

P1394b combined SCAT and C Code bug list

Done	45
Not submitted	2
Not yet done	40
Open	51
Recommend Accept	89
Recommend Accept in principle	16
Recommend Accept with question	1
Recommend Reject	5
<i>Total</i>	249

ID 151	Balloter Biva	How submitted Post ballot	Fixed in version
Title	Disabled Standby Port?		Status Not yet done
C code files		Owner PhyDogs	SCAT <input checked="" type="checkbox"/> BRAT 0 Page 178 Line 33

Problem description

Is disable allowed for a standby port? In that case, once it goes to disabled state and is thereafter enabled, should it go back to Standby state or Untested state? Please note that the in_standby flag is not cleared on entering Disabled state from Standby. Currently the state machine only provides transition to Untested state for this condition. This will enable the port, but the restoration will not succeed, since Restore Configuration packets will not be exchanged. Can the port go to Restoring state directly when enabled?

Bug fix description

Force an apparent disconnect to the peer when entering disabled from standby - by turning off bport_on for and holding off the background processing in a way exactly analogous to hard_disable (but don't actually use the hard_disable flag) for 2*DISCONNECTED_TONE_INTERVAL. The in_standby flag is (conveniently) used to determine that we entered disabled from standby.

The peer will see an apparent disconnect, will go to disconnected. When the timer expires on the local port, we allow the background processing to start up a new connection. The peer will go to untested, and wind up in loop_disabled with untested_fault set.

A bus reset will occur at the local node if the disable is given by a remote command packet but not if the disable is by a direct write to the PHY register bit (as in a). At the peer node, a bus reset will occur as a result of the apparent disconnection (see SCAT #247).

ID 126 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Don't want B-only PHY's to need arb_timers for collisions or isbr **Status** Open
C code files various **Owner** PhyDogs **SCAT** **BRAT** **Page** 0 **Line** 0

Problem description

The current code uses arb_OK within ds_arb_functions to arbitrate for resolving collisions and scheduling an isbr. However, arb_ok implies an arbitration timer (sensitive to gaps). For B-only PHY's we didn't want to require such a timer.

Since resolve_collision_request doesn't require any arbitration time, it isn't too difficult. But for isbr, we have to wait until we know we're in the async phase. Wouldn't it be better to mark the isbr as a "current_request" and let it be handled like other beta/BOSS requests? Any reason it must be done as a Legacy style request to the parent??

Bug fix description

ID 172 **Balloter** **How submitted** **Fixed in version**
Title Local_plug_present **Status** Open
C code files **Owner** PhyDogs **SCAT** **BRAT** 0 **Page** 237 **Line** 38

Problem description

Description in table 14-2 states that lpp=0 if phy is dc connected to connector. This would mean that a bilingual port would not bother to tone -

This needs some thought. D0.17 used dc_connected to force toning in this case - but this has been removed. I can't remember why we did this in the first place - it was something a bit subtle.

Bug fix description

ID 176 **Balloter** **How submitted** **Fixed in version**
Title MIN_PKT_SPACING and MIN_PHY_PKT_SPACING **Status** Open
C code files **Owner** PhyDogs **SCAT** **BRAT** 0 **Page** 250 **Line** 14

Problem description

The two parameters Min_Pkt_Spacing, Min_Phy_Pkt_Spacing are used in Figure 15-2 and its accompanying text - but never defined or even mentioned elsewhere in the document.

Bug fix description

ID 181 **Balloter** **How submitted** **Fixed in version**
Title CONFIG_TIMEOUT **Status** Open
C code files **Owner** PhyDogs **SCAT** **BRAT** 402 **Page** 254 **Line** 45

Problem description

The value of CONFIG_TIMEOUT needs to be adjusted to prevent false positive loop detects (always was possible in the Legacy draft, but the effect now is less benign).

Bug fix description

See SCAT

ID 212 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Value for ROOT_CONTENTEND_FAST **Status** Open
C code files **Owner** MT **SCAT** **BRAT** 0 **Page** 253 **Line** 54

Problem description

ROOT_CONTENTEND_FAST is shorter than the maximum round-trip time. This may result (state T3:Root Contention) in both nodes of a contending pair seeing the peer's PARENT_NOTIFY after the contention timeout, and concluding that they are the child, and moving into T1: Child Handshake. I think that this value should be doubled to 1.6 usec for a Beta link. However, when the peer is a Legacy port, it is important to use the current 0.8 usec value. Note that the current 3.2 usec value for ROOT_CONTENTEND_SLOW will give a bias towards the 1394b PHY becoming root in the case of contention between a Legacy PHY and a 1394b PHY.

Bug fix description

ID 248 **Balloter** **How submitted** **Fixed in version**
Title LC connector drawings **Status** Not yet done
C code files **Owner** Max. **SCAT** **BRAT** 0 **Page** 0 **Line** 0

Problem description

AR: Max Bassler find connector reference drawings for LC connector for use in the GOF chapter.

Bug fix description

ID 245 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Update references to Bport active in comments **Status** Not yet done
C code files **Owner** JH **SCAT** **BRAT** 0 **Page** 0 **Line** 0

Problem description

need to scrub the C code comments for references to bport active and change to bort on

Bug fix description

ID 246 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Untested port when all others are suspended **Status** Not yet done
C code files **Owner** JH **SCAT** **BRAT** 0 **Page** 0 **Line** 0

Problem description

Need to resume all suspended ports and wait for the resumption to complete before starting to test the new connection.

Bug fix description

Do this at the same time as the restore test (when a new connection invalidates nephew qualification)

ID 247 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Disconnection during suspend **Status** Not yet done
C code files **Owner** JH **SCAT** **BRAT** 0 **Page** 0 **Line** 0

Problem description

disconnection during suspend should cause a bus reset

Bug fix description

ID 221 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title PTX3:PTX1 transition **Status** Not yet done
C code files **Owner** JH **SCAT** **BRAT** 0 **Page** 167 **Line** 20

Problem description

I was wondering what good the PTX3:PTX1 transition does given P1394b as we know it today.

Specifically, we no longer attempt to transparently (to higher layers) resync like we once did. I believe the current intent is that once sync is lost, force a disconnect and bring the port connection up from scratch. Nominally, this happens as follows (I think) Local end of peer-to-peer link is in P2:Active, PTX3:Transmit, and PRX4:Receive.

After the invalid_count reaches 4 (5 consecutive errored codewords), sync_error_signal is set forcing PRX4:PRX1. sync_lost_signal is set FALSE, forcing PTX3:PTX1. This causes the transmitter to send training requests and to set bport_sync_ok to FALSE.

With bport_sync_ok set false, receive_ok is cleared forcing the P2:P3 transition.

In P3, the port notes that lost of sync caused the problem and sets sync_fail. It then deactivates the port (bport_active = FALSE), and heads off to P5.

In suspended_actions, sync_fail causes connected to be cleared and, as a consequence, P5:P0 is taken.

Now on the far end of the peer-to-peer link, the generation of TRAINING requests in step 3 may cause the far end invalid_count to approach the limit 4 and cause the above cascade of events to occur on the far end. Even if the training requests don't trigger the above events, the fact that the local port goes to P0 and starts toning for a minimum interval will guarantee the far end visits P0 as well.

So I'm wondering what value the PTX3:PTX1 brings, particularly the part about sending TRAINING characters again. This seems to have no purpose, given that bport_active will be turned off shortly and force both ends of the wire into P0. And sending TRAINING at an arbitrary instant may or may not be detected at the far end. For example, if sent outside of the packet context, then every TRAINING symbol causes the invalid_count to be bumped up quickly to the limit. But if PTX1 is entered during the data context, then only 1 in 64 TRAINING symbols will be detected as invalid (the comma is detected as invalid, all others are interpreted as valid DATA symbols). So the invalid_count probably doesn't build up to 4, and we never get the ending symbols required to take the far end receiver out of the data context. So just sending TRAINING at any ole time doesn't seem to work as expected ... we still require the visit to P0 (bport_active being turned off) to guarantee recovery.

