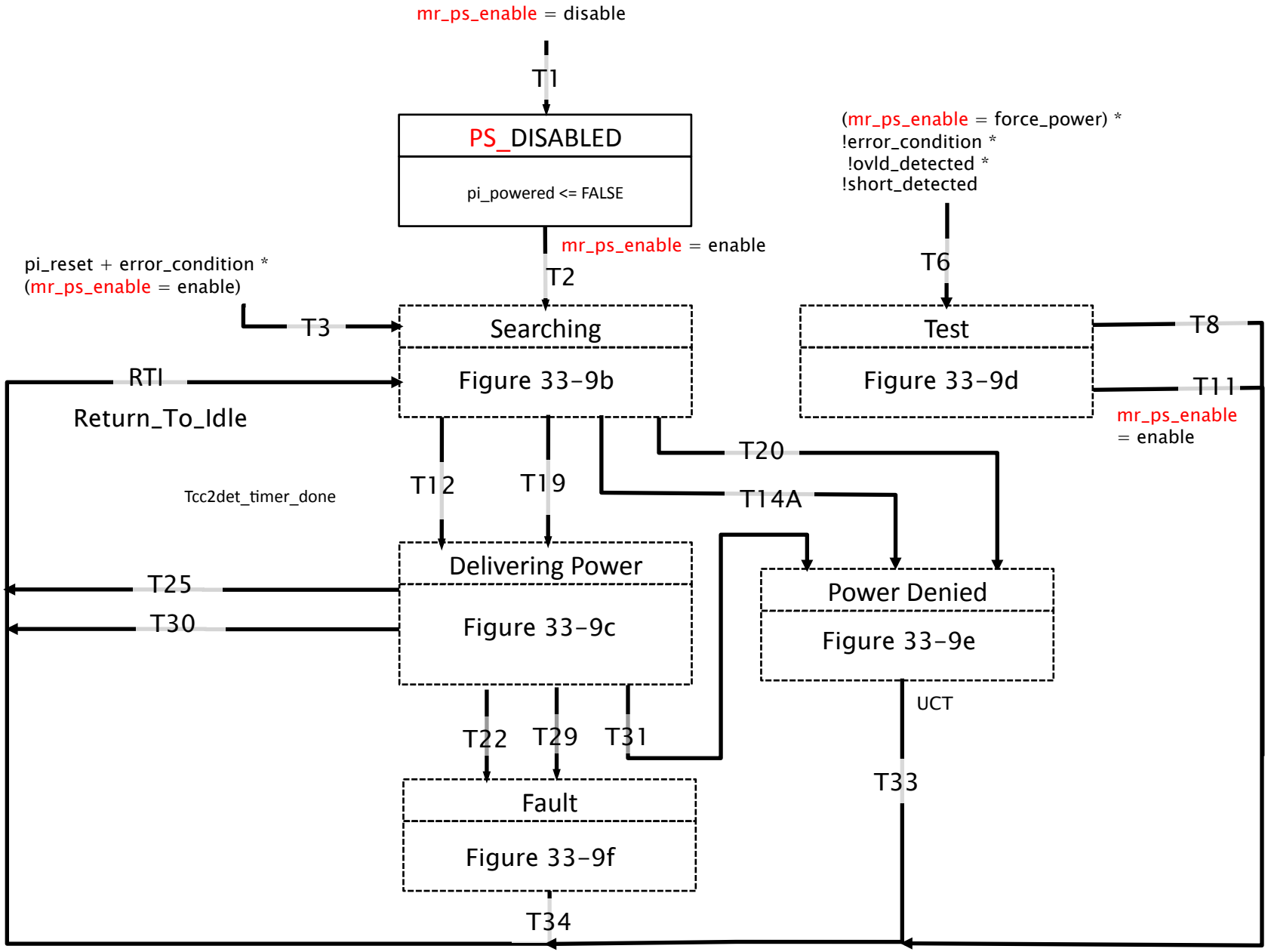
Depending on how you want to move from tcc2det timer popping, you can either return to connection check,or you can stay within the ps_control process.

Note: For dual capable PSEs with dual-signature PDs, this diagram indicates the PSE will operate as two distinct Pair_Set_Control state machines with distinct variables for each.

