800GBASE-LR1 & ER1 support of APSU with refactored FSMs

Addresses comment 140 against D3.0

Marco Mascitto, Nokia
Luz Osorio, Nokia
Tom Huber, Nokia

Supporters

- Matt Brown (Qualcomm)
- Mike Dudek (Marvell)
- Gary Nicholl (Cisco)
- Sebastien Gareau (Ciena)
- Jeffery Maki (HPE)

Overview

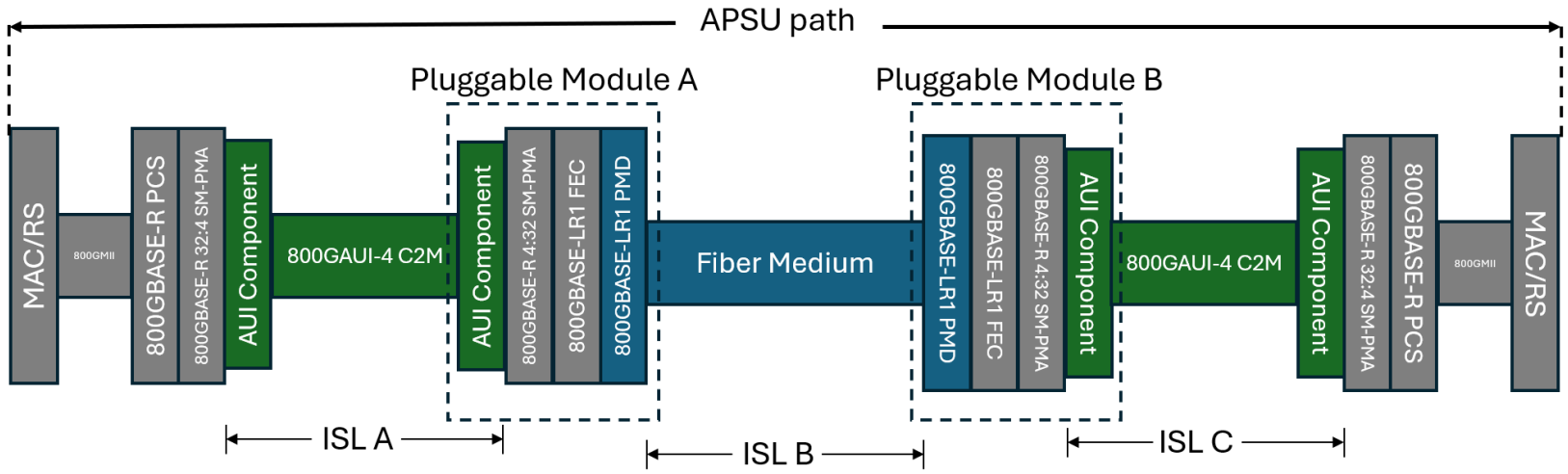
- The presentation explores how one might map the functionality described in brown_178b_01_260422 (for 800GBASE-LR1) and nicholl_178b_01_260422 (for 800GBASE-ER1) if the APSU FSMs were refactored according to huber_178b_01_260414.
- Key postulates in the refactoring proposal:
  - “Training” is PMD specific and means different things for different PMDs. Not all PMDs need to be trained, but all need to support APSU.
  - The refactored ILT FSM only provides the functionality to train lanes of an interface and report this status to the RTS FSM.
    - There is one instance of this ILT FSM per lane.
    - It does not switch the transmit mode of a lane.
  - The refactored RTS FSM is wholly responsible for setting the transmit mode and includes a new tx_mode <= RTS, allowing the transmitter to signal RTS
    - Signaling RTS is PMD specific and is a different signal for different PMDs.

Signaling RTS with refactored FSMs

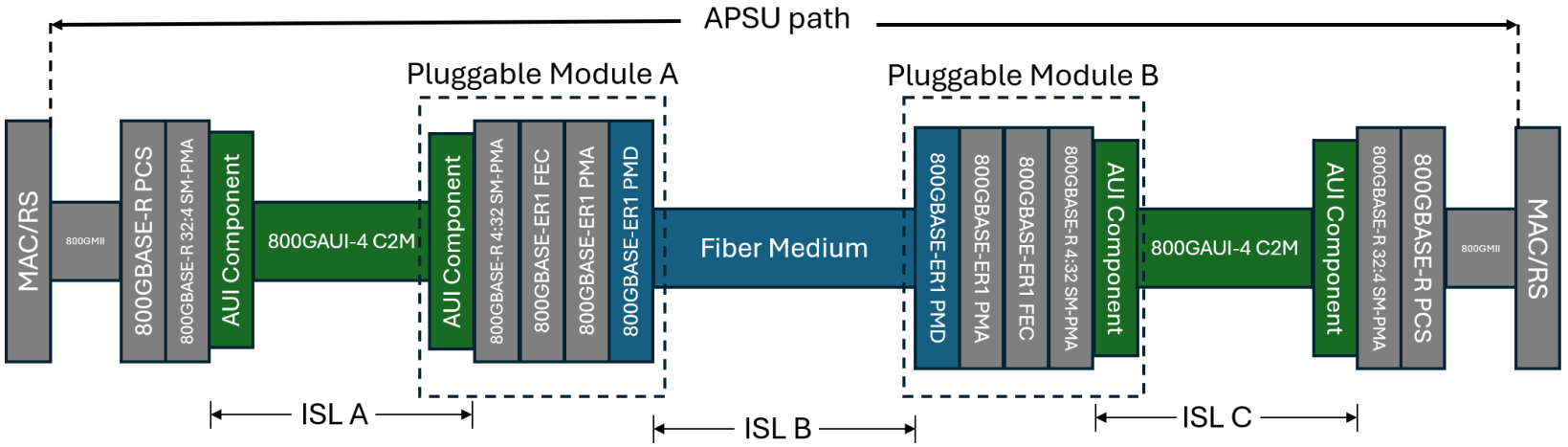
- The signal used to represent RTS is now interface specific.
- A tx_mode <= RTS is how the RTS signal is forwarded from an interface to its peer interface.
- Those interfaces that make use of the E1 format or the O1 format use the CT bit to signal RTS (i.e., nothing changes for these).
- The LR1 and ER1 PMDs don’t train and don’t have associated training frame formats, so there is no CT bit to exploit. The current proposal is:
  - LR1 represents RTS by unsquelching the transmitter.
  - ER1 represents RTS with a specific MNT value in the CSTAT octet.
- These coherent PMDs don’t train, so there is no instantiation of the ILT FSM per lane (or per interface) required.
- These coherent PMDs need to support APSU, so there is an instance of the RTS FSM per interface.

Test the new paradigm: APSU should complete in these two applications with **only RTS FSMs instantiated on the blue ISLs**. Green ISLs have both ILT and RTS FSMs.