So I was tempted to delete PTX3:PTX1 and add "| sync_lost_signal" to the PTX3:PTX0 transition. (Also, the bit of code which counts TRAINING as INVALID symbols could be removed as well, I think). However, such a fix is still "broken". If we run off to PTX0 before the connectivity management state machine had a chance to clear bport_active, then we'd wind up back in PTX1 by virtue of the PTX0:PTX1 transition.

So I guess my real question is: should we avoid sending TRAINING requests immediately on loss of sync, or not bother since we know a visit to P0 is imminent? If we want to bother, any ideas for a simple code fix?

Have I missed something more subtle??

Bug fix description

Delete this transition

ID 237 *Balloter* DW *How submitted* Post ballot *Fixed in version*
Title restore packet from Uncle *Status* Not yet done
C code files *Owner* JH *SCAT* *BRAT* 0 *Page* 286 *Line* 16

Problem description

when PHY receives restore packet from Uncle. This is not repeated on the PHY-link interface

Bug fix description

Amend C code (change TRUE to FALSE, and change the comment)

ID 234 *Balloter* DW *How submitted* Post ballot *Fixed in version*
Title Full FIFO behavior *Status* Not yet done
C code files bport_rx.c *Owner* JH *SCAT* *BRAT* 0 *Page* 306 *Line* 55

Problem description

Need to allow the port state machine to push ending symbols into the FIFO, no matter what is already there.

Bug fix description

Change the C-code for the fifo between the port and arb state machines so that the port will write a value and then check to see if the fifo is full. If it is, it will not advance the pointer.

ID 235 *Balloter* DW *How submitted* Post ballot *Fixed in version*
Title Bport error counter behavior *Status* Not yet done
C code files bport_rx.c *Owner* JH *SCAT* *BRAT* 0 *Page* 312 *Line* 39

Problem description

Don't bump the error counter if padding is in the wrong place, or data is in the wrong place. This would most likely happen if the speed code was corrupted. If so, then we'll get tons of misplaced data and padding, causing the error count to run amuck even though we only had a single bit error.

Bug fix description

Only bump the error count for invalid 8/10 decodes ... not protocol layer stuff.

ID 236 *Balloter* DW *How submitted* Post ballot *Fixed in version*
Title Bport FIFO Data behavior *Status* Not yet done
C code files *Owner* JH *SCAT* *BRAT* 0 *Page* 312 *Line* 43

Problem description

Must push one data byte into the FIFO for each expected data byte. So if we were expecting data and got either padding or invalid symbol, we want to still push into the fifo to prevent underrun. What we push is somewhat arbitrary ... the C code currently uses push_data_zero(). David prefers repeating the pad, but can live with any arbitrary data.

Bug fix description

Change push_data_zero to add weasel comments that what is pushed isn't important. As a corollary, we affirmed that we don't push at all when we weren't expecting a data byte.

ID 241 *Balloter* CWS *How submitted* Post ballot *Fixed in version*
Title DATA_NULL handling *Status* Not yet done
C code files ds_arb_functions.c *Owner* JH *SCAT* *BRAT* 0 *Page* 325 *Line* 54

Problem description

DATA_NULL doesn't seem to cause nodes to exit A0:Idle. Also, DATA_NULL followed by an arb or an IDLE state doesn't seem to allow receive actions to exit.

Bug fix description

ID 243 *Balloter* CWS *How submitted* Post ballot *Fixed in version*
Title Auto advancing through received tokens *Status* Not yet done
C code files process_req.c *Owner* JH *SCAT* *BRAT* 0 *Page* 333 *Line* 0

Problem description

While in RX:Receive, and a token is received (AST, ARBRST, CST), we set a flag. But the token doesn't appear to be removed from the FIFO as I think was intended.

Bug fix description

ID 223 *Balloter* DW *How submitted* Post ballot *Fixed in version*
Title register 0 in self_ID *Status* Not yet done
C code files self_id.c *Owner* JH *SCAT* *BRAT* 0 *Page* 343 *Line* 9

Problem description

move register 0 transfer so that it is earlier than it is now in self_ID transactions so that it is coincident with data_end indications. (Phydogs 8/18)

Bug fix description

ID 228 **Balloter** Jim Skidmore **How submitted** Post ballot **Fixed in version**
Title grant_received_from_senior in calls to boss_end_packet_actions() **Status** Not yet done
C code files **Owner** JH **SCAT** **BRAT** 0 **Page** 350 **Line** 33

Problem description

boss_end_packet_actions is always called with the grant_received_from_senior parameter always set to FALSE. Is this an oversight?

Bug fix description

```
case GRANT:
case DATA_END:
...
else {
  if (rx_arb_state == GRANT) {
    boss_end_packet_actions(receive_port == senior_port);
  } else if ((iso_cycle || (byte_count == 1)) && (receive_port != senior_port)) {
    boss_end_packet_actions(FALSE);
  } else {
    stop_tx_packet(DATA_END, cur_speed, DATA_END_TIME, cur_format, NPORT);
  }
}
}
break;
```

ID 95 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Wait more times through the reset loop before declaring a loop? **Status** Open
C code files node.c **Owner** JH **SCAT** **BRAT** **Page** 287 **Line** 27

Problem description

Current code declares a loop must be present if we get to reset_count > 2. Since we count from 0, I think this means that we go into R0 4 times before assuming a loop. This may be okay.

My fear was that we weren't waiting long enough. For suspend/resume and loop breaking, lots of code defers an inbound bus reset while bus_initialize_active is true due to an ongoing reset. This period could last a long time (twice a long bus reset plus some?). So it seems like the node sending the "inbound bus reset" may go through R0 a few times until it is recognized. Maybe the first time is a short bus reset, the second is a long, the third is long again and hopefully seen in a worst case non-looping topology. But we are getting pretty close to that magic number of 4

Bug fix description

ID 206 **Balloter** **How submitted** Post ballot **Fixed in version**
Title Speed exchange in self_ID **Status** Open
C code files transmit_functions **Owner** JH **SCAT** **BRAT** 0 **Page** 318 **Line** 43

Problem description

S2:S3 calls portTspeed to send raw speed signal during speed signalling ... not clear how this causes dsport_tx to call ds_portTspeed.

Bug fix description

Make clear (probably with new arb token) whether we want DP to be transmitted with the speedcode.

ID 218 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Maybe should allow multiple ARBRST's in a row? **Status** Recommend Accept
C code files **Owner** JH **SCAT** **BRAT** 0 **Page** 0 **Line** 0

Problem description

Various possible corner cases, and lost token conditions (see email thread)

Bug fix description

we must have the timeout for the case where the ARBRST* token is lost and phases don't advance

ID 161 **Balloter** Biva **How submitted** **Fixed in version**
Title No active ports in Loop Prevention **Status** Recommend Accept
C code files **Owner** JH **SCAT** **BRAT** 0 **Page** 185 **Line** 21

Problem description

Biva:-

The textual description in Sec11.7.12 (No Active Ports) indicates that the test interval may be set to 0 if the PHY currently has no ports in the active state. Is this supported by the C-code ?

[JH] I don't believe the decribed behavior is desired or necessary. The text should be changed. A related discussion is in the e-mail thread "RE: isolated nodes and such". Basically, we now allow single-port PHY's to skip the establishing dominance phase. Isolated nodes with multiple ports do not skip the dominance phase because they may still cause a loop or suffer from a race condition in which they send an attach out port #0 at the same time a connection to another isolated node reports an attach_request inbound on port #1. Now we may form a loop, etc. if we aren't really careful. Since it doesn't take an isolated node long to establish dominance, we opted not to make the optimization and deal with all of the hazards. So I consider this closed provided Wooten updates the text.

Bug fix description

ID 224 **Balloter** DW **How submitted** Post ballot **Fixed in version**
Title Qualification of configuration requests **Status** Not yet done
C code files **Owner** DW **SCAT** **BRAT** 0 **Page** 0 **Line** 0

Problem description

Configuration requests need to be sent such that they can be 'debounced'. We can't have a configuration request sent as a single symbol. That symbol can be corrupted by a single-bit error and that could, in some cases, cause problems. On the other hand, there are cases where a node may receive a single configuration request symbol. That symbol cannot be accepted without qualification (it could be received as a result of noise on the line). Rather than go down every case of bad configuration token to decide if each of them leads to a correctable condition, it is better to debounce them. Debouncing would mean that they are only accepted in a specific context (i.e., only accept a disconnect request after seeing a PHY packet.) It might be easier, however, simply to state that each configuration request must be followed by another configuration request of the same type. If we couple this with a statement that each configuration request must be sent for at least 8 symbol times, then we have a secure way of getting the configuration requests through (it becomes just at least reliable as the two control tokens that we have at the beginning and end of each packet.)