Type 3 and Type 4 PSE Hierarchical State Diagram

PSE Heirarchical Top Level Diagram

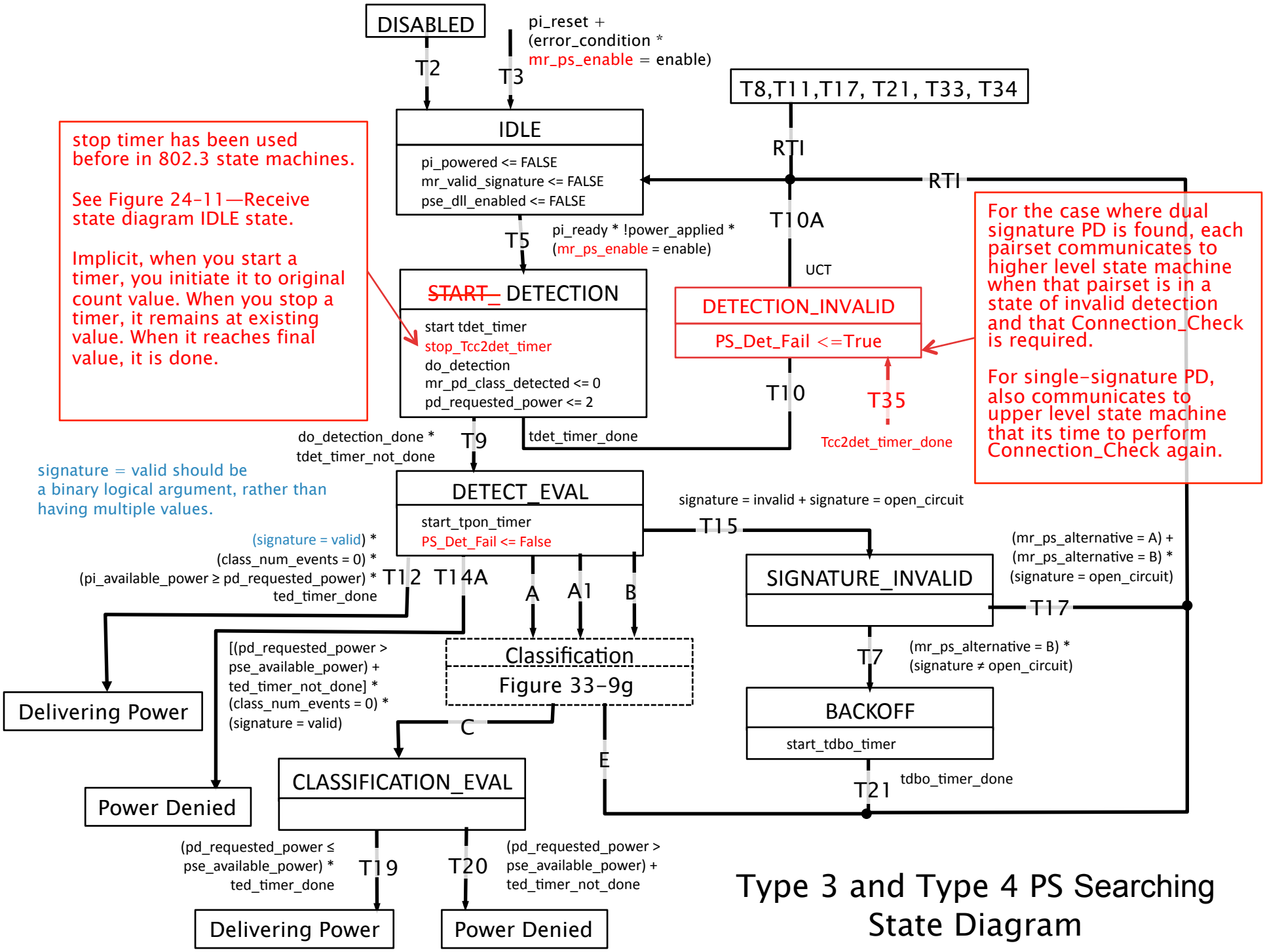
- Added entry for case where both Pairset Controllers determine they are unattached to a valid PD.
  - For dual-signature PDs, its possible one pairset controller could be disconnected while the other is powering or operating in a valid state. This would require both pairset controllers to determine that they are disconnected before returning to the higher level state block.
  - PSEs always have the ability to remove power and return to the higher level, this is a mechanism to ensure automatic return when a cable is disconnected.
- Corrected variable name to "Signature_Type" to be consistent with D1.2 for Connection_Check output.
  - Note: Case of PD Signature_Type=single, but PSE requiring power on only one pairset returns to Disable State. In essence, this would be a Type 1 or Type 2 PD, and does not want to receive power on 4 pairs.
- PS_Det_FailA and PS_Det_FailB (formerly BeamMeUp) create a nomenclature challenge. Global variables vs Local variables and how to describe them.
  - For instance, each PS controller has a variable named PS_Det_Fail. The value of that variable maps to the appropriate global variable PS_Det_FailA or PS_Det_FailB
  - I would like feedback on how to implement this paradigm in the spec.
- Placement of start_tcc2det_timer
  - If it is a global variable, placement of its initialization state is probably correct.
  - If it is a local variable for each PS controller, probably should be moved to the Config_Alt_A and Config_alt_B states.