800GBASE-LR1 application



800GBASE-ER1/ER1-20 application

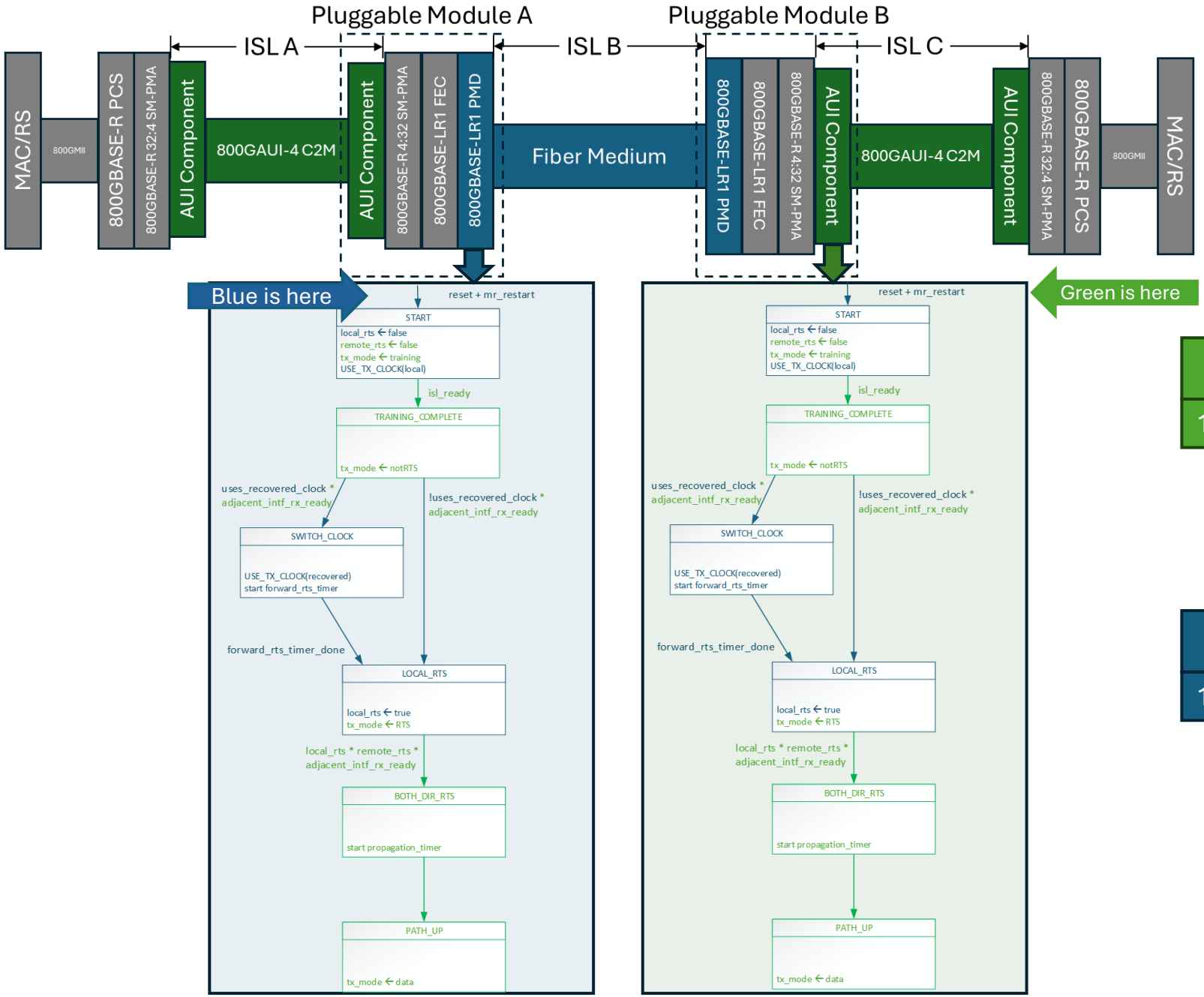


From mascitto_3dj_01_2603

800GBASE-LR1

Supporting APSU with refactored FSMs

reset or mr_restart



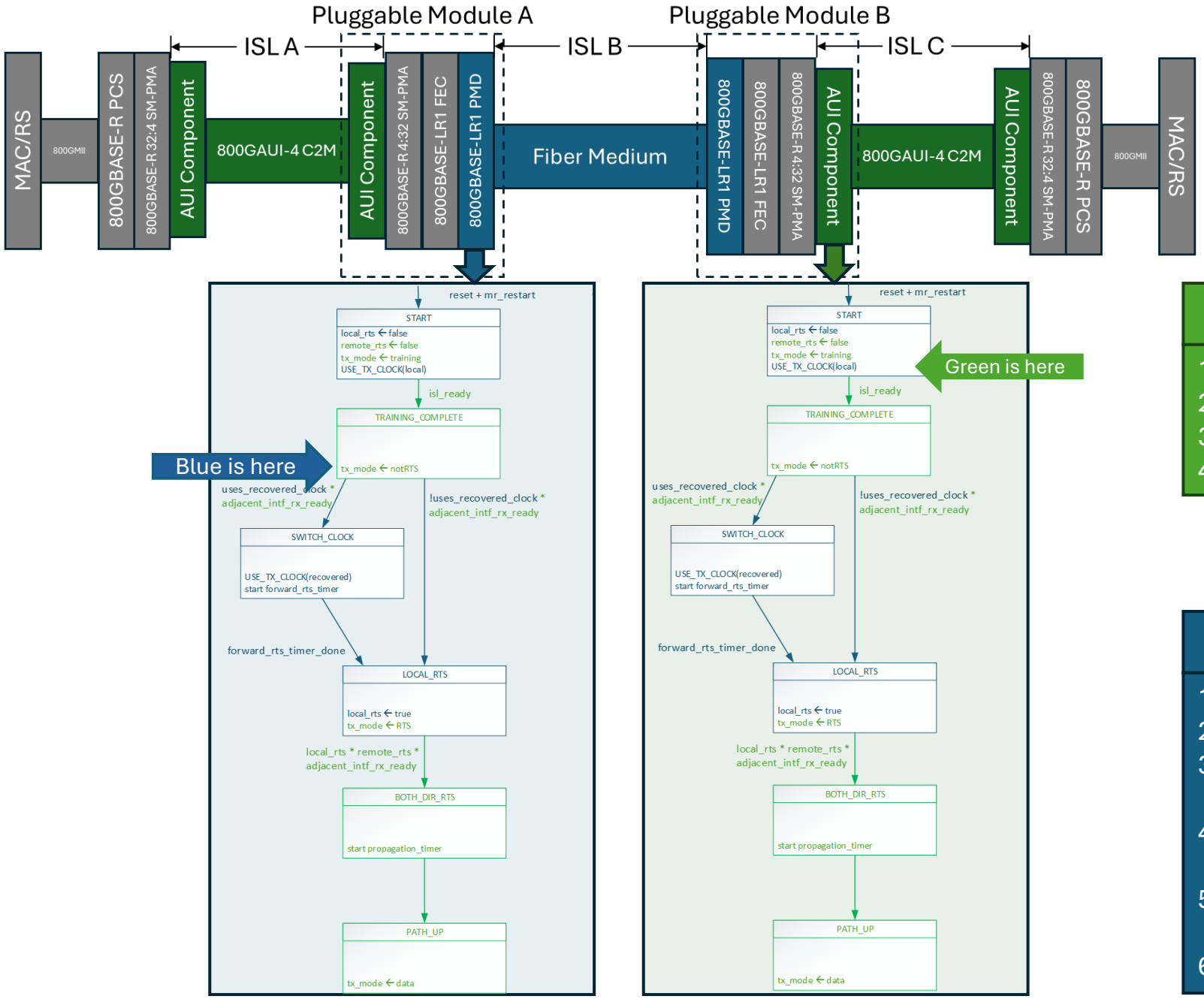
Green Interface

1. Stay here unit reset and mr_restart are false

Blue Interface

1. Stay here until reset and mr_restart are false

Start training

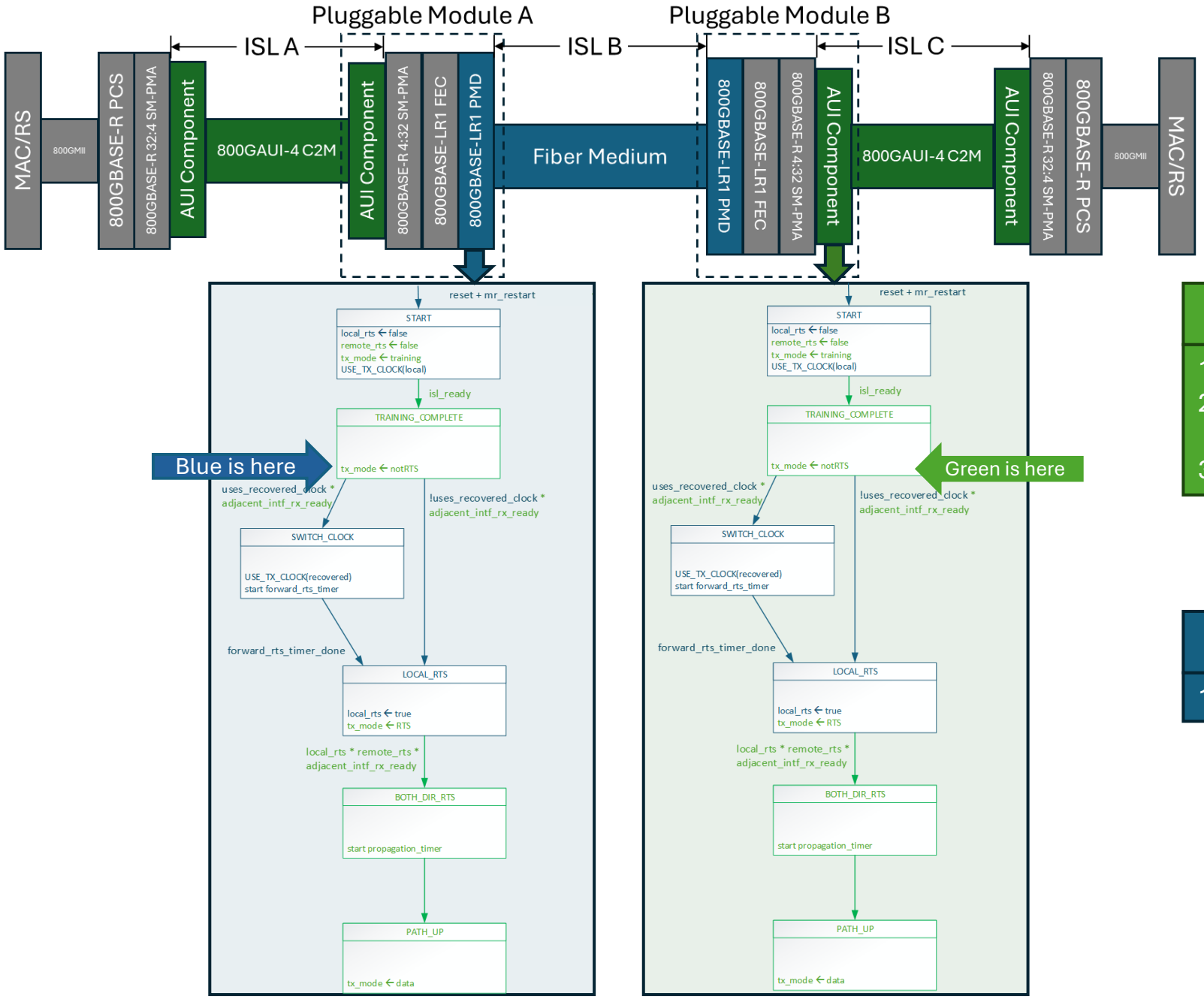


Green Interface

1. Set local_rts and remote_rts to false
2. Use local clock
3. Change tx_mode to training
4. Stay here until isl_ready

Blue Interface

1. Set local_rts and remote_rts to false
2. Use local clock
3. Set tx_mode <= training, which is equivalent to tx_disable <= true
4. Isl_ready implicitly always true, so transition to TRAINING_COMPLETE
5. Set tx_mode <= !RTS, which is equivalent to tx_disable <= true
6. Stay here until adjacent_intf_rx_ready



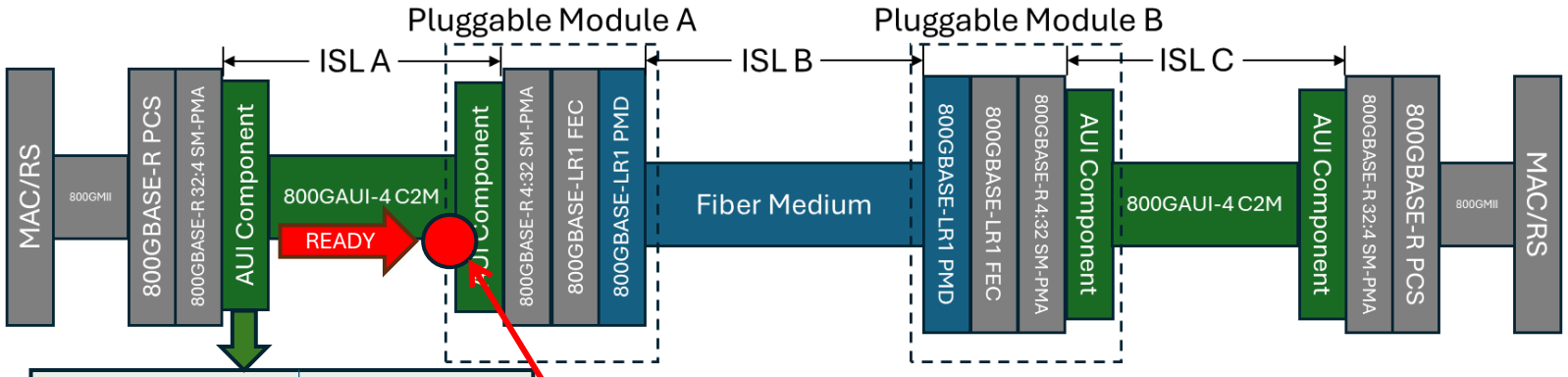
Green isl_ready

Green Interface

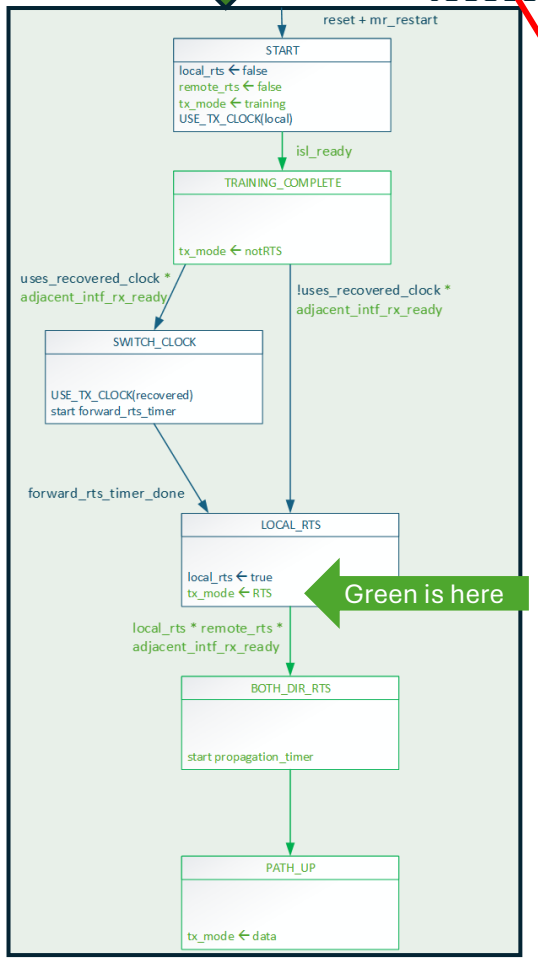
1. Transition to TRAINING_COMPLETE
2. Set tx_mode <= !RTS, which means keep sending E1 frames on local clock with CT bit set to 1
3. Stay here until adjacent_intf_rx_ready