Bug fix description

I will write a piece of text that describes why we are qualifying the configuration requests. Basically, we want the configuration requests that result in bus fragmenting to be as reliable as the start-of-packet or end-of-packet delimiters (mostly, when we miss either a SOP or EOP we end up fragmenting the bus.) Getting a double-bit error in an arbitration request that turns it into a valid configuration request is less likely than getting a double-bit error affecting two consecutive control codes. However, arbitration requests occur with much higher frequency so the chances of fragmenting the bus due to an erroneous configuration request is higher than fragmenting the bus due to loss of an SOF or EOF. However, if we require that the configuration requests come in pairs, the chances of two codes being changed in such a way that they descramble and decode into the same configuration request is very small.

ID 232 **Balloter** DW **How submitted** Post ballot **Fixed in version**
Title New link cancellation rules **Status** Not yet done
C code files **Owner** DW **SCAT** **BRAT** 0 **Page** 199 **Line** 30

Problem description

sometime during the period between BUS_RESET or PH_RESTORE* and the register 0 transfers, the PHY needs to ignore or cancel most requests with cycle start being the exception.

Bug fix description

The text as David presented is a sufficient description ... the interface behaves as if the requests are continuously ignored and not queued by the PHY.

ID 231 **Balloter** DW **How submitted** Post ballot **Fixed in version**
Title FOP/nephew restore condition **Status** Not yet done
C code files **Owner** DW **SCAT** **BRAT** 0 **Page** 228 **Line** 44

Problem description

A FOP which is also a nephew (when it's only active port on the serial bus is in standby as a nephew) needs to initiate a restore if it receives a bus request from the PIL.

Bug fix description

Something like if the best request to forward on the senior port (which is in standby) is non-null, then restore? I'll try this route unless someone has a better idea ...

ID 230 **Balloter** DW **How submitted** Post ballot **Fixed in version**
Title we need PH_RESTORE* defined for P2P **Status** Not yet done
C code files **Owner** DW **SCAT** **BRAT** 0 **Page** 229 **Line** 0

Problem description

Bug fix description

ID 203 **Balloter** **How submitted** **Fixed in version**
Title Int_enable bit **Status** Not yet done
C code files **Owner** DW **SCAT** **BRAT** 0 **Page** 237 **Line** 25

Problem description

Description of the Int_enable bit in table 14-2 should include the Standby bit (i.e., changes in the standby bit should cause an interrupt if the Int_enable bit is set for this port). More issues on standby_fault and restore_fault, and correct action on behavior leading to restore_fault.

Bug fix description

(partially done, matters arising need dealing with)

Also requires a change to the description of the Port_Event bit. Also should include Standby_fault.

ID 204 **Balloter** **How submitted** **Fixed in version**
Title PIL-FOP negotiation **Status** Recommend Accept in principl
C code files **Owner** DW **SCAT** **BRAT** [^] 0 **Page** 227 **Line** 0

Problem description

PIL-FOP negotiation should be performed at the time of speed signaling, rather than by straps etc. Also an "external" FOP should be supported.

Bug fix description

Extra bits are used in the speed negotiation. ("Done" in Connection Management Chapter)

ID 170 **Balloter** **How submitted** **Fixed in version**
Title untested_fault visible to software? **Status** Recommend Reject
C code files **Owner** DW **SCAT** **BRAT** 0 **Page** 236 **Line** 1

Problem description

Currently, a port can be in P12 for two reasons ... one is loop_disabled while the other is if some sort of fault situation occurred during which a loss of sync occurred. The loop_disable case is visible to software via a per port register bit, the untested_fault is not. Is this desirable, or should a second fault bit be provided?

As a reminder, when a loop is found, one side of an untested connection sets loop_disabled to initiate the transition to P12. The far end sees this as an untested_fault. So in topologies with loops, untested_faults should be "common".

Bug fix description

Decide that this need not be visible to software.

ID 244 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title removal of restore_fault **Status** Not yet done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 177 **Line** 0

Problem description

If a restore fails, the current action is to go to P9, with no reporting on either uncle or nephew. It is not clear what can be done anyway under these circumstances. Better is to go to p0 disconnected, and remove all references to restore_fault

Bug fix description

ID 156 **Balloter** **How submitted** **Fixed in version**
Title missed reset in standby **Status** Not yet done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 182 **Line** 15

Problem description

The spec needs a healthy dose of clarification on the missed reset issues. For example, how does the phy register bit Standby_reset (one per node) relate to missed reset (one per port)? Is Standby_reset only meaningful at the nephew? How can it have meaning at the Uncle?

And how does all of this relate to the link interface? We had talked about telling the link if it missed a reset while the interface was asleep (even if none of the PHY ports were in standby), but I don't understand if this has any impact on the bit or not.

Bug fix description

ID 157	Balloter	How submitted	Fixed in version
Title	restore indications	Status	Not yet done
C code files	Owner	CWS	SCAT <input checked="" type="checkbox"/> BRAT 0 Page 182 Line 23

Problem description

restore_actions() sets port_event to true, but fails to give a PH_EVENT_indication(PH_INTERRUPT) which is the norm. But it does give a PH_EVENT of PH_RESTORE_*. Both the PH_INTERRUPT and PH_RESTORE* have encodings for the parallel PHY/Link interface.

This is a question of intent.

- 1) Should restore cause a port_event, and by extension, an interrupt?
- 2) Should a restore cause both an interrupt transfer and a restore transfer as status to the link? Which is first?
- 3) What meaning does PH_RESTORE_NO_RESET have at the uncle which can have multiple ports restore simulatneously?? Or differently, I don't think it is possible for there to have a reset which the uncle missed.
- 4) And related to a previous question/bug, how should the link get it's new nodeID if PH_RESTORE_RESET is returned. It can't always snoop the packet, because it may not have activated the PHY/Link interface until the port_event for restore occurred (LinkOn).

Bug fix description

If the host sets the port interrupt enable bit, then both indications go across as separate indications.

The PH_INTERRUPT happens last

ID 166	Balloter	How submitted	Fixed in version
Title	standby_reset and missed_reset	Status	Not yet done
C code files	Owner	CWS	SCAT <input checked="" type="checkbox"/> BRAT 0 Page 233 Line 19

Problem description

relationship between this and missed_reset?

Bug fix description

ID 189	Balloter	MS	How submitted	Fixed in version
Title	Automatic Setting of IBR register	Status	Not yet done	
C code files	Owner	CWS	SCAT <input checked="" type="checkbox"/> BRAT 0 Page 234 Line 26	

Problem description

change to rwu

Bug fix description

#2A. Just a comment: I think we didn't attempt to forward CS tokens in start_rx_packet() because it may have been difficult to stomp (having to detect data_coming concurrently with generating token at this point in the C code) and because there are cases in which we'd have to wait until IDLE anyway (such as CS being received while we already started repeating the CS packet payload). No action required for #2A.

#2B. Yes, there is a problem for an all B bus, if the cycle master doesn't send the CS token at the slowest port speed in the _network_ AND if the amount of time in A0:IDLE before the end of the isoch period doesn't exceed the symbol time of the slowest network port. The fix in the C code is to require the CS token to always be generated at an S100 symbol time for a B Bus, and to refrain from stomping in a B Bus. This means that PPM differences won't having us accidenatly stomping the guaranteed S100 CS token and we can regenerate the S100 timings in a B Bus without fear of overflow.

#3. Correct, it must be a requirement for B links with pending isoch traffic to generate cycle start token requests when a cycle start packet is received. Furthermore, this request needs to follow the same timings as a P1394a issued IsoReq to ensure that the token reaches the "root" before a subaciton gap pops. Recommended action is to have DRW improved the language in 12.5.2.1 to clarify that it is a must (not a may) if the link has isoch traffic queued, and that the timings must be obeyed.