Type 3 and Type 4 PSE Pair Set Control State Diagram

PS Control Top Level Diagram

- For Type 3 and Type 4 PDs that have dual-signature, each pairset controller operates as if its an independent PSE, subject only to the higher level PSE Control State Diagram.
  - This allows maximum flexibility on the type of implementations that can be supported, and simplifies the complex set of cases that we would have to address if both pairsets were controlled by a single state machine.
- For Type 3 and Type 4 PDs that have single-signature, a single pairset controller operates on both pairsets. This is done by using the variable mr_pse_alternative set to “both”.
- There are documentation challenges that are required to communicate how to define local variables for each pairset controller, while simultaneously retaining the original Type 1 and Type 2 paradigm within the standard where all variables were global.



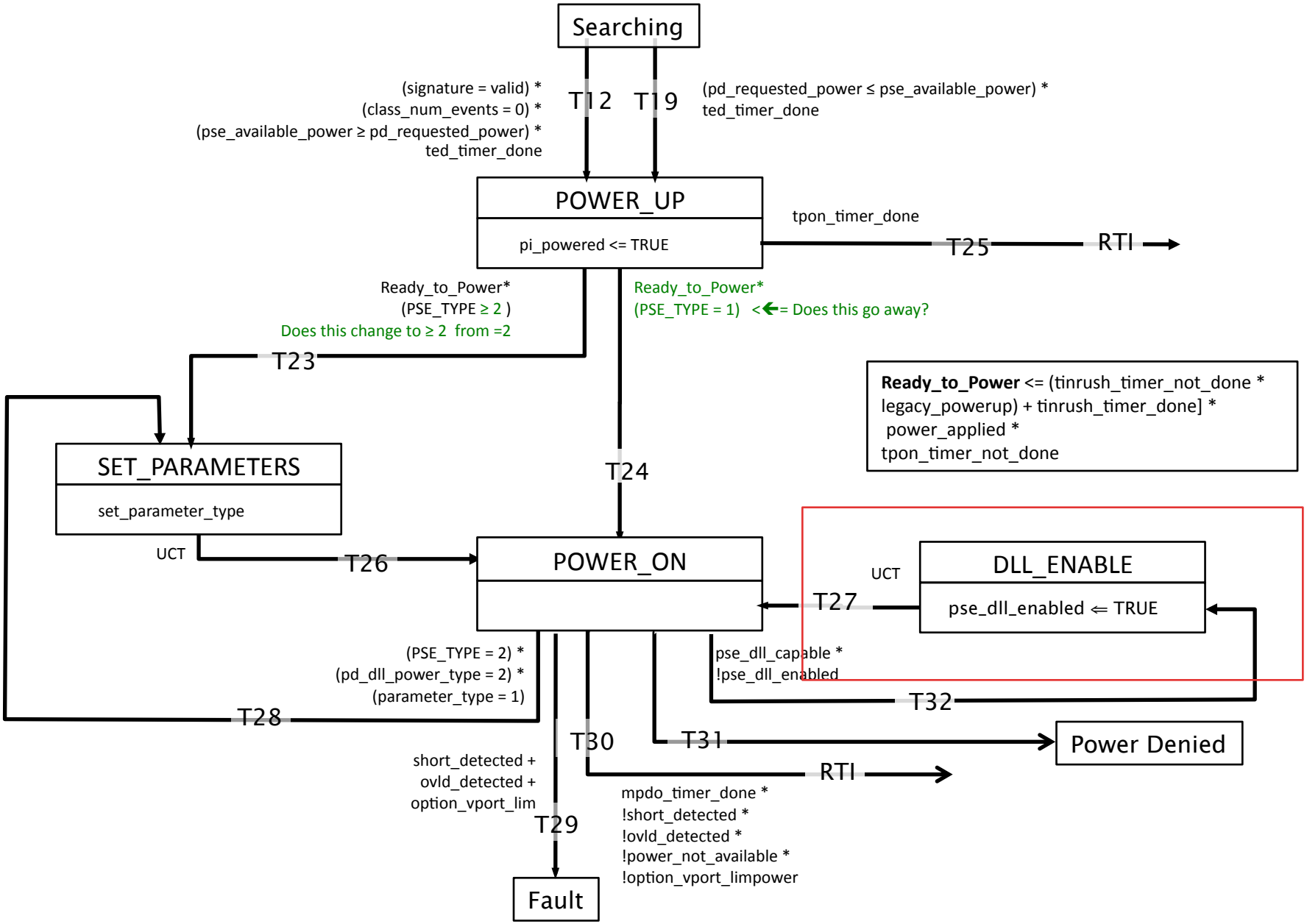
Type 3 and Type 4 PS Searching State Diagram