Blue Interface

1. Stay here until adjacent_intf_rx_ready



Meanwhile, at ISL A…



ISL A has finished training and transitions to the LOCAL_RTS state

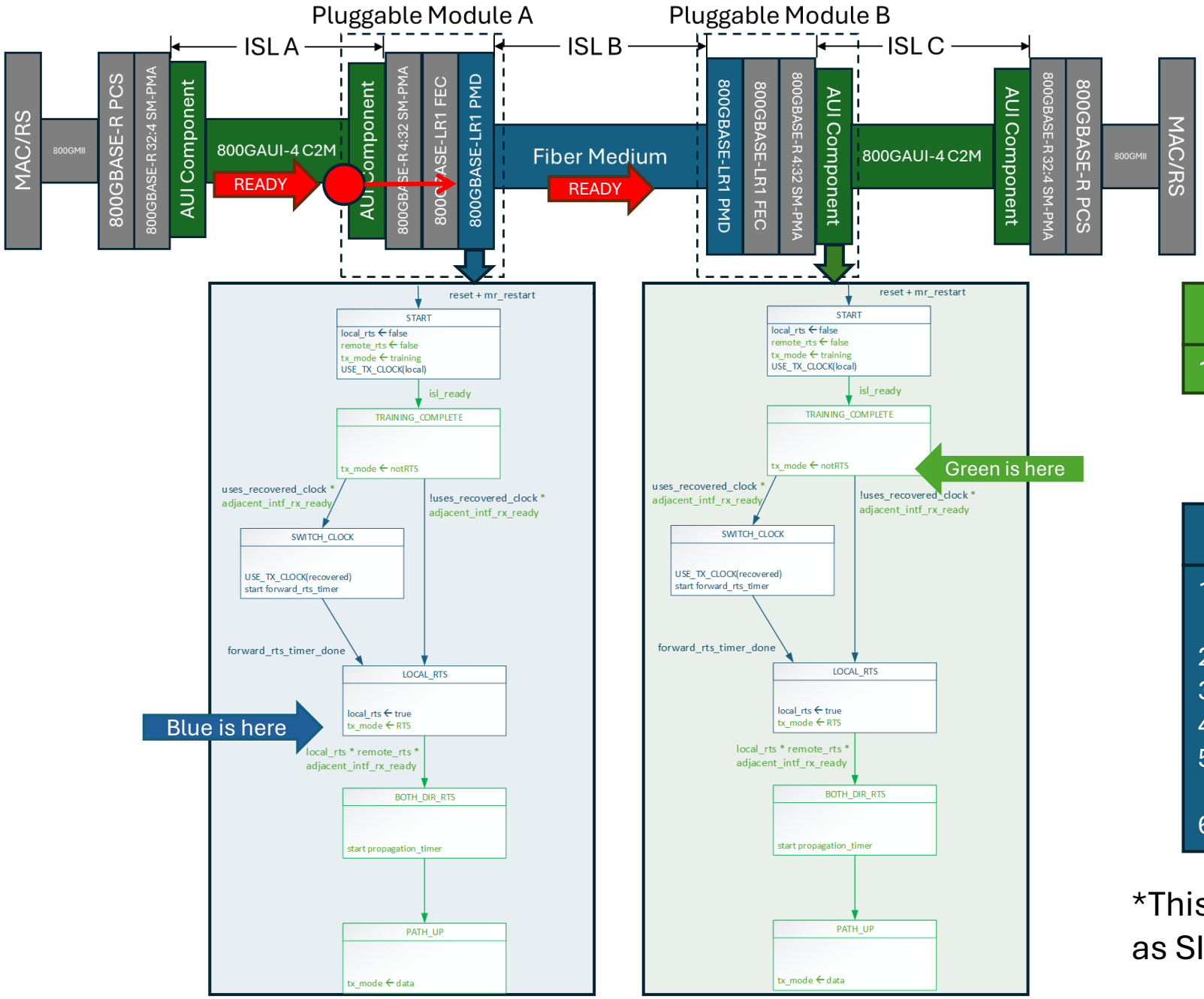
Green Interface

1. TRAINING_COMPLETE
2. Set tx_mode <= !RTS, which means keep sending E1 frames on local clock with CT bit set to 1
3. This interface !uses_recovered_clock and its adjacent_intf_rx_ready is always true
4. Transitions to LOCAL_RTS
5. Set local_rts <= true
6. Sets tx_mode <= RTS, which means send E1 frames with CT = 0 using mission clock
7. Stay here until remote_rts

This AUI component's receiver now seeing E1 frames with CT = 0 (so it has remote_rts), which is the implicit signal for SIGNAL_OK parameter = READY

(i.e., E1 frames with CT = 0 AND local_tf_lock means service interface can signal READY)

Blue interface READY



Green Interface

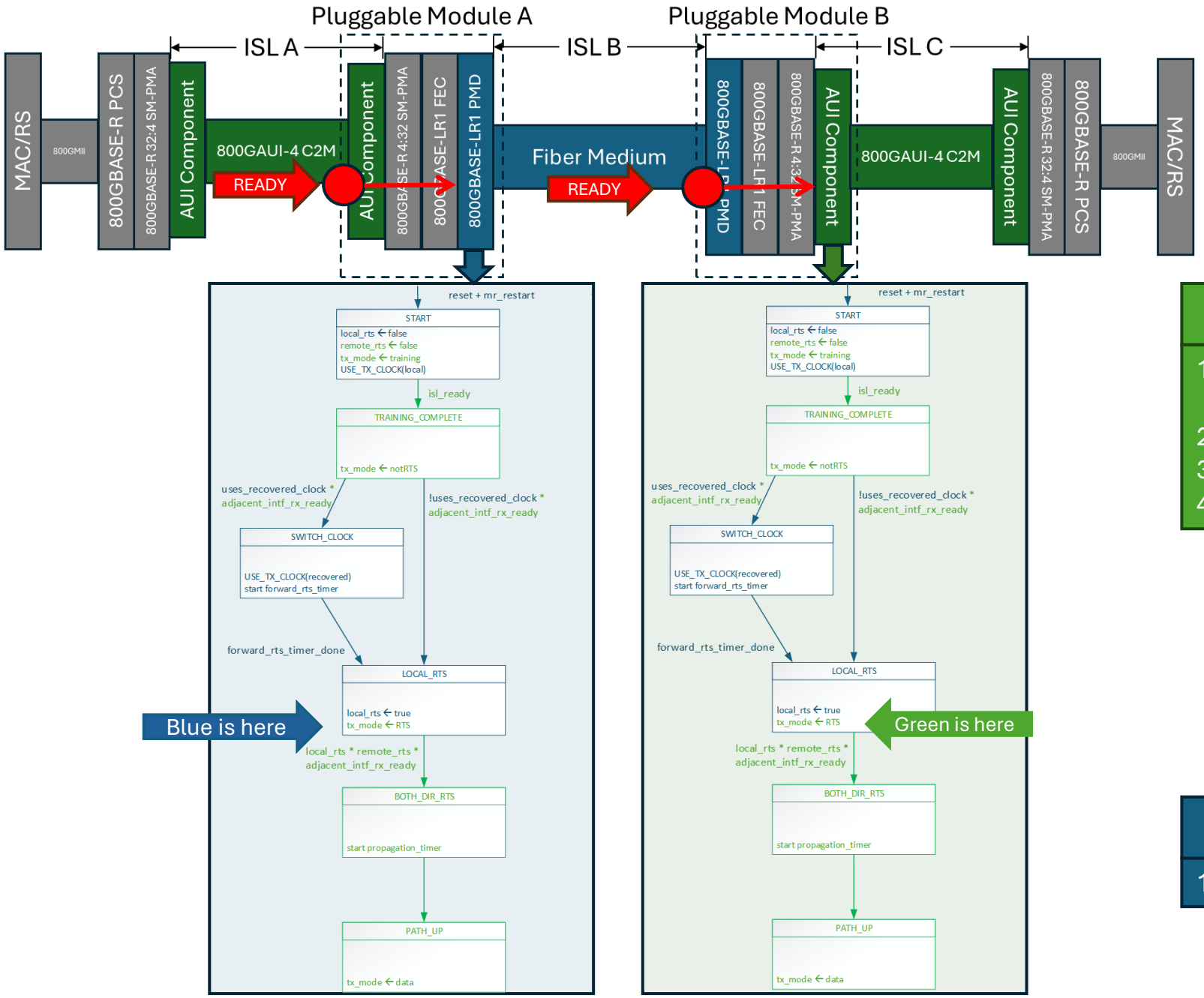
1. Stay here until adjacent_intf_rx_ready

Blue Interface

1. SIGNAL_OK parameter = READY, therefore adjacent_intf_rx_ready
2. Switch to using the mission clock
3. Wait until forward_rts_timer expires
4. Set local_rts <= true
5. Set the tx_mode <= RTS, which means unsquelch and send local_pattern* using mission clock
6. Stay here until remote_rts

*This PMD interprets reception of local_pattern as SIGNAL_OK = READY

Green interface READY

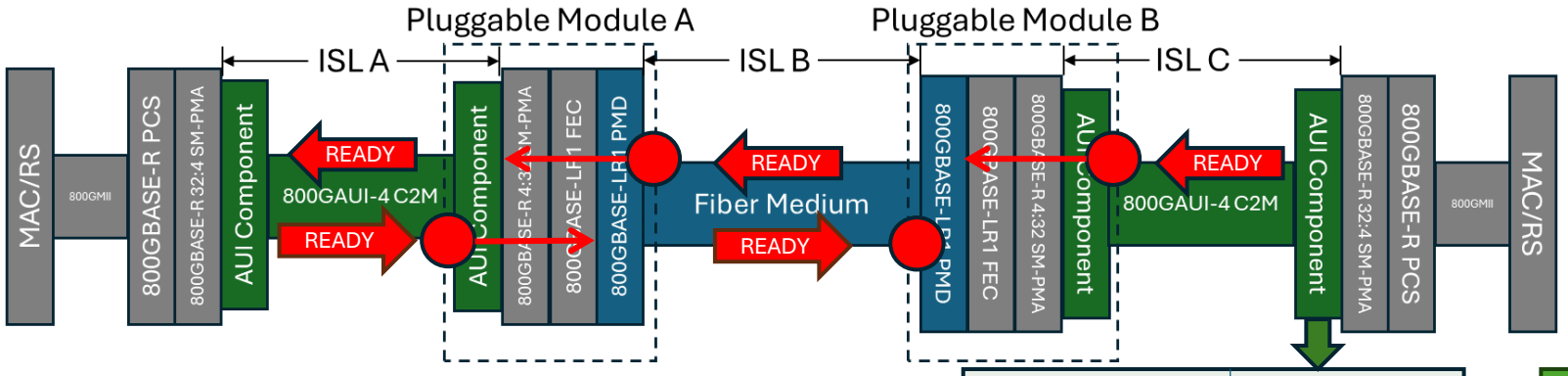


Green Interface

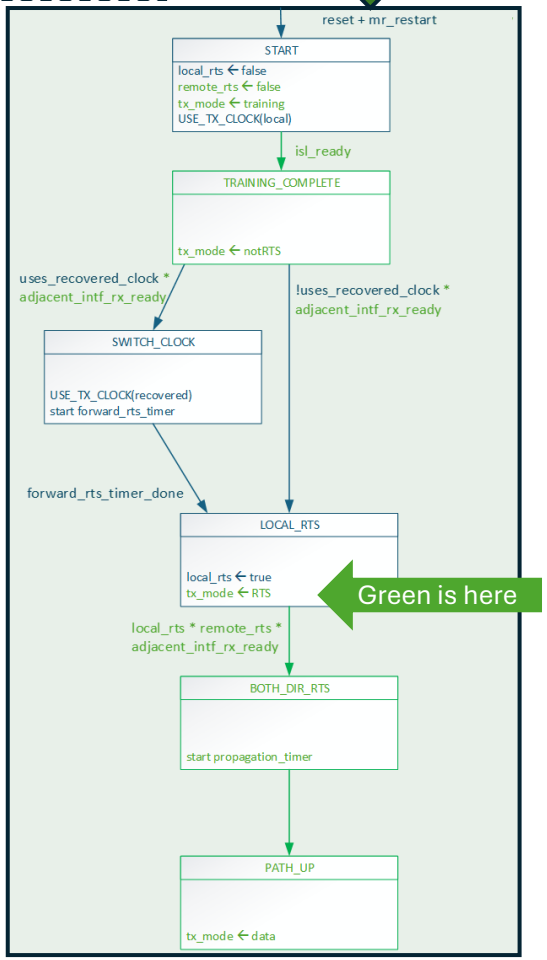
1. SIGNAL_OK parameter = READY, therefore adjacent_intf_rx_ready
2. Switch to using the mission clock
3. Wait until forward_rts_timer expires
4. Transition to LOCAL_RTS