#4) I can't figure out how any of the inband status transfers can be sent simultaneously, but the text in 12.8.1.2 before Table 12-18 sure confuses this point. Recommendation is to clarify in 12.8.1.2 that either the bits don't happen simultaneously, or what interpretation should be given to various ambiguous bit pairs.

This item is being kept open for the moment pending consideration of further issues affecting preservation of gap times

<i>ID</i>	239	<i>Balloter</i>	CWS	<i>How submitted</i>	Post ballot	<i>Fixed in version</i>	
<i>Title</i>	FIFO contents when port goes inactive				<i>Status</i>	Open	
<i>C code files</i>		<i>Owner</i>	CWS	<i>SCAT</i>	<input type="checkbox"/>	<i>BRAT</i>	0 <i>Page</i> 333 <i>Line</i> 0

Problem description

What happens to FIFO when a port goes inactive? Will process requests go to quickly after lactive? Is it possible for the FIFO to still contain data (maybe a confirmation packet) which should be forwarded on before killing the port? And what about error cases when we lose a port in the middle of receiving a packet. Does a syntheszied ending arb state get pushed into the FIFO always, or does the reading end have to time out after a long while? How do we get the FIFO clearly flushed before it starts up again?

Bug fix description

<i>ID</i>	208	<i>Balloter</i>	CWS	<i>How submitted</i>	Post ballot	<i>Fixed in version</i>	
<i>Title</i>	Max_Legacy_path_speed not initialised				<i>Status</i>	Recommend Accept	
<i>C code files</i>	reset.c	<i>Owner</i>	CWS	<i>SCAT</i>	<input type="checkbox"/>	<i>BRAT</i>	0 <i>Page</i> 337 <i>Line</i> 36

Problem description

Need to be initialised in R0

Bug fix description

ID 240 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Incorrect term in standby_actions() **Status** Recommend Accept with questi
C code files port.c **Owner** CWS **SCAT** **BRAT** 0 **Page** 298 **Line** 36

Problem description

The following conditional statement in standby_actions() is incorrect:

```
if (!(Beta_mode && receive_ok) || (Beta_mode && !bport_sync_ok))
    activate_connect_detect(0);
```

When Beta_mode is TRUE, the first term, !(Beta_mode && receive_ok), evaluates the condition to TRUE. The second term, (Beta_mode && !bport_sync_ok), would have no effect upon the condition regardless of the state of bport_sync_ok.

Bug fix description

Indeed, the whole section is wrong. The model of what is correct is a bit higher up in suspend_actions(). However, standby can only happen if we're operating in Beta mode!!

```
while ((!bport_sync_ok)
    && (connect_timer < 12 * RESET_DETECT))
;
if (!bport_sync_ok)
    activate_connect_detect(0);
```

Why do we need 12*RESET_DETECT??? This number was in 1394a suspend/resume for the Legacy camcorder problem.

ID 168 **Balloter** **How submitted** **Fixed in version**
Title Encoding of Max._legacy_path_speed **Status** Recommend Reject
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 234 **Line** 42

Problem description

pointless having 3 bits for this, and raises implementation red-herring questions

Bug fix description

No confusion as it is. All speeds are encoded in three bits for consistency.

ID 3 **Balloter** CWS **How submitted** Not submitted **Fixed in version**
Title register read/write should not be PHY_arb_req services **Status** Not submitted
C code files phy_services, process requests **Owner** **SCAT** **BRAT** **Page** **Line**

Problem description

Bug fix description

ID 10 **Balloter** CWS **How submitted** Not submitted **Fixed in version**
Title DS port receive FIFO **Status** Not submitted
C code files dsport_rx.c **Owner** **SCAT** **BRAT** **Page** **Line**

Problem description

The FIFO is going to get filled in the local clock domain for arb states, and in the received clock domain for data. This is not captured in the C code (in particular, switching between the two).

Bug fix description

under study

ID 138 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Register read/write services **Status** Not yet done
C code files phy_services.h **Owner** **SCAT** **BRAT** **Page** 276 **Line** 50

Problem description

Ideally, register read/write requests should be a separate service indication since not all implementations use LREQ. Not a very high priority, but it is nice to have a separate service model for the register block. Additionally, the read/write services, regardless of how they are defined, are not currently described in Clause 15 service model.

Recommendation would be to remove PH_REG_READ and PH_REG_WRITE from C code completely, and optionally have Mike add a new service description in the text.

Bug fix description

ID 124 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title node_status_monitor() accounting **Status** Not yet done
C code files node.c **Owner** **SCAT** **BRAT** **Page** 283 **Line** 21

Problem description

Gut feel on this one is that the counting of suspended_ports in node_status_monitor is not as desired. Specifically, it seems that some ports which are in P7 or P8 are counted as suspended ports (since in_standby isn't true yet). Is that desired?

Also, a port in P12 isn't counted as suspended if untested_fault is true, but is counted if loop_disabled is true.

Bug fix description

change in_standby to be set on entry to P7 and P8 (rather than P9) so that all of P7, P8, P9 and P10 count as "in_standby".

Add untested_fault into the suspended test

Set in_untested_actions() and maybe clear untested_state immediately before exiting untested_actions or on the state transitions

Clear untested_state on All:P6

connection_status - implement correctly the comment which says
// check for continuous signaling or bias in relevant port states (P1, P2, P3, P4, P7, P8 and P11)
currently the test does not include P8, and it needs to, and it needs to include P10 to prevent falling thru to the next block, also it does not capture P4. Fix by replacing these tests by tests on explicit port states.

Issue of turning on toning after turning off the port - this needs to be done relatively quickly, to prevent a disconnect from being detected at the far end. Proposal is to do this in the same block of code if receive_ok goes away (in Beta mode).

Fix the comment in the next section to include the standby state

Also, resume_all_ports should test for P3, P4 and P5

ID 101 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Second speed signal ignored by bport_rx. **Status** Open
C code files bport_rx.c **Owner** **SCAT** **BRAT** **Page** 0 **Line** 0

Problem description

When a bport receives a concatenated null packet it will ignore the second speed code. Speed_known is set upon receipt of the first speed symbol and it is not cleared until a non data prefix symbol and a non speedc is received. Therefore the speed_known will not be cleared until the speeda or speedb symbol of the concatenated packet is received. This is too late for bspeed_filter to do its job.

Bug fix description

ID 104 **Balloter** Biva **How submitted** Post ballot **Fixed in version**
Title Beta port startup latency **Status** Open
C code files port.c **Owner** **SCAT** **BRAT** **Page** 291 **Line** 50

Problem description

When the speed handshake successfully ends for a bport, it immediately sets bport_active TRUE and the beta port starts training. This point of switching from (speed) toning to training may fall right after the last speed tone bit position during the last lap of speed handshaking. Now, assuming that the tone sampling points of the peer is delayed and the waveshape of signal_detect for the peer is distorted (meaning tone durations are stretched and inter-tone gaps shrunk), it may be possible that the peer sees a false tone at the last speed bit sampling point.

For illustration, assume that the local node sends a speed code of S100. The tone/signal transmission sequence from the local node is:

```
HLLLHLLLHLLLLLLLLLHHHHH....  
1 2 3 4 56  
* * * * *
```

- 1-> reference tone
- 2-> ack
- 3,4,5-> S100 tone pattern
- 6-> switching to training phase.
- *-> tone sampling positions for the peer.

In case of the above sequence, the speed pattern detected by the peer may possibly be 101 (instead of 100 for S100), because the last "*" happens to fall after the local port has started off with the training phase. Hence it may incorrectly detect a received speed of S1600.

I am not sure if such a scenario will actually happen, since the c-code implies that the sampling points remain locked on closely to the starting edge of tone positions of the peer. But since the positions 5 & 6 are just adjacent, I am a bit apprehensive of a false speed tone detection here.

Wouldn't it be safer to give a SPEED_TONE_BIT_INTERVAL gap after the last speed bit position before the port can initiate continuous bitstream transmission? Will it have an adverse effect on the max latency in beta port startup?