Searching Block Diagram

- To ensure that the time from Connection Check to Detection on each pairset does not exceed acceptable limits, the tcc2det_timer is started after Connection Check, and stopped at the beginning of detection.
  - The concept of stopping a timer has been used previously within 802.3
  - We need to think about the global/local nature of this variable. If one pairset controller stops the timer, is that sufficient or do we require that both pairset controllers must stop the timer to ensure detection on both pairsets is complete prior to tcc2det_timer_done?
  - One could have the stop_timer action be on a local version of the timer (ie: Both controllers have one and the global variable tcc2det_timer_done is true only both local timers are done)
  - Or we could have tcc2det_timer_done be a global variable that is true if either timer is stopped.
- Addition of the Detection_Invalid state
  - Not required for Type 1 or Type 2 PSEs, this state is used to communicate to the higher level block that the PS controller has failed to detect a PD or that local tcc_2det_timer has expired.

Searching Block Diagram (cont)

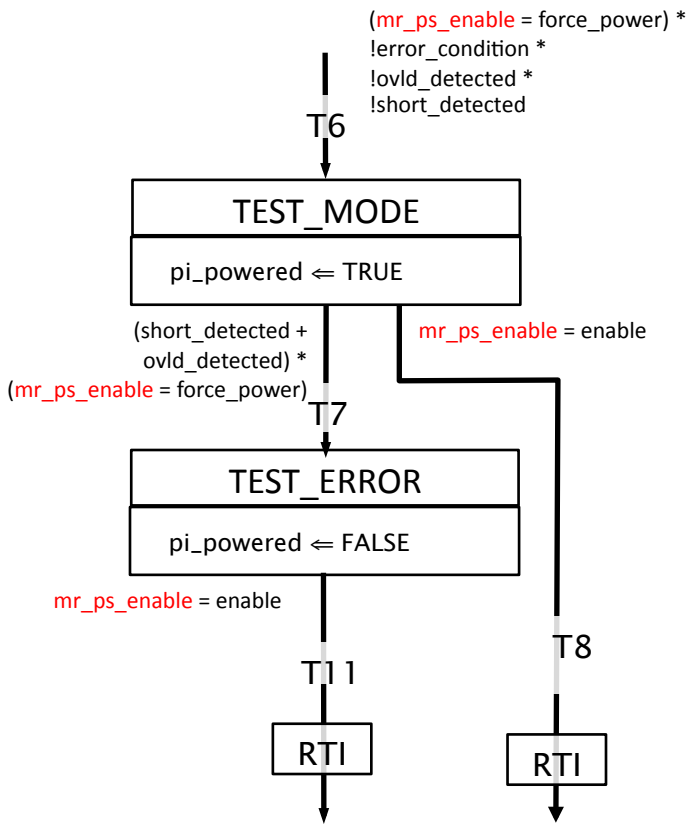
- Added "PS_Det_Fail <= False" into Detect_Eval state since the detection would have been successful and therefore this variable should be set to False.
  - Its possible a pairset controller may have cycled multiple times through IDLE, Start_Detection and Detection_Invalid states before successfully completing Detection. So this is necessary to ensure the PSE Controller doesn't retake control of the pairset.