Blue Interface

1. Stay here until remote_rts



*This allows ISL B to enter the READY state, which in turn allows ISL A to enter the READY state

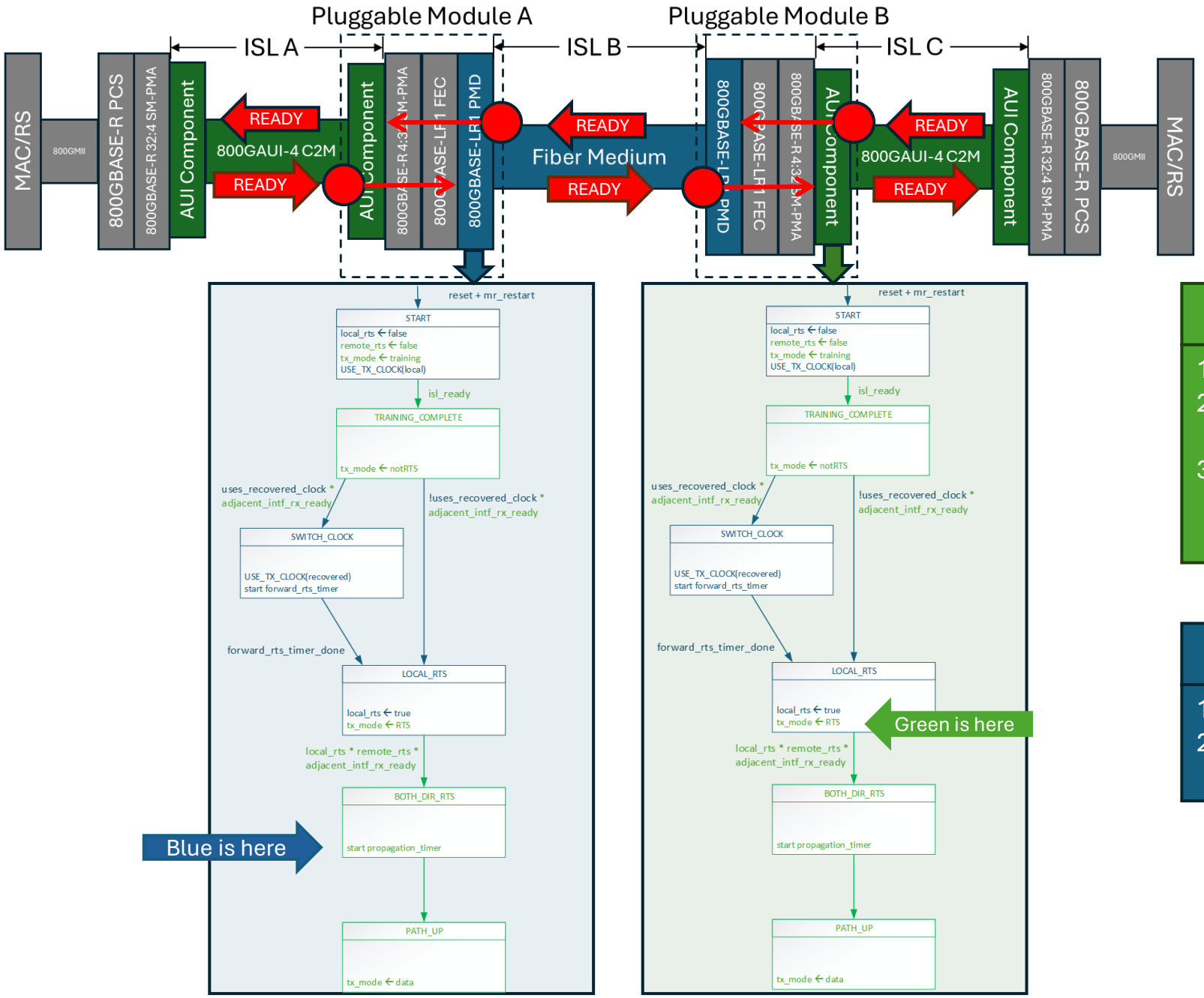


Meanwhile, at ISL C…

Green Interface

1. TRAINING_COMPLETE
2. Set tx_mode <= !RTS, which means keep sending E1 frames on local clock with CT bit set to 1
3. This interface !uses_recovered_clock and its adjacent_intf_rx_ready is always true
4. Transitions to LOCAL_RTS
5. Set local_rts <= true
6. Sets tx_mode <= RTS, which means send E1 frames with CT = 0 using mission clock*

Green interface forwards RTS



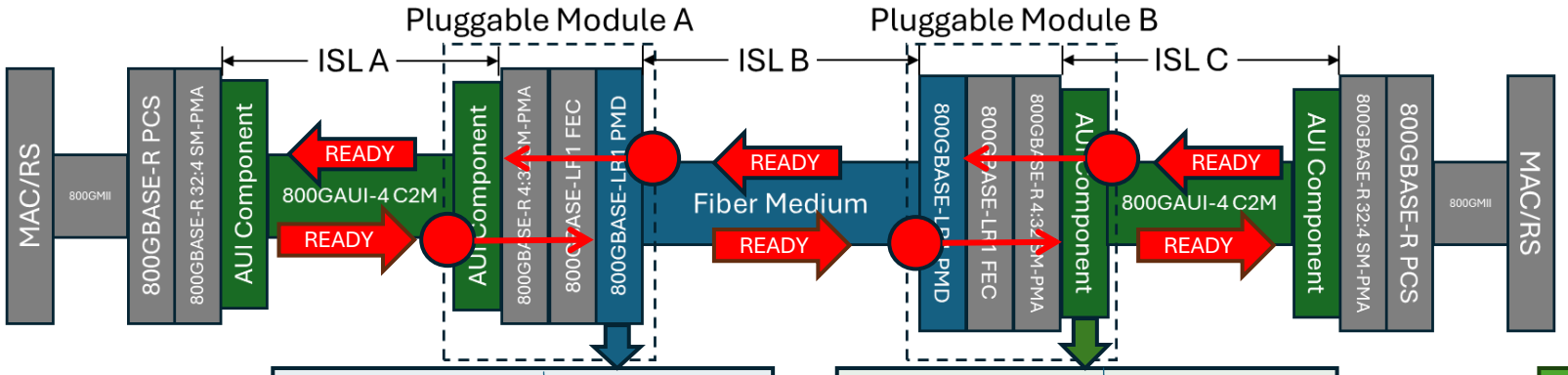
Green Interface

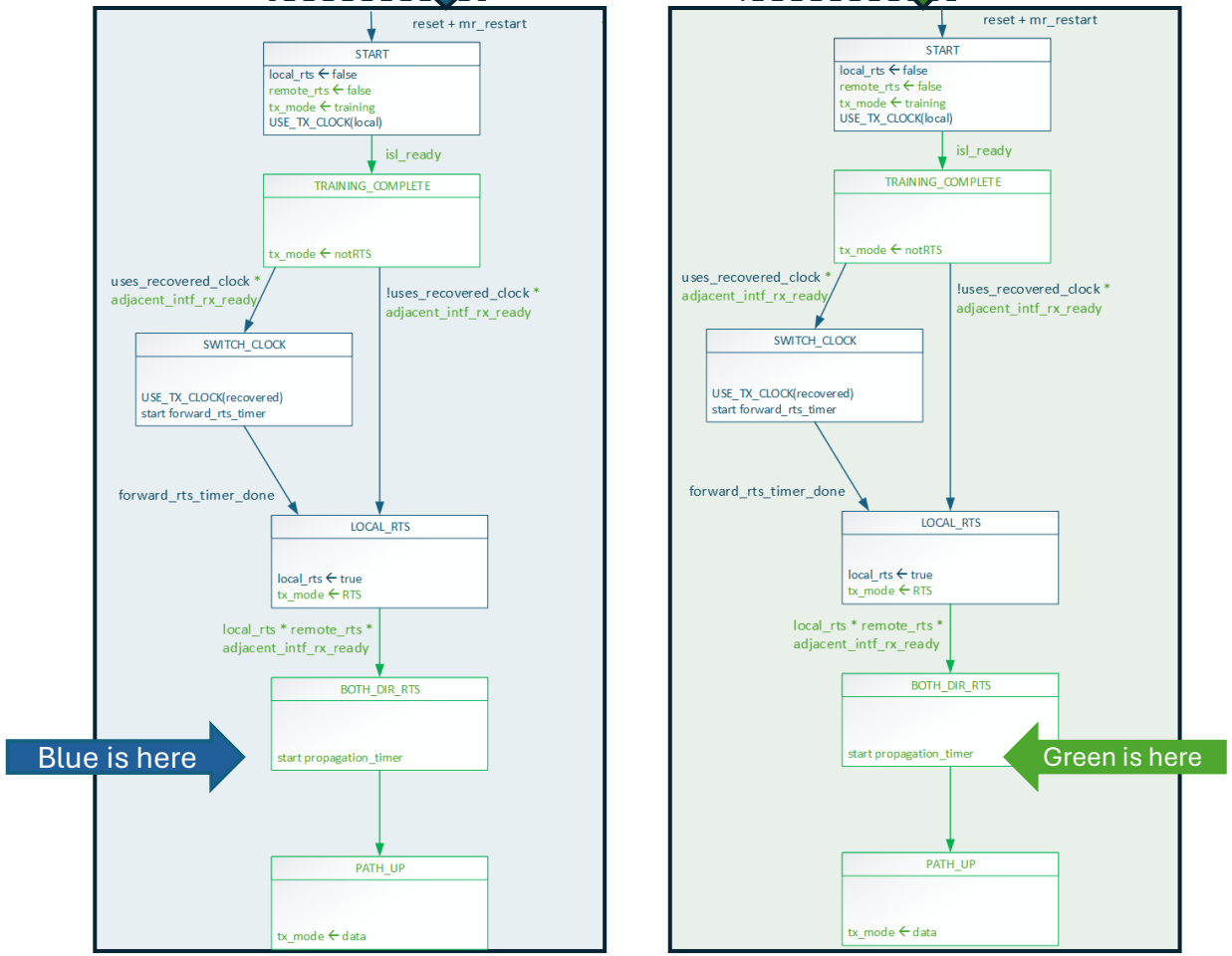
1. Set local_rts <= true
2. Set tx_mode <= RTS, which means send E1 frames with CT = 0 using mission clock
3. Peer interface receives RTS and transitions to BOTH_DIR_RTS state

Blue Interface

1. Both local_rts and remote_rts are true
2. Transition to BOTH_DIR_RTS state and wait until the propagation_timer expires