Bug fix description

See also #41
[CWS] It may be a bit worse than this, we probably should ensure that all 6 bits are sent before signaling EVT_SENT_ACK, particularly considering the alterations put in to allow interoperability with non-1394 devices

ID 128 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title connection_status and port state machine contend on toning and start_port(**Status** Open
C code files port.c **Owner** **SCAT** **BRAT** **Page** 296 **Line** 4

Problem description

When suspended, loop_disabled, or disabled, connection_status() is "stuck" in a bit of code which can consume long periods of time during which it is toning out and listening for tone or continuous tone coming in.

When the port state machine is in one of these states, an event asynchronous to connection_status can cause a port state machine transition. For example, clearing of the disabled flag could allow P6:P11. Clearing of the loop_disabled flag by a bus reset could cause P12:P11, and setting of the resume bit can cause P5:P1. And I think Standby to restore can happen by the restore flag causing the P9:P10 transition.

So while the connection_status() routine is stuck toning for finite time, the port state machine can easily enter P11, P10, or P1 and start to signal continuously. As a result, we have two drivers on the tpB output conflicting with each other.

Bug fix description

ID 122 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Danger in overloading of immediate_request **Status** Open
C code files port.c **Owner** **SCAT** **BRAT** **Page** 297 **Line** 36

Problem description

Probably a nit: suspend_target_actions() sets immediate_request to force suspend related packets and signaling on the bus. However, immediate_request can be asynchronously cleared by the PHY/Link interface. Consequently, it is technically possible for the variable to be cleared before being serviced, causing a delay (and problems?) before the suspend actions are completed properly. Would it be safer to define an independent request which, like immediate_request, causes an unconditional transition into TX?

Bug fix description

ID 87 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title To Interrupt or not on Restore ... and what indication to Uncle? **Status** Open
C code files port.c **Owner** **SCAT** **BRAT** **Page** 298 **Line** 13

Problem description

restore_actions() sets port_event to true, but fails to give a PH_EVENT_indication(PH_INTERRUPT) which is the norm. But it does give a PH_EVENT of PH_RESTORE_*. Both the PH_INTERRUPT and PH_RESTORE* have encodings for the parallel PHY/Link interface.

Bug fix description

ID 49 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title dsport_transmit_actions **Status** Open
C code files dsport_tx.c **Owner** **SCAT** **BRAT** 466 **Page** 304 **Line** 8

Problem description

(Comment #23)

Table 16-9, pg. 304, function dsport_transmit_actions

1) Typographic/spelling error in comment--change line ~8 from:

```
} else { // tag == ARB_START  
to:
```

```
} else { // tag == ARB_STATE
```

2) When SPEED is transmitted, don't necessarily send data-prefix (e.g., during self-ID).

Delete pg. 304 line ~22:

```
ds_portTarb(TX_DATA_PREFIX);
```

When DATA_END is transmitted, don't necessarily send dribble bits (e.g., null packet).

Change pg. 304 line ~26 from:

```
tx_dribble_bits(TX_DATA_END);  
to:
```

```
if (ds_in_packet) tx_dribble_bits(TX_DATA_END);
```

Bug fix description

ID 139 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Does bport_rx deal with concatenated speed codes? **Status** Open
C code files bport_rx.c **Owner** **SCAT** **BRAT** **Page** 306 **Line** 1

Problem description

Withdrawn link requests can cause a speed code to be sent on an otherwise empty packet. Concatenate a few of these together, and you get speed code followed by DP followed by another speed code. Doesn't look like bport_rx handles that too well.

Bug fix description

ID 92 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Inconsistent clearing of FIFO pointers **Status** Open
C code files bport_rx.c **Owner** **SCAT** **BRAT** **Page** 310 **Line** 50

Problem description

The FIFO pointers fifo_wr_ptr and fifo_rd_ptr are not cleared simulatenously in all cases. The fifo_rd_ptr is always cleared at power_reset. In ds mode fifo_wr_ptr is also cleared at power_reset. However, in beta mode fifo_wr_ptr is also cleared when bport_active goes false.

Bug fix description

Suggest that both pointers only be cleared at power_reset.
in inactive_actions(),

fifo_wr_ptr = 0;

is moved up into the power_reset loop

[JH] Actually, the current C code for the DS port has the write pointer being reset whenever there is a power_reset or dsport_active is false (see decode_bit()). Consequently, this proposed fix will make the beta and ds ports different in behavior, not the same. Cosequently, I recommend no action until we determine what to do about FIFO clean-up in underrun, overrun, and suspend/standby cases.

ID 57 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title send_control **Status** Open
C code files transmit_functions.c **Owner** **SCAT** **BRAT** 478 **Page** 318 **Line** 28

Problem description

(Comment #31)
Table 16-14, pg. 318, function send_control
Control symbols should be sent to active ports only.
Change pg. 318 line 28 from:

if (Beta_mode[i]) {
to:

if (active[i] && Beta_mode[i]) {

and change line 35 from:

if (out_of_packet && i != not_to_port && Beta_mode[i]) {
to:

if (out_of_packet && i != not_to_port && active[i] && Beta_mode[i]) {

Bug fix description

ID 134 *Balloter* CWS *How submitted* Post ballot *Fixed in version*
Title Cycle Master is not receiving PH_ISOCH_EVEN/ODD indications *Status* Open
C code files transmit_functions.c *Owner* *SCAT* *BRAT* *Page* 319 *Line* 11

Problem description

In start_tx_packet when the cycle master is granted to send a cycle start packet, a PH_ISOCH_EVEN/ODD indication is _not_ sent to the link. I think this means that the link will not know the isoch phase, ever, unless it counts (modulo 2) after bus resets. Seems a little kludgy. And it is important for OHCI style applications to know the isoch phase so that cycle matching can be done properly. (Want an isoch stream to start up in a particular cycle ... need to know if that cycle will be odd or even so that the proper beta request is made and granted.)

But not sure of the fix ... for parallel link, PH_ISOCH_EVEN/ODD can not be delivered to link if the link already owns the interface ... so we would need to send it before granting the link. This is probably okay ... we don't grant until after calls to start_tx_packet return.

Bug fix description

ID 58 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title data-prefix padding following speed-signal sequence **Status** Open
C code files transmit_functions.c **Owner** **SCAT** **BRAT** 479 **Page** 319 **Line** 19

Problem description

(Comment #32)

Table 16-14, pg. 318-319, function start_tx_packet

I don't think data-prefix padding following speed-signal sequence is correct. The fork process is unnecessary and doesn't really accomplish the goal of ensuring enough deletable symbols when a cycle-start symbol is to be sent. The min_OK_port_speed variable is unnecessary (symbol stretching/padding is done at the pkt_speed). I suggest the following.

Delete all references to min_OK_port_speed.

Change pg. 319 lines 19-55 from:

```
fork
...
join
```

to:

```
#define MIN_DATA_PREFIX_TX 180 // Per 1394a, min data-prefix time preceding packet data
// (SPEED_SIGNAL_LENGTH + DATA_PREFIX_HOLD + 40)
```

```
if (sent_cycle_start)
    wait_next_symbol(min_operating_speed);
```

```
for (i = 0; i < NPORT; i = i + 1) {
    if (speed_OK[i]) {
        portTarb(i, DATA_PREFIX); // Ensures DS ports get data-prefix.
        if (!(pkt == LEGACY) && (link == LEGACY_LINK) && (pkt_speed == S100))
            portTspeed_general(i, pkt_speed, pkt);
    } else
        portTarb(i, DATA_NULL);
}
```

```
wait_next_symbol(pkt_speed); // Wait till speed-sig symbol completed.
```

```
for (i = 0; i < NPORT; i = i + 1)
    // Now send the next symbol on the Beta mode ports.
    if (Beta_mode[i])
        portTarb(i, speed_OK[i] ? DATA_PREFIX : DATA_NULL);
```

```
// Ensure timing for starting packet format, plus timing for deletable symbols. The
// requirement is that the originating node shall ensure that there is at least 17ns
// of deletable symbols at the start of each packet since the last deletable symbol
// was transmitted. The following code achieves this conservatively. Optimized
// implementations which recognize that deletable symbols have already been
// transmitted and reduce the time here accordingly are compliant.
```

```
if (pkt == LEGACY) {
    // Leave the speed signal for longer, send the next symbol on the DS ports.
    wait_time (SPEED_SIGNAL_LENGTH - symbol_time(rx_speed));
    for (i = 0; i < NPORT; i = i + 1)
        // Harmless writing this to the beta mode ports as well!
        portTarb(i, speed_OK[i] ? DATA_PREFIX : DATA_NULL);
    wait_time (convert(MIN_DATA_PREFIX_TX + DELETABLE_SYMBOL_TIME, pkt_speed) -
        SPEED_SIGNAL_LENGTH);
} else {
    wait_next_symbol(pkt_speed);
    wait_time (convert(DELETABLE_SYMBOL_TIME, pkt_speed));
}
```

Bug fix description

ID 59 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title stop_tx_packet **Status** Open
C code files transmit_functions.c **Owner** **SCAT** **BRAT** 481 **Page** 320 **Line** 20

Problem description

(Comment #33)
Table 16-14, pg. 320, function stop_tx_packet
Must add 20ns to end time to account for dribble bits on DS ports.