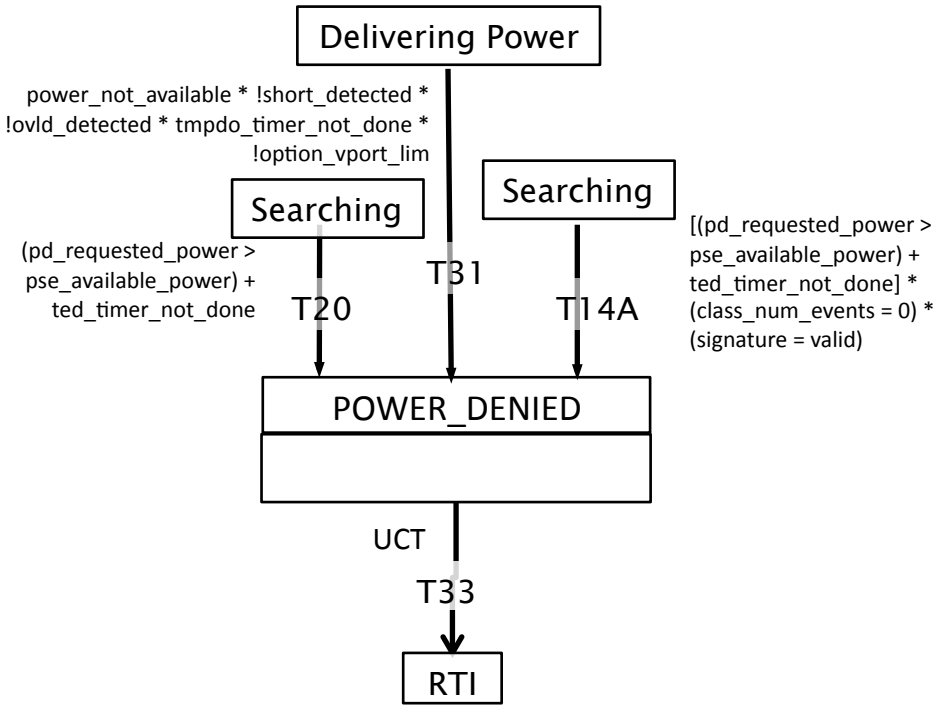
Type 3 and Type 4 PS Delivering Power State Diagram

Delivering Power Block Diagram

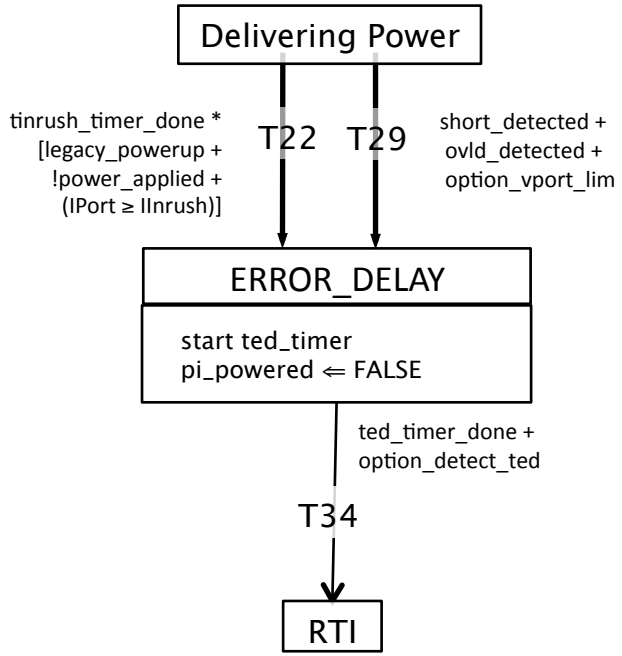
- PSE_Type = 2?
  - This term in the state diagram needs to be changed to address Type 3 and Type 4 PSEs.
  - Do we assume that the PSE Type will be Type 3 or Type 4, or else the Type 1 and Type 2 state diagram takes precedence?
- Currently, this state diagram is not intended to deprecate Type 1 and Type 2 state diagrams, so should we remove references to such, or leave them?
- PSE_Type = 1?
  - Do we remove this transition from the diagram?
- DLL_Enable?
  - In the case of a dual-signature PD, both PS controllers may set this variable.
  - Given that it's a single PSE, do we just let either controller set it when it gets to this location? (ie: It's a global variable that is set by either controller)
    - For single-signature PDs, only one controller is used, and would set it.
    - For dual-signature PDs, either controller would set it which is appropriate since one controller may not ever reach the Power_On state.