Green interface BOTH_DIR_RTS



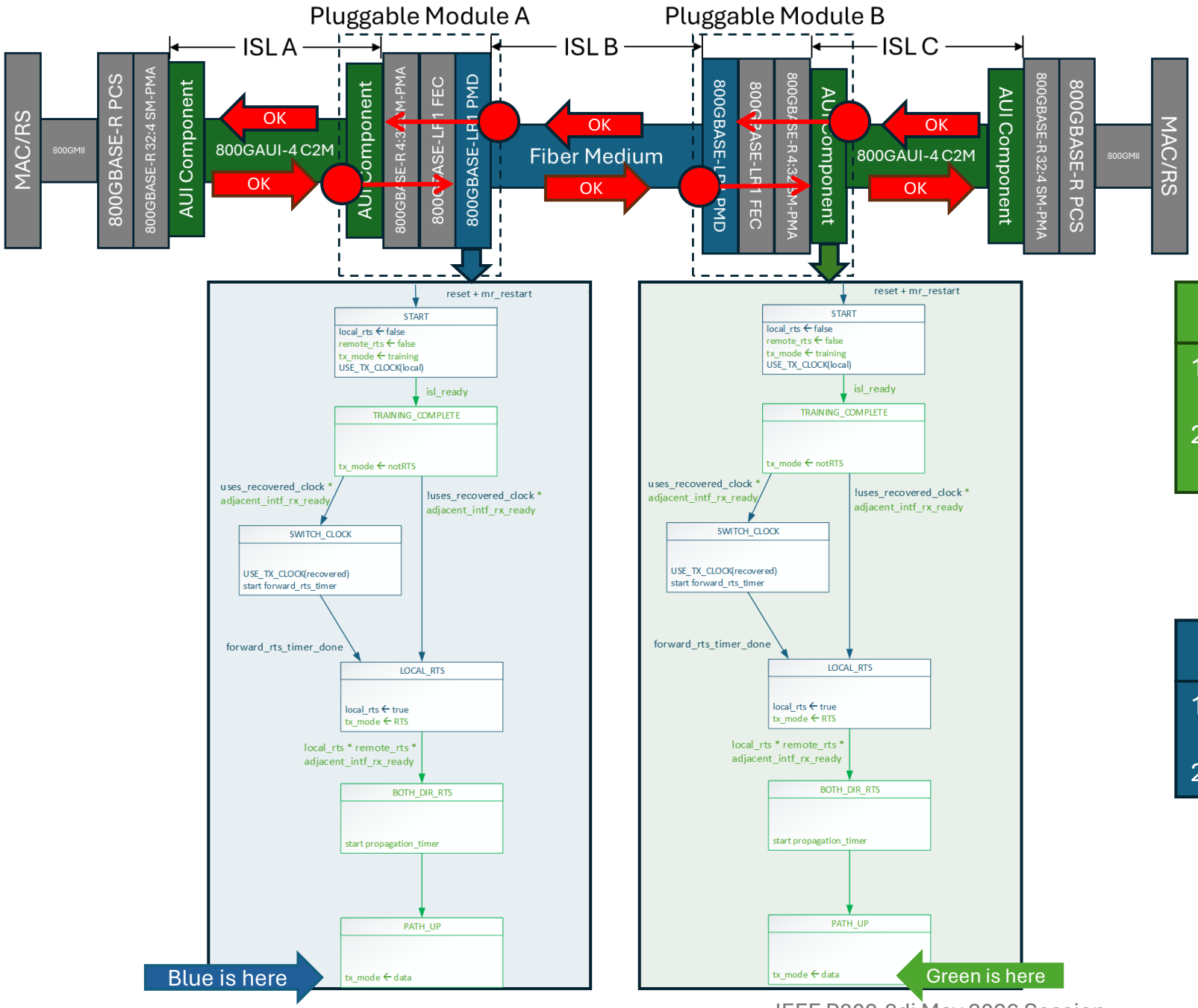


Green Interface

1. Both local_rts and remote_rts are true
2. Transition to BOTH_DIR_RTS state and wait until the propagation_timer expires

Blue Interface

1. Waiting until propagation_timer expires



Tx_mode <= data

Green Interface

1. Transitions to PATH_UP state after propagation timer expires
2. Sets tx_mode <= data

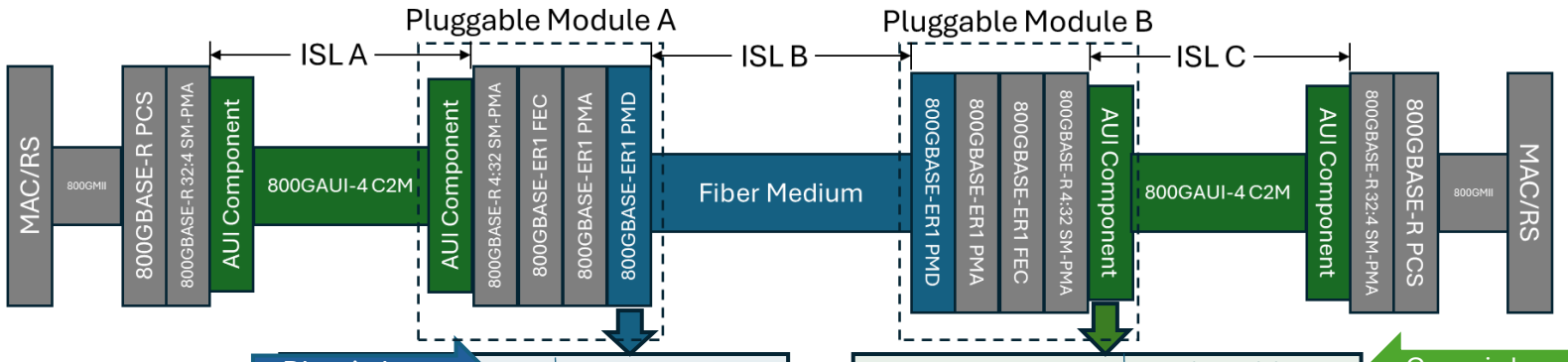
Blue Interface

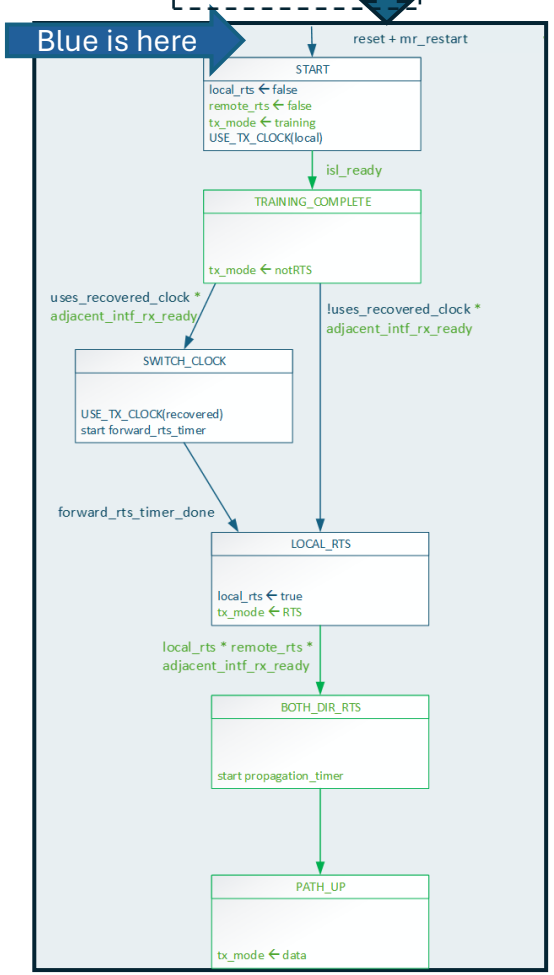
1. Transitions to PATH_UP state after propagation timer expires
2. Sets tx_mode <= data