Change pg. 320 line 20 from:

```
if (pkt == LEGACY) wait_time(hold_time);
```

to:

```
if (pkt == LEGACY) wait_time(hold_time+20); // account for dribble bits on DS ports
```

Bug fix description

ID 60 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title portR_next_arb **Status** Open
C code files receive_functions.c **Owner** **SCAT** **BRAT** 482 **Page** 320 **Line** 43

Problem description

(Comment #34)
Table 16-15, pg. 320, function portR_next_arb
I don't think the "if (packet_ending[port_num]) { ... }" is needed.

Bug fix description

ID 61 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title wait_fifo_fill_time **Status** Open
C code files receive_functions.c **Owner** **SCAT** **BRAT** 483 **Page** 320 **Line** 54

Problem description

(Comment #35)
Table 16-15, pg. 320-321, function wait_fifo_fill_time
I don't think this function is correct (or even necessary). The requirement is that we wait at least 17 ns so as to guarantee that the rx FIFO won't underflow. To accomplish this, we need merely wait a couple of S100 tics. This function could simply be replaced with:

```
wait_time(20);
```

Bug fix description

[CWS] I think that there's a chance that the FIFO may already be sufficiently full, and if a blind wait is done, then it might overflow

ID 62 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title multiple received speed-signals **Status** Open
C code files receive_functions.c **Owner** **SCAT** **BRAT** 484 **Page** 321 **Line** 45

Problem description

(Comment #36)
Table 16-15, pg. 321, function start_rx_packet
Just as in 1394a, this function does not correctly handle the case of multiple received speed-signals (still a possibility).
I suggest the following to replace the start_rx_packet function in its entirety.
The wait_Betaport_event function can be eliminated.

```
#define MIN_DATA_PREFIX_RX (SPEED_SIGNAL_LENGTH + DATA_PREFIX_HOLD)

void start_rx_packet() { // Send data prefix and do speed signaling
    // receive on DS or Beta, repeat to both

    int i;
    ArbState arb;
    boolean data_started; // actual data now in rx fifo

    arb_timer = 0;

    if (!DS_stuck) {
        max_beta_timer = 0; // Timer only needs to be implemented in border capable nodes,
        DS_stuck = TRUE; // and note that this will have to be released by sending
        // a Legacy format packet.
    }

    portTarb(receive_port, IDLE); // Immediately send idle on the receive port
    // (i.e., remove grant, if any).
    data_started = FALSE; // no data yet

    format = LEGACY; // Assume legacy format until proven otherwise.

    // Wait for data to begin. While waiting, process any speed-signaling received or
    // any gap events (subaction gap or arb-reset gap) which occur.

    // Upon entry into start_rx_packet, the incoming arb state on the receive_port
    // will be either DATA_PREFIX, DATA_NULL, or SPEED. If receive_port is a DS port,
    // then the arb state will be DATA_PREFIX.

    // Also ensure that for legacy packet the min data-prefix requirements are met.

    while (!data_started ||
        (arb_timer < convert(MIN_DATA_PREFIX_RX, rx_speed) && format == LEGACY)) {

        arb = portRarb[receive_port];

        // Process any gap events which may have occurred.
        if (send_async_start_token || arb_reset) begin
            gap_repeat_actions(FALSE); // This may or may not advance one S100 symbol.
            portTarb(receive_port, IDLE); // Restore ports to previous state.
            tx_control(DATA_PREFIX, receive_port); //
        end

        // Now process the last arb state read from rx fifo.

        if (data_started) // If data already started, then we must just be killing
            ; // time till enough data-prefix sent.
        else if (arb == SPEED) {
            rx_speed = portRspeed[receive_port];
            format = current_pkt[receive_port];
            tx_control(DATA_PREFIX, receive_port); // Take care of DS ports, which must set
            // the DP before the speed-signal is sent
            // (but no harm to Beta-mode ports).
            advance_OK[receive_port] = TRUE; // Allow next symbol to be read from rx FIFO.
        }
    }
}
```

```

// Note: There must necessarily be another symbol in the rx FIFO by the time
// the following processing completes.

// Repeat the speed signal.
for (i = 0; i < NPORT; i = i + 1)
  if (i != receive_port && active[i]) {
    speed_OK[i] = (rx_speed <= port_speed[i]) && (Beta_mode[i] || format == LEGACY);
    // Do not repeat Beta format packets on DS ports!
    if (speed_OK[i])
      portTspeed_general(i, rx_speed, format);
      // format only needed for Beta-mode ports
    else
      portTarb(i, DATA_PREFIX);
  }
wait_next_symbol(rx_speed); // Wait for speed symbol/DP to be sent.

// Now send next symbol on Beta-mode ports.
for (i = 0; i < NPORT; i = i + 1)
  if (i != receive_port && Beta_mode[i])
    portTarb(i, DATA_PREFIX);

  if (format == LEGACY) {
    // Extend the speed signal.
    wait_time(SPEED_SIGNAL_LENGTH - symbol_time(rx_speed));
    // Now send next symbol on DS ports.
    for (i = 0; i < NPORT; i = i + 1)
      if (i != receive_port)
        portTarb(i, DATA_PREFIX); // Harmless writing this to beta-mode ports as well!
    // Advance to next symbol boundary beyond MIN_DATA_PREFIX.
    wait_time(convert(MIN_DATA_PREFIX, rx_speed) - SPEED_SIGNAL_LENGTH);
  } else
    wait_next_symbol(rx_speed); // Wait for DP/DN symbol to be sent.
}
else if (arb == DATA_BYTE) {
  data_started = TRUE; // Exit loop (when enough DP).
} else if (arb == DATA_PREFIX || arb == DATA_NULL) {
  tx_control(arb, receive_port); // Repeat the arb state.
  advance_OK[receive_port] = TRUE; // Allow next symbol to be read from rx FIFO.
}
else
  return; // There's no packet, so bomb out.
}
tx_speed = rx_speed;
wait_time (20);
}

```

Bug fix description

ID 63 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title decode_phy_packet **Status** Open
C code files decode_phy_pkt.c **Owner** **SCAT** **BRAT** 487 **Page** 324 **Line** 18

Problem description

(Comment #37)
Table 16-16, pg. 324, function decode_phy_packet
Must process only received self-ID packets.
Change pg. 324 line 18 from:

```
if (!Beta_mode[receive_port] || !received_speed_signal) { // originated from node with Legacy link or DS node
```

to:

```
if (receive_port < NPORT &&  
    (!Beta_mode[receive_port] || !received_speed_signal)) { // originated from node with Legacy link or DS node
```

Setting of senior_border and senior_port variables should only be done after transmission of own self-ID.
Change pg. 324 line 26 from:

```
if (Beta_mode[receive_port]) {
```

to:

```
if (Beta_mode[receive_port] && PHY_state == RX) {
```

Brackets needed.
Change pg. 324 lines 45-49 from:

```
else if (phy_pkt.ext_type == 0xF) // Resume packet?  
...  
else if (phy_pkt.ext_type == 0xE) { // LTP packet?
```

to:

```
else if (phy_pkt.ext_type == 0xF) { // Resume packet?  
...  
} else if (phy_pkt.ext_type == 0xE) { // LTP packet?
```

Bug fix description

ID 64 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title data_coming **Status** Open
C code files ds_arb_functions.c **Owner** **SCAT** **BRAT** 488 **Page** 325 **Line** 54

Problem description

(Comment #38)
Table 16-17, pg. 325, function data_coming
Data coming is also indicated by reception of DATA_NULL (see also process_requests, which treats DATA_NULL the same as DATA_PREFIX, and sets the in_packet flag, which halts auto increment of the rx FIFO).
Change pg. 325 line 54 from:

```
if ((portRarb[i] == DATA_PREFIX) || (portRarb[i] == SPEED)) {  
to:  
  
if ((portRarb[i] == DATA_PREFIX) || (portRarb[i] == DATA_NULL) || (portRarb[i] == SPEED)) {
```

Bug fix description

ID 93 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Can get into A1 and A2 without requesting port being set correctly **Status** Open
C code files ds_arb_functions.c **Owner** **SCAT** **BRAT** **Page** 326 **Line** 0

Problem description

arb_ok() can cause converted_request to be set true. However, requesting_port may not be set properly to point to the port which caused converted_request to be true. Seems like whenever async_pending or isoch_pending is set, we also need to capture which port was preferred so that arb_ok can set requesting_port when converted_request is also set.