Type 3 and Type 4 PS Test State Diagram



Type 3 and Type 4 PS Power Denied State Diagram



Type 3 and Type 4 PS Fault State Diagram

Connection Check & Detection

The goal is to allow the widest range of implementation possible.

One can capture portions of data used for the "Detection" function during Connection Check and store them, then later when operating within the PS Control state machine, use that data in conjunction with other data captured during do_detection to complete the detection process. This would potentially speed up the process and ensures that a robust detection is completed no longer before power is applied than the current Type 1 and Type 2 state diagram.

OR

One can perform Connection Check and disregard the data captured after the decision is made to exit that state, and then when entering the DETECTION state, perform an existing do_detection function as currently implemented.

The former would be faster and reduce the number of steps required to complete detection.

Example Text for PD_Check

- ***Add new function do_PD_check as follows:***

do_PD_check

This function is to be used only for Type 3 and Type 4 PSEs and works in conjunction with connection check defined in Section 33.2.5.0 and determines whether a PD can accept power over a single alternative PI configuration or both at the same time. This function returns the following variables:

PD_alt: This variable indicates the type of PD signature is connected to the PI, with respect to 4-pair operation.

Values:

A: The PSE has determined PD appears capable of accepting power only on Alt-A.
B: The PSE has determined PD appears capable of accepting power only on Alt-B.
Both: The PSE has determined PD appears capable of accepting power on Alt-A and Alt-B
Invalid: The PD_check function has not determined a valid value.

A Discussion on Functions

A Function returns a value for a variable or set of variables.
A Function call does not necessarily mean that the actions required to get the variable values must take place at the time of (or after) the function call is made.
For example: do_connection_check returns a value to the variable Signature_Type.

To perform do_connection_check, a number of stimuli and measurements may occur in order to determine the value of Signature_Type.
The stimulus (to both pairsets) and resultant measurements MAY be used to capture values that can later be used for do_detection or do_PD_check analysis.

Later functions are not necessarily required to perform the same stimulus and measurements to complete the assessment of their variables, as long as the timing of the stimulus and measurement falls within an acceptable timeframe for making those measurements.

What does this mean? (Bottom Line)
Some of the stimulus and measurements made to conclude a valid detection *may* have been performed during Connection Check or PD Check if the implementation ensures those values are still "fresh enough" to ensure validity.
The order of the function calls determines the order of assessing values captured, not necessarily the order or timing of the measurements being made.

Summary

- I believe the proposed approach, having a higher level hierarchical block that performs Connection_Check and PD_check to direct operation, enables us to keep the existing PSE control state machine very close to the original Type 1 and Type 2 designs.
- Key questions remain about how to deal with local (pairset) variables vs global (PSE) variables within the document.
  - We might provide text describing all Type 3 and Type 4 pairset state diagram variables as local unless otherwise noted
  - We might modify their names to communicate loc vs glob
  - We should carefully assess what conditions may occur on a pairset that would require immediate transition up to the PSE_Disabled state.
    - Currently, only failure of both pairsets to detect, or failure to enter detection within tcc2det leads to this transition.
    - Pairset faults only return the pairset controller to its own IDLE state. They do not return the entire PSE to its Disabled state.
- While review additional work is required, I recommend that the Task Force adopt the updated state diagrams, incorporate them into FrameMaker format, and use this as the basis for further review & advancement of the draft.

Q&A