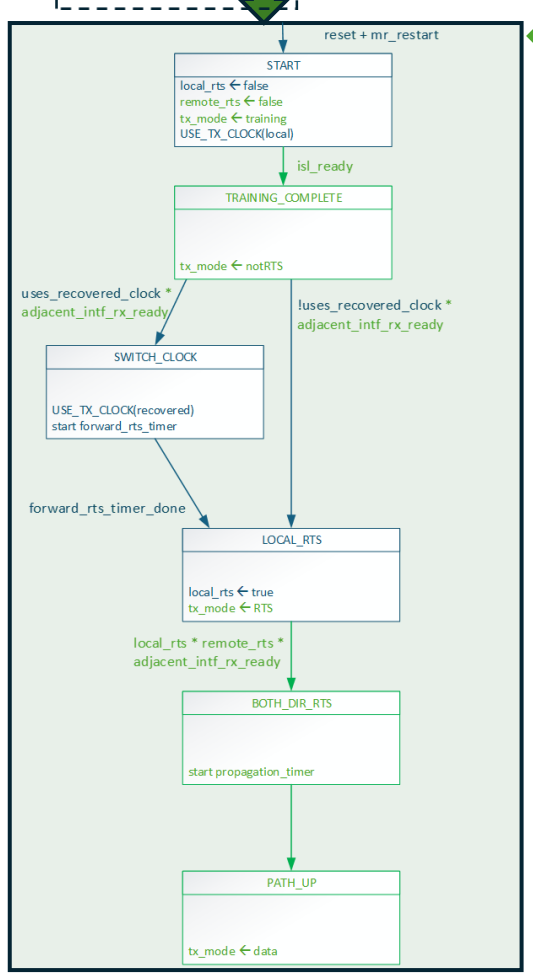
800GBASE-ER1

Supporting APSU with refactored FSMs

reset or mr_restart







Green is here

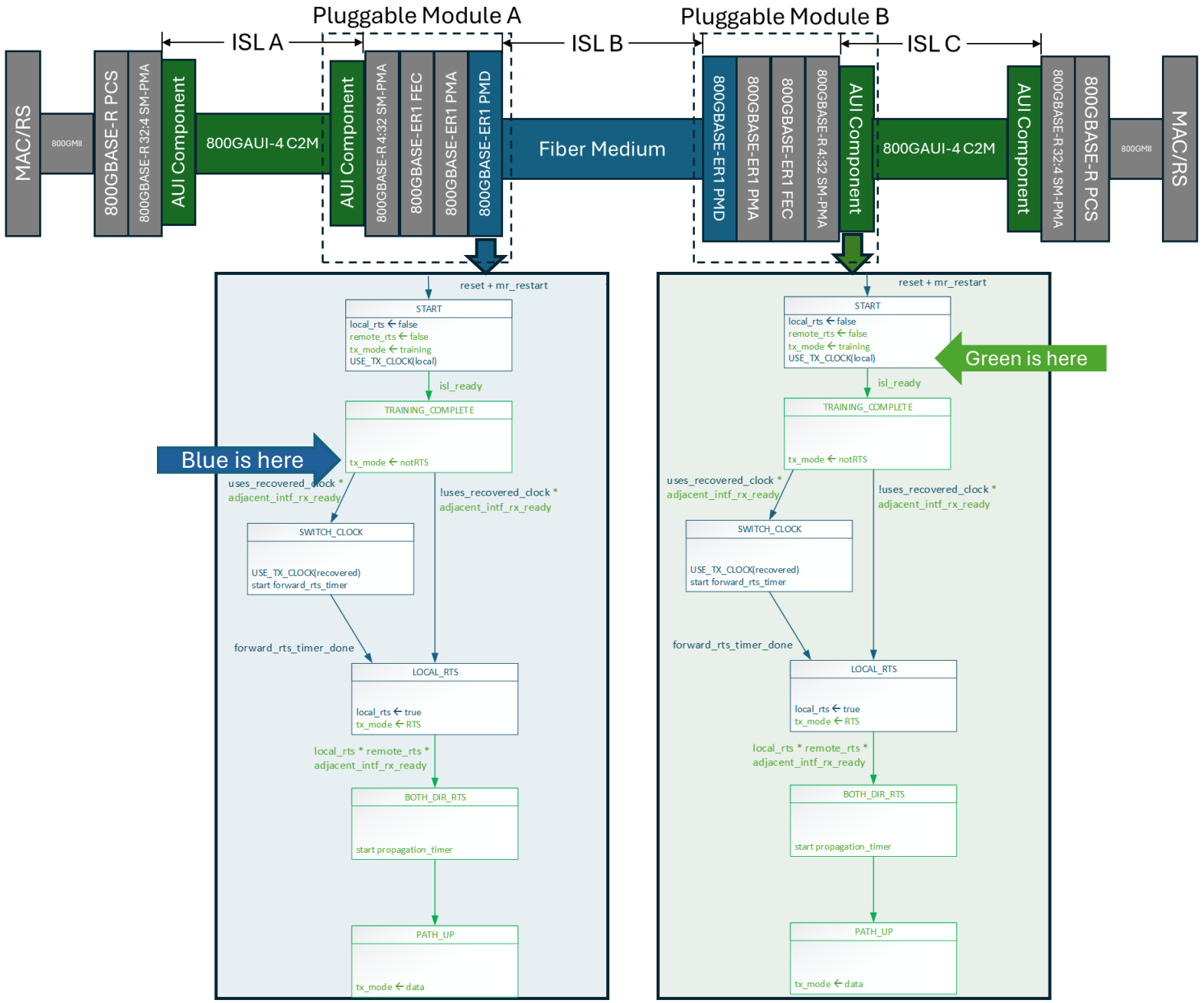
Green Interface

1. Stay here unit reset and mr_restart are false

Blue Interface

1. Stay here until reset and mr_restart are false

Start training

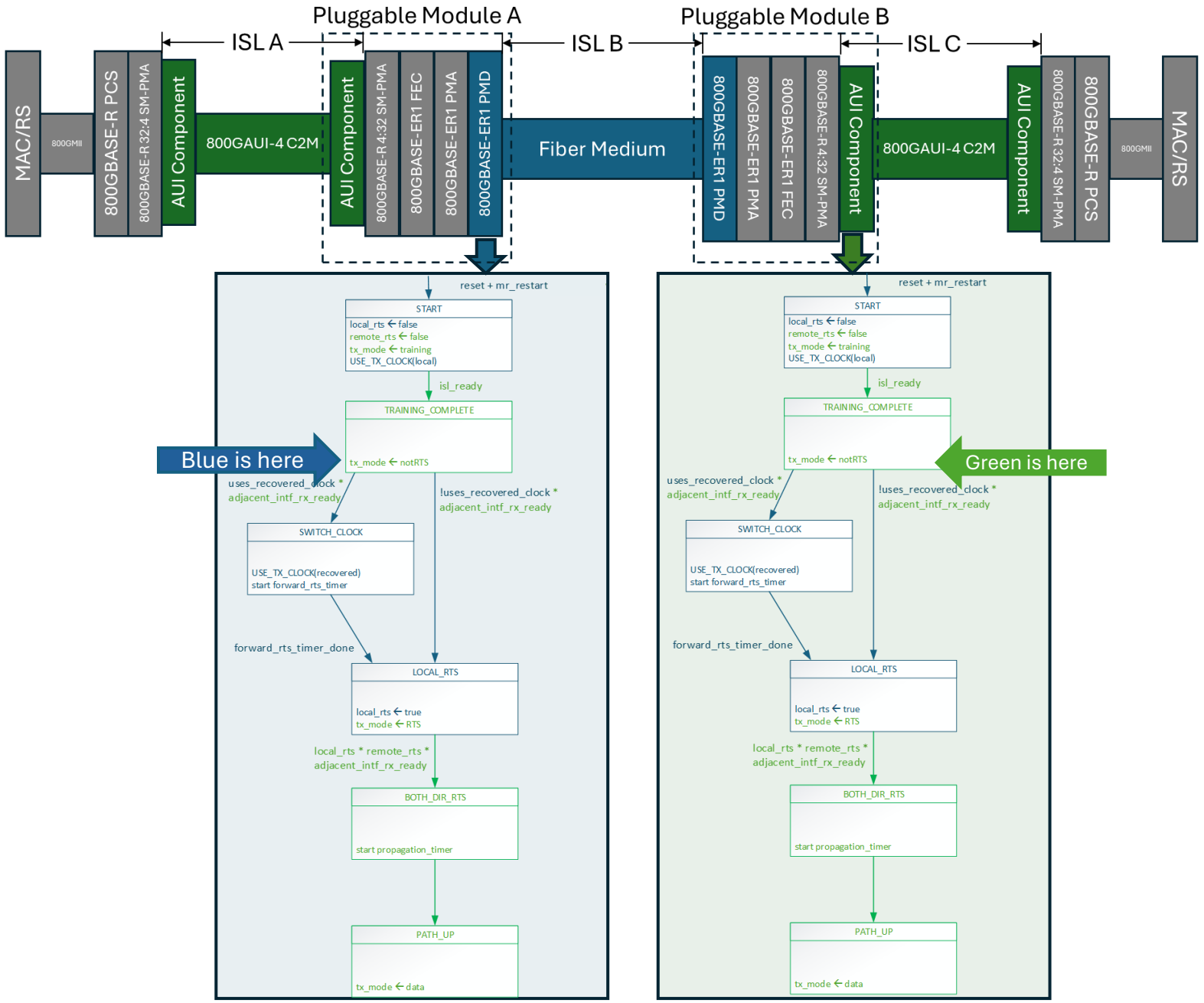


Green Interface

1. Set local_rts and remote_rts to false
2. Use local clock
3. Change tx_mode to training
4. Stay here until isl_ready

Blue Interface

1. Set local_rts and remote_rts to false
2. Use local clock
3. Set tx_mode <= training, which sets the MNT bits to 010
4. Isl_ready implicitly always true, so transition to TRAINING_COMPLETE
5. Set tx_mode <= !RTS, which sets the MNT bits to 010 (IN_PROGRESS)
6. Stay here until adjacent_intf_rx_ready



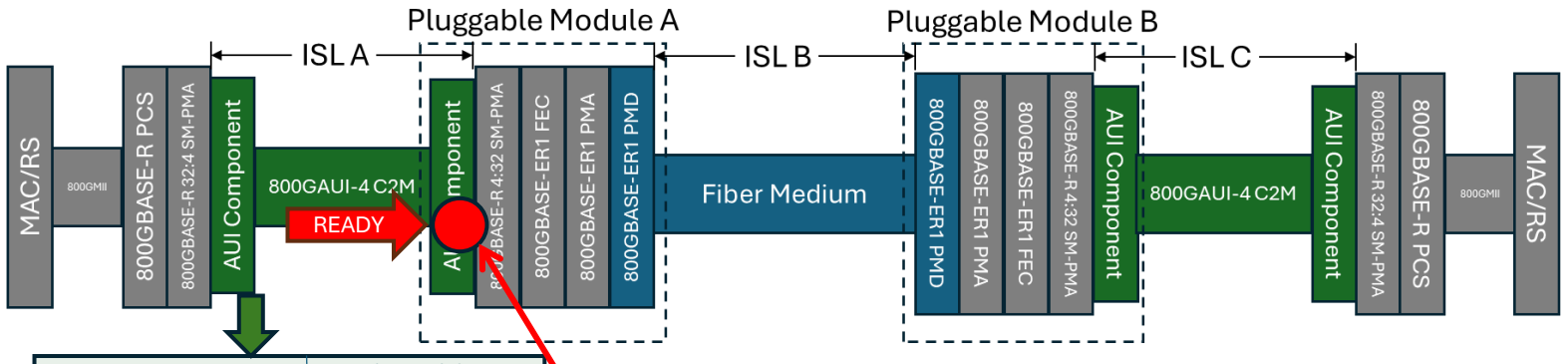
Green isl_ready

Green Interface

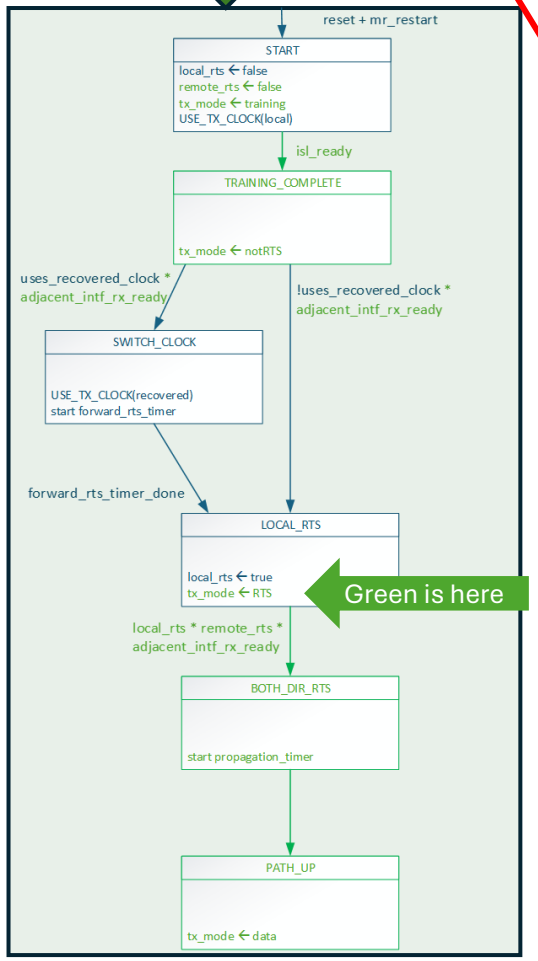
1. Transition to TRAINING_COMPLETE
2. Set tx_mode <= !RTS, which means keep sending E1 frames on local clock with CT bit set to 1
3. Stay here until adjacent_intf_rx_ready