Also, doesn't appear own_pending, isoch_pending, or async_pending are ever initialized.

Bug fix description

ID 65 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title arb_OK **Status** Open
C code files ds_arb_functions.c **Owner** **SCAT** **BRAT** 489 **Page** 326 **Line** 34

Problem description

(Comment #39)
Table 16-17, pg. 326, function arb_OK
If con_mgmnt_request is set, then initiate async arbitration, similar to the isbr flag.
At pg. 326 line ~34, insert the following just before "else if (breq == PRIORITY_REQ)":

```
else if (con_mgmnt_request)  
own_request = async_arb_OK;
```

Bug fix description

ID 123 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Cycle start starvation code doesn't grant senior port as intended **Status** Open
C code files beta_arb_functions.c **Owner** **SCAT** **BRAT** **Page** 327 **Line** 31

Problem description

The test_requests() routine within beta_arb_functions performs the 3 step check to avoid cycle start starvation. If the conditions are met for possible starvation, any asynchronous request is stomped by setting bap to NPORT and clearing the in_phase_async_request flag. However, the current code execution will land at the line stating "if (*best_port == NPORT) return (grant_link);"

This is incorrect in that we were intending to grant the senior port instead.

Perhaps the test for starvation was intened to be just `_before_` the test on `"in_phase_isoch_request || in_phase_async_request"`?

Bug fix description

ID 67 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title boss_end_packet_actions **Status** Open
C code files beta_arb_functions.c **Owner** **SCAT** **BRAT** 491 **Page** 328 **Line** 51

Problem description

(Comment #41)
Table 16-18, pg. 328-329, function boss_end_packet_actions
When granting link, need to set the grant_self flag as well as the link_concatenation flag.
At pg. 328 line 51 and pg. 329 line 36 (2 places), just after the `"*link_concatenation = TRUE;"` statement and before the `"return;"` statement, insert:

```
grant_self = TRUE;
```

Bug fix description

ID 129 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title process_requests **Status** Open
C code files process_req.c **Owner** **SCAT** **BRAT** 494 **Page** 334 **Line** 54

Problem description

(Comment #43) (cont)

The in_packet flag should only be set after we've passed tree-ID because it's possible to get data-prefix arb state on DS ports during tree-ID, but it doesn't indicate the start of a packet. Also, DATA_NULL can indicate the end of a packet as well as the start so set packet_ending flag if DATA_NULL recieved if in_packet already set.

Change pg. 334 lines 54-55 from:

```
if (!in_packet[i]) in_packet[i] = TRUE; // throttle the FIFO
else next_arb[j] = TRUE; // signal that this is the latest
```

to:

```
if (!in_packet[i] && PHY_state >= S0)
    in_packet[i] = TRUE; // throttle the FIFO
else {
    next_arb[j] = TRUE; // signal that this is the latest
    if (portCurrent'arb == DATA_NULL)
        packet_ending[i] = TRUE;
}
```

The received_speed_signal must be maintained til end of packet so that decode_phy_packet can use it.

At pg. 335 line 16 and line 37 (2 places), delete:
received_speed_signal = FALSE;

Bug fix description

ID 75 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title transmit_actions **Status** Open
C code files transmit_actions.c **Owner** **SCAT** **BRAT** 502 **Page** 347 **Line** 38

Problem description

(Comment #09)

1) Table 16-25, pg. 347-348, function transmit_actions

The immediate_request variable is used later in the function and should not be set FALSE until then. Delete pg. 347 line 38:

```
immediate_request = FALSE;
```

2) The code does not correctly reset an immediate request from a legacy link. Move pg. 348 line 7

```
immediate_request = FALSE;
```

to immediately before the while statement at pg. 348 line 15.

3) The while loop which processes the data_to_transmit from the link does not quantize/stretch the data-prefix symbols when PH_REQ_HOLD is indicated by the link (all symbols transmitted in a packet from the speed-signal sequence to the terminating ending symbols must be stretched or padded according to the packet speed). I suggest the following. Change pg. 348 lines ~16-21 from:

```
do { // Wait for data or release from the link
    wait_event(PH_BYTE_CLOCK); // intended to synchronize with clock
    PH_CLOCK_indication();
    data_to_transmit = waitPH_DATA_request();
} while (data_to_transmit.reqType == PH_REQ_HOLD);
// Hold only valid before data starts
if (data_to_transmit.reqType == PH_REQ_DATA) {
```

to:

```
// Wait for data or release from the link
waitPH_DATA_request(data_to_transmit);
if (data_to_transmit.DreqType == PH_REQ_HOLD) { // Hold only valid before data starts
    PH_CLOCK_indication();
    wait_next_symbol(tx_speed);
} else if (data_to_transmit.DreqType == PH_REQ_DATA) {
```

NOTE: An alternative is to change the portTarb function so that the packet speed is maintained (once determined) instead of changed to DEFAULT so that the tx_character function does the character stretching/padding automatically.

Bug fix description

1) [JH] Agreed in principle ... see next comment.

2) [JH] Agreed in principle. I like the pairing of clearing requests with indications to the link, however. So ultimately, BREQ and immediate_request are no longer cleared on page 347 ~38. Instead, when the legacy link is granted, both breq and immediate_request are cleared. The clearing of immediate_request at pg 348 line 15 stays as is as a consequence.

3) [JH] Needs further discussion, but preliminary thought is to disagree. Although the packet specifications may need clarification to allow this, there should be no reason to require a rigid number of symbols between the minimum amount of speed signaling and DP and the start of data. Should be okay to have this optional "HOLD" period be purely elastic. It does make a worse min/max PHY_DELAY jitter, but can improve latency in some cases. Certainly the PHY/Link interface is not required to insert a symbol's worth of hold cycles. For a legacy interface, holds come in 20 ns granularity. So pli block would have to buffer internally until the next clock_indication arrived. Not conclusive, but suggestive.

ID 118 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Speed filtering of Link? **Status** Open
C code files receive_actions.c **Owner** **SCAT** **BRAT** **Page** 349 **Line** 40

Problem description

When attached to a legacy link, a PHY needs to speed filter beta packets from being delivered across the interface. Particularly in receive_actions, data bytes should not be delivered and, when the end of a packet is reached, the link interface should stay "locked up". As such, fly_by_permitted can not be called in such a situation. Said differently, I think fly_by_permitted should only be called if the current packet format is legacy.

Similarly, when a beta parallel link is attached, packets >S800 must not be pushed across the interface.

Bug fix description

ID 116 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title receive_actions() fails to conclude null beta packets properly **Status** Open
C code files receive_actions.c **Owner** **SCAT** **BRAT** **Page** 350 **Line** 20

Problem description

if receive_actions() repeats a null packet, it always waits DATA_END_TIME. This is not correct if the null packet was of BETA format. (Such as a DATA_NULL concluded with GRANT.)

Also, start_rx_packet() appears to be borken in that it won't return if a DATA_NULL/GRANT sequence is received.