Blue Interface

1. Stay here until adjacent_intf_rx_ready



Meanwhile, at ISL A…



ISL A has finished training and transitions to the LOCAL_RTS state

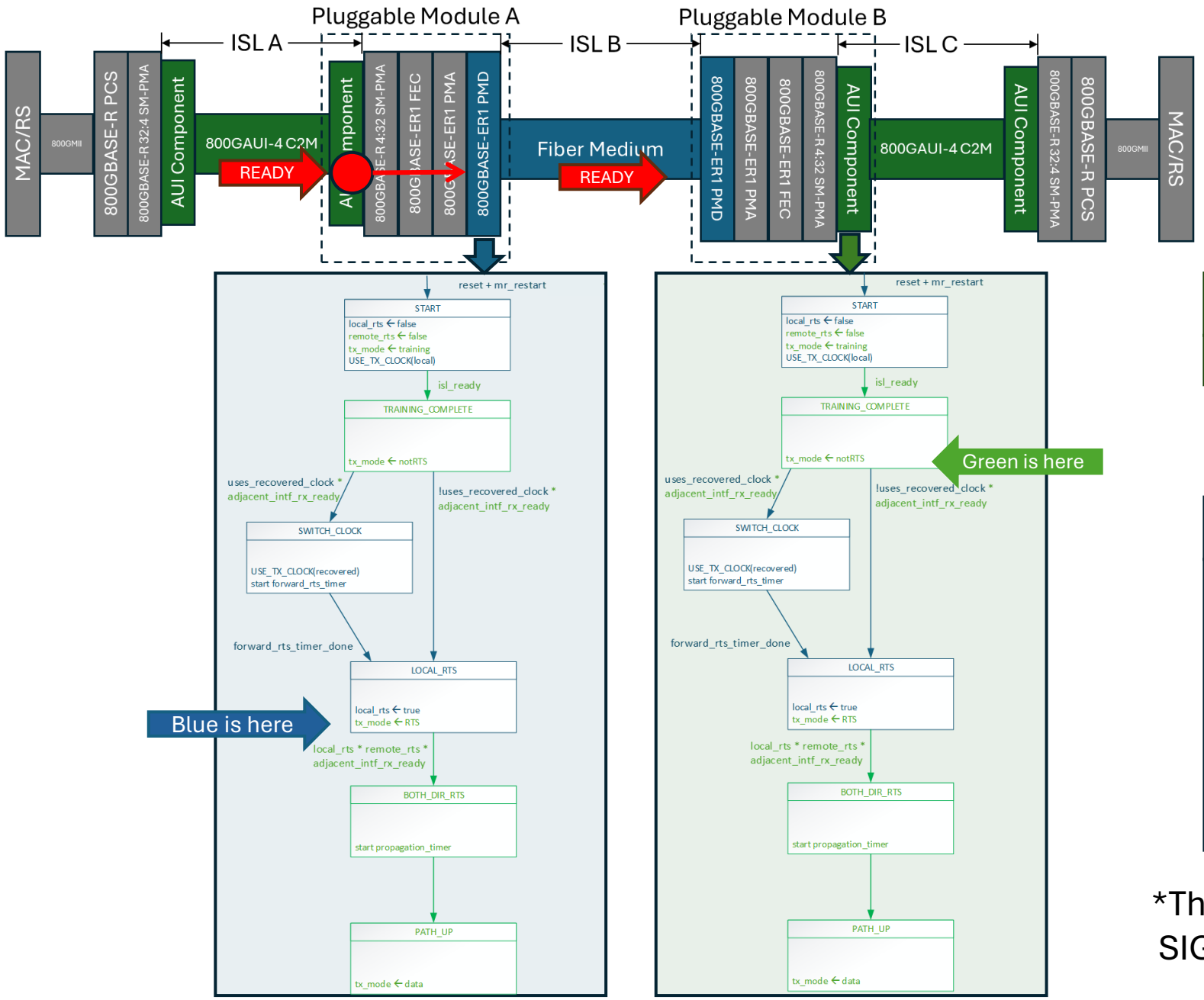
Green Interface

1. TRAINING_COMPLETE
2. Set tx_mode <= !RTS, which means keep sending E1 frames on local clock with CT bit set to 1
3. This interface !uses_recovered_clock and its adjacent_intf_rx_ready is always true
4. Transitions to LOCAL_RTS
5. Set local_rts <= true
6. Sets tx_mode <= RTS, which means send E1 frames with CT = 0 using mission clock
7. Stay here until remote_rts

This AUI component's receiver now seeing E1 frames with CT = 0 (so it has remote_rts), which is the implicit signal for SIGNAL_OK parameter = READY

(i.e., E1 frames with CT = 0 AND local_tf_lock means service interface can signal READY)

Blue interface READY



Green Interface

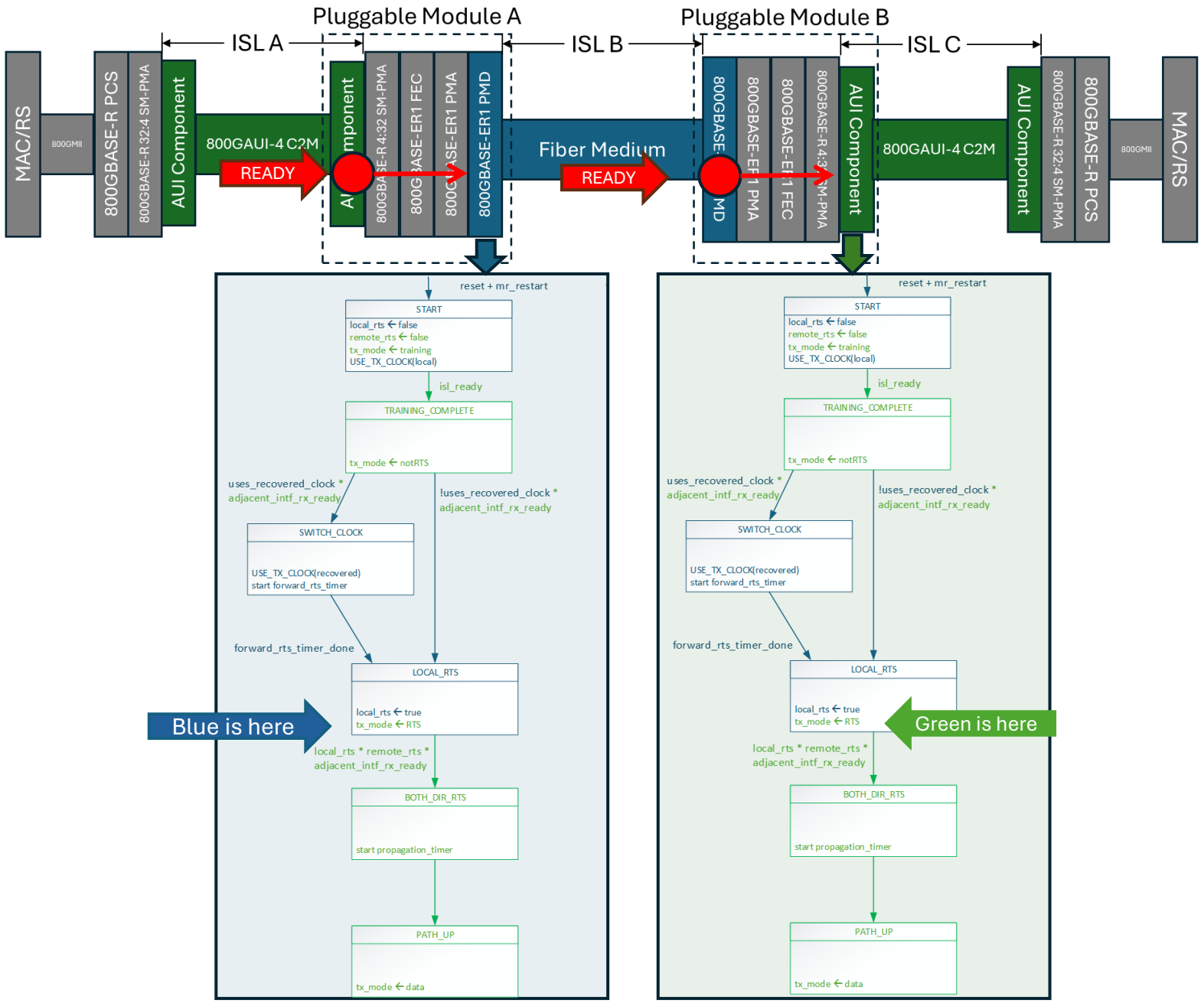
1. Stay here until adjacent_intf_rx_ready

Blue Interface

1. SIGNAL_OK parameter = READY, therefore adjacent_intf_rx_ready
2. Switch to using the mission clock
3. Wait until forward_rts_timer expires
4. Set local_rts <= true
5. Set the tx_mode <= RTS, which sets the MNT bits to 001 (READY)
6. Stay here until remote_rts

*This PMD interprets reception of MNT = 001 as SIGNAL_OK = READY

Green interface READY

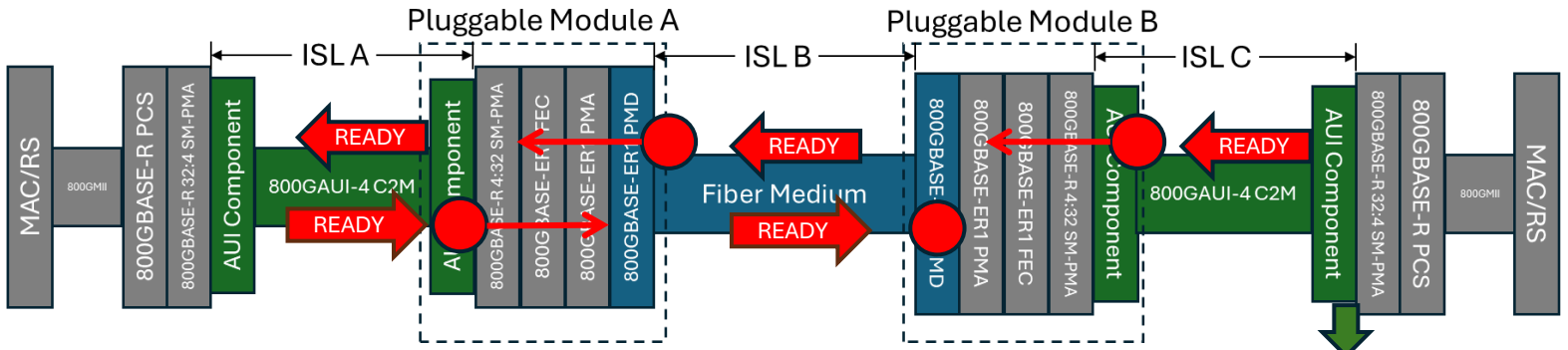


Green Interface

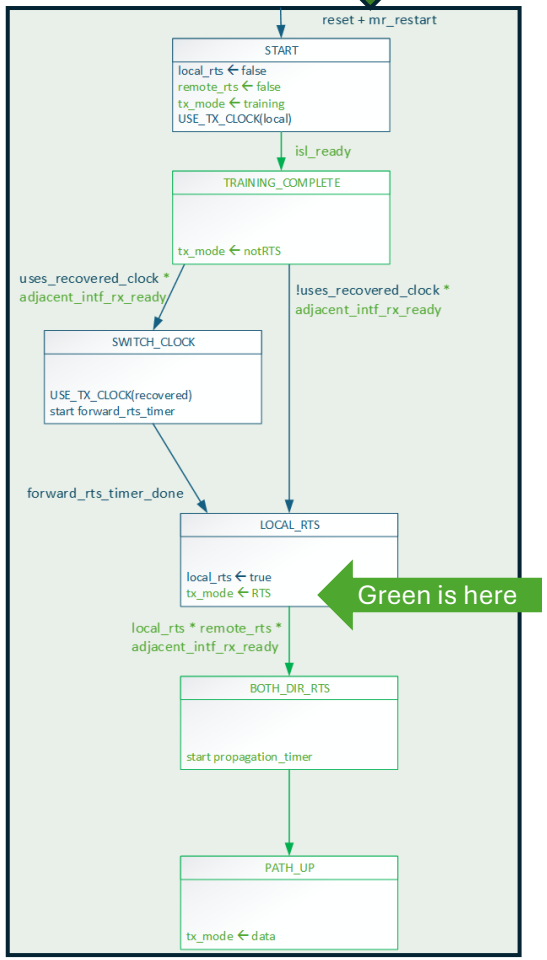
1. SIGNAL_OK parameter = READY, therefore adjacent_intf_rx_ready
2. Switch to using the mission clock
3. Wait until forward_rts_timer expires
4. Transition to LOCAL_RTS

Blue Interface

1. Stay here until remote_rts



*This allows ISL B to enter the READY state, which in turn allows ISL A to enter the READY state

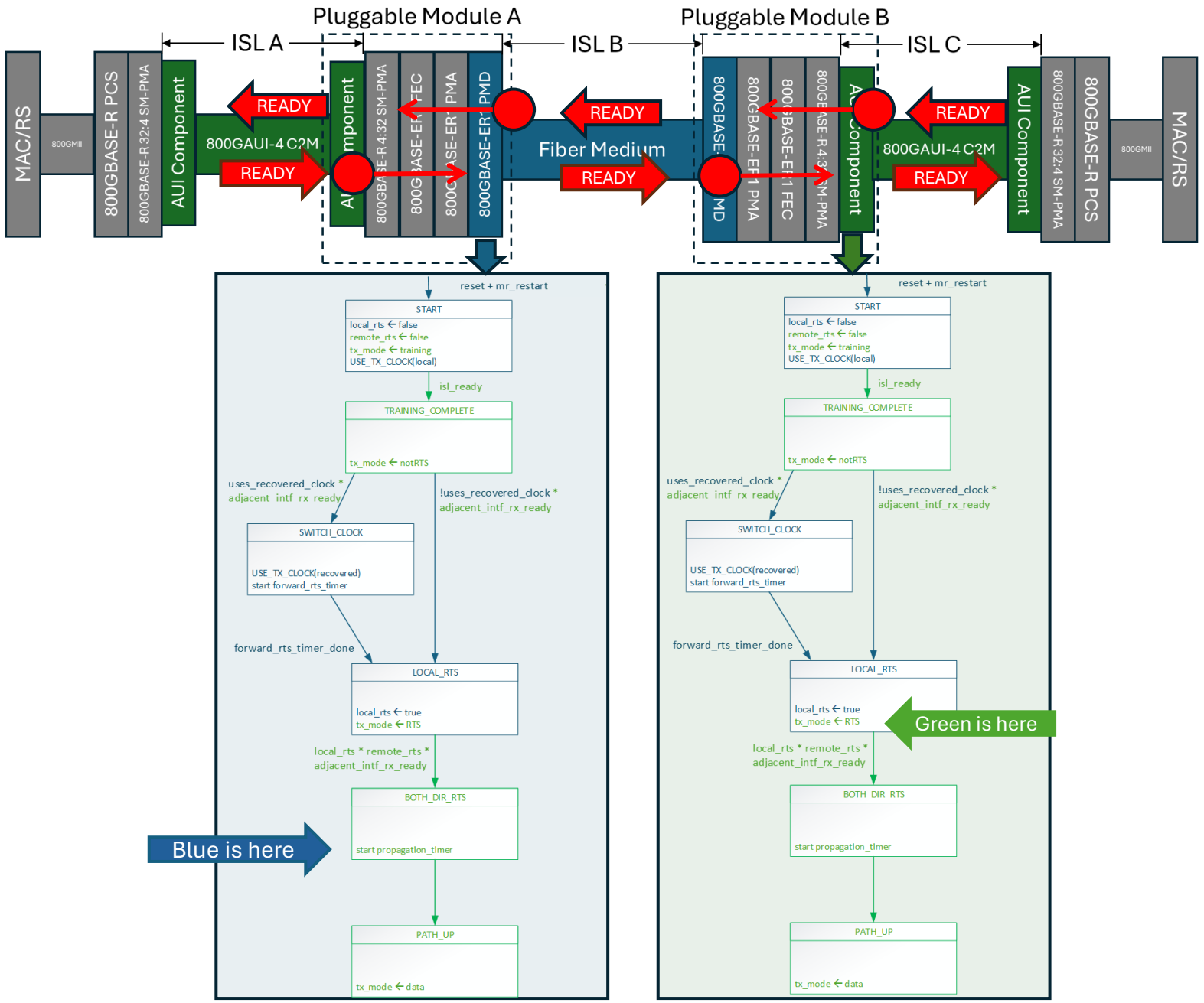


Meanwhile, at ISL C…

Green Interface

1. TRAINING_COMPLETE
2. Set tx_mode <= !RTS, which means keep sending E1 frames on local clock with CT bit set to 1
3. This interface !uses_recovered_clock and its adjacent_intf_rx_ready is always true
4. Transitions to LOCAL_RTS
5. Set local_rts <= true
6. Sets tx_mode <= RTS, which means send E1 frames with CT = 0 using mission clock*

Green interface forwards RTS



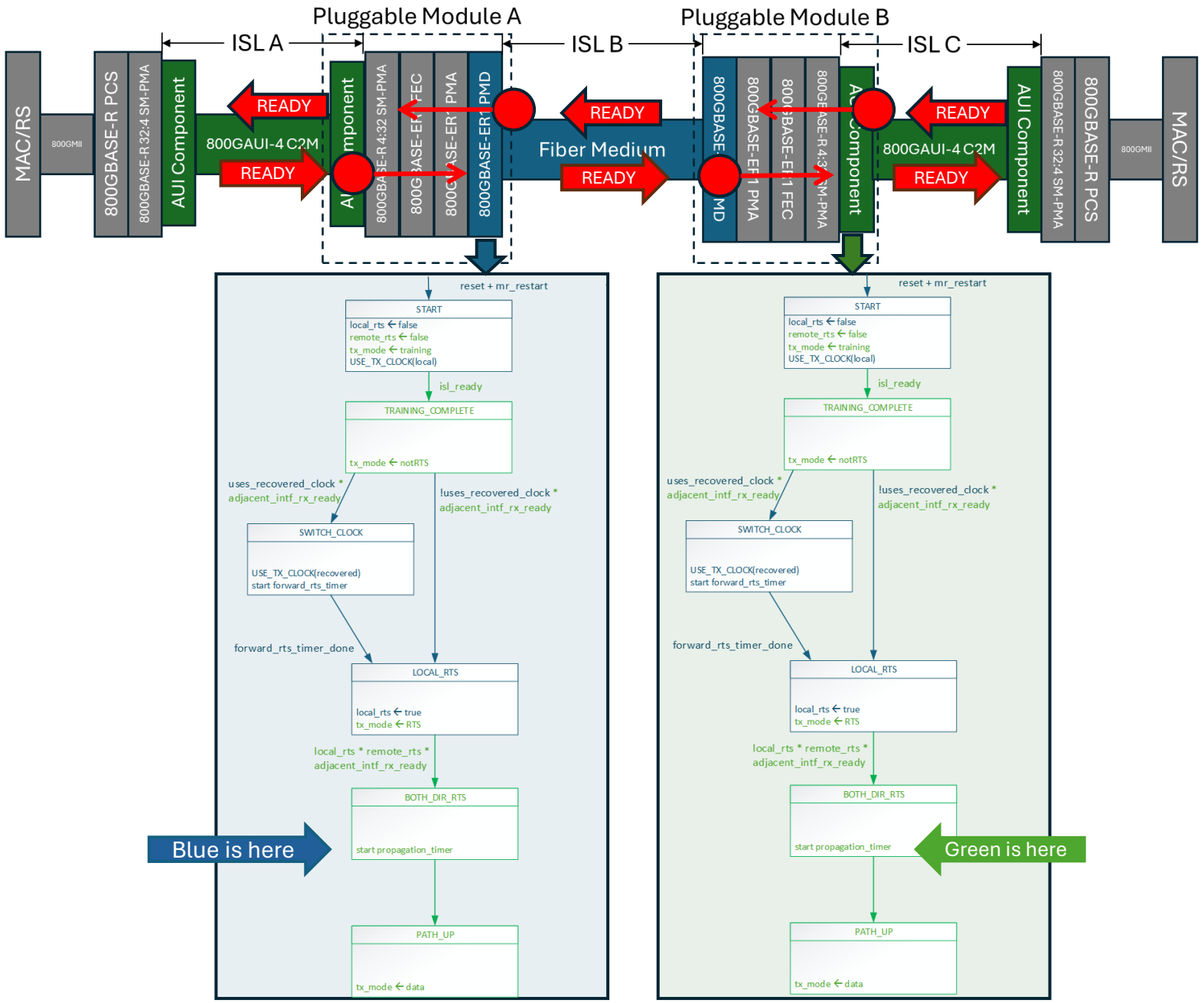
Green Interface

1. Set local_rts <= true
2. Set tx_mode <= RTS, which means send E1 frames with CT = 0 using mission clock
3. Peer interface receives RTS and transitions to BOTH_DIR_RTS state

Blue Interface

1. Both local_rts and remote_rts are true
2. Transition to BOTH_DIR_RTS state and wait until the propagation_timer expires

Green interface BOTH_DIR_RTS

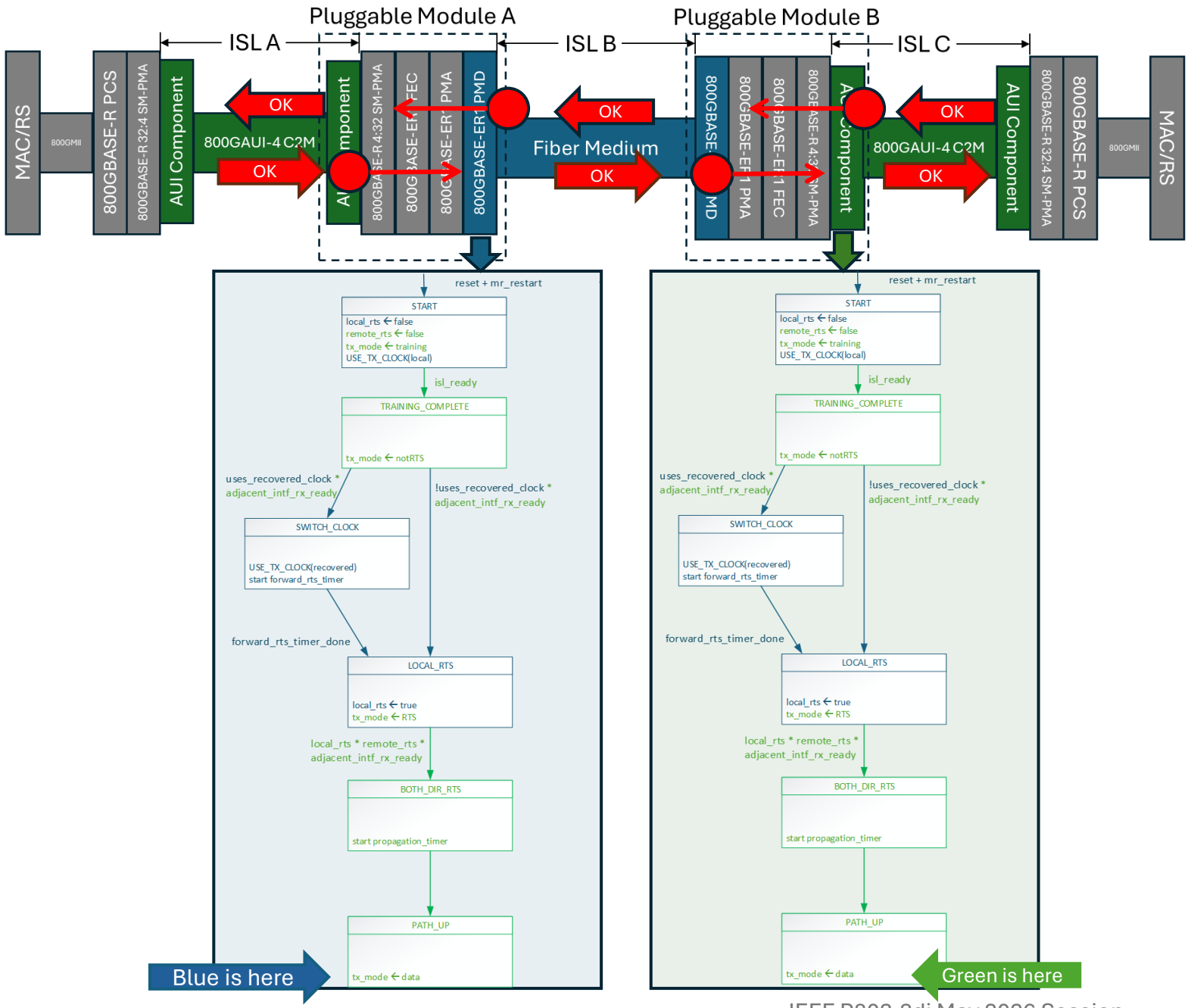


Green Interface

1. Both local_rts and remote_rts are true
2. Transition to BOTH_DIR_RTS state and wait until the propagation_timer expires

Blue Interface

1. Waiting until propagation_timer expires



Tx_mode <= data

Green Interface

1. Transitions to PATH_UP state after propagation timer expires
2. Sets tx_mode <= data

Blue Interface

1. Transitions to PATH_UP state after propagation timer expires
2. Sets tx_mode <= data, which sets the MNT bits to 000 (OK)

Summary

- APSU can complete successfully for these two coherent PMD applications by only instantiating an RTS FSM at the PMD
  - No ILT FSM was required
- The result is a very natural, intuitive mapping of the signals chosen to represent RTS for LR1 and ER1
- Future PMDs can benefit from the refactored model, as they may not support ILT but can take advantage of APSU’s plug-n-play functionality