Bug fix description

ID 76 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title receive_speed_signal **Status** Open
C code files receive_actions.c **Owner** **SCAT** **BRAT** 504 **Page** 350 **Line** 52

Problem description

(Comment #45)
Table 16-26, pg. 349-351, function receive_actions
The receive_speed_signal needs to be cleared at end of processing.
At pg. 350 line 52, just before the "return;" statement and at pg. 351 line 17, just before the "}" (2 places), insert:

```
received_speed_signal = FALSE;
```

Bug fix description

ID 137 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Missing PMD services **Status** Recommend Accept
C code files phy_services.h **Owner** **SCAT** **BRAT** **Page** 276 **Line** 27

Problem description

For posterity, the following signals should be replaced with PMD services. Extremely low priority ... entered to allow sticky to be removed but not forgotten.

bias_detect, connect_detect, ds_portR, local_plug_present, rx_S200, rx_S400, signal_detect, ds_portT, ds_portTarb, ds_portTspeed, tpBias

Bug fix description

ID 45 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title loop_disabled_actions **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 459 **Page** 300 **Line** 23

Problem description

(Comment #19)
Table 16-7, pg. 300, function loop_disabled_actions

In the P12: Loop-Disabled state, the port is inactivated (see activate_connect_detect). The port will continue to send tones in order to maintain connectivity with its peer port.

The peer port, in turn, will lose synchronization, detect that it is no longer receiving a continuous signal, set its untested_fault flag and also transition into the P12: Loop-Disabled state.

In order that the the local port not immediately transition back into the the P11: Untested state, the port must wait for its peer port to become inactive before exiting.

Bug fix description

At pg. 300 line ~23, insert the following just after the "activate_connect_detect(0);" statement:

```
while (receive_ok && loop_disabled)
; // Wait til peer become inactive or loop_disabled flag is cleared.
```

ID 30 **Balloter** CWS **How submitted** Ballot comment **Fixed in version**
Title pkt and pkt_prefix **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** 471 **Page** 309 **Line** 32

Problem description

Notice the use of pkt and pkt_prefix. Requests are only expected when we aren't inside a packet (pkt) and when we aren't expecting a packet to begin (pkt_prefix).

Data is the logical inverse ... we are in a packet or expecting one to begin.

But comma is a little strange ... the code implies we expect a comma sometimes when we are in pkt_prefix (since we only ignore the comma inside of actual data).

Why isn't comma detection restricted in the same fashion as request types (since commas can only be inserted in a string of requests)? And if it does no harm to detect commas during the packet prefix, why not detect during "data" as well?

From Alistair:-

In normal operation a comma should not be received during packet or packet prefix or any request other than TRAINING or OPERATION. See 10.3.1.2.

So it would be consistent to make the comma decode conditional on !pkt and !pkt_prefix, same as requests.

You might want to think about what happens in abnormal operation if it is possible for the receiver to think it is pkt or prefix when the transmitter is trying to send a TRAINING request with commas. The comma will appear as an invalid to the rx but the rest of the TRAINING request will appear as valid, so does the rx ever get to realise it has lost sync??

Regards,
Alistair

Bug fix description

For consistency, comma can only be detected where it is expected ... in a request position. So qualification for detecting a comma has been enhanced to be conditional on !pkt and !pkt_prefix.

Alistair's second question is deeper and touches on a related topic ... why PTX3:PTX1 is needed. A new bug will be logged to deal with this aspect of the problem.

ID 136 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title incorrect comment in start_rx_packet() **Status** Recommend Accept
C code files receive_functions.c **Owner** **SCAT** **BRAT** **Page** 322 **Line** 5

Problem description

Line in start_rx_packet()

```
cur_format = LEGACY;                      // assume legacy format, if see a speed signal, might learn better
```

Comment seems incorrect. I don't think it is possible to have a non Legacy format packet if DATA_PREFIX is received before a speed code. Said differently, a leading DP is guaranteed to be legacy format packet. Comment should be removed.

Bug fix description

ID 69 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version**
Title process_requests **Status** Recommend Accept
C code files process_req.c **Owner** **SCAT** **BRAT** 494 **Page** 334 **Line** 52

Problem description

(Comment #43)
Table 16-19, pg. 334-335, function process_requests

Typo in comment, line 52, change from:
// flag to ignore the test of the packet
to:
// flag to ignore the rest of the packet

Bug fix description

Typo comment accepted

ID 127 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Legacy loop while already in P11:Untested **Status** Recommend Accept in principle
C code files node.c **Owner** **SCAT** **BRAT** [^] **Page** 287 **Line** 30

Problem description

If a border node finds a loop in the a cloud, it sets force_retrain on any beta port which is in P2:Active. The intent is to force the P2:P11 transition to P11:Untested. In P11:Untested, the first thing we do is force the far port to resync with us (causing him to break out of whatever he was doing). If successful, we then as a border send and LTS with ATTACH_IN_PROGRESS until the Legacy loop is resolved. Then we move into our normal testing loops.

The question, what should happen to a port which is already in P11: Untested when a legacy loop is discovered? Currently, the C code will do nothing. We'll continue to send whatever LTS we were sending, etc. And the far end may actually send us an ATTACH_REQUEST to which we will never be able to respond since we don't see bus_initialize_active ever go false. Consequently, the far end will be forced to send the bus reset which we can't hear either.

Bottom line seems to be we need to make sure the port exits early from any wait actions if it won't be able to process/hear the incoming ATTACH_REQUEST.

Bug fix description

A fix has been detailed on paper, but not yet entered into the C code.

ID 111 **Balloter** Biva **How submitted** Post ballot **Fixed in version**
Title test interval for no active ports **Status** Recommend Reject
C code files node.c **Owner** **SCAT** **BRAT** **Page** 288 **Line** 25

Problem description

The textual description in Sec11.7.12 (No Active Ports) indicates that the test interval may be set to 0 if the PHY currently has no ports in the active state. Is this supported by the C-code ?

Bug fix description

I don't believe the decribed behavior is desired or necessary. The text should be changed. A related discussion is in the e-mail thread "RE: isolated nodes and such". Basically, we now allow single-port PHY's to skip the establishing dominance phase. Isolated nodes with multiple ports do not skip the dominance phase because they may still cause a loop or suffer from a race condition in which they send an attach out port #0 at the same time a connection to another isolated node reports an attach_request inbound on port #1. Now we may form a loop, etc. if we aren't really careful. Since it doesn't take an isolated node long to establish dominance, we opted not to make the optimization and deal with all of the hazards. So I consider this closed provided Wooten updates the text.

ID 113 **Balloter** Biva **How submitted** Post ballot **Fixed in version**
Title senior_port in self_ID **Status** Recommend Reject
C code files self_id.c **Owner** **SCAT** **BRAT** **Page** 342 **Line** 11

Problem description

In self_ID_transmit_actions() (P.342, L.11) the "senior_port = parent_port;" should, in my opinion, be replaced with
senior_port = (root || Beta_mode[parent_port]) ? NPORT : parent_port;
// We can neither set it blindly to NPORT, nor to parent_port.

Bug fix description

[JH] Similar bug reported as Skid #8 (C code bug #72) which blindly sets senior_port to NPORT. senior_port is then updated later by decode_phy_packet if necessary. Basically, senior_port points toward proxy_root the same way parent_port points to root. senior_port == NPORT only at proxy_root. The fix is checked in as PRN #372

The solution takes the approach: at the end of tree-id, parent_port == senior_port and root == proxy_root, so the two "trees" are coincident. During self_id, if a border is advertised, proxy_root and senior_port are updated. Whenever senior_port is assigned, so is proxy_root to enforce that one implies the other. In fact, proxy_root == TRUE iff senior_port == NPORT.

ID 178 **Balloter** **How submitted** **Fixed in version**
Title Cmnd 4 clears Standby_Fault **Status** Done
C code files **Owner** MT **SCAT** **BRAT** 0 **Page** 251 **Line** 23

Problem description

add "and Standby_fault"

Bug fix description

ID 179 **Balloter** **How submitted** **Fixed in version**
Title BIAS_FILTER_TIME **Status** Done
C code files **Owner** MT **SCAT** **BRAT** 0 **Page** 254 **Line** 22

Problem description

Delete from here - its defined in table 11-3

Bug fix description

ID 180 **Balloter** **How submitted** **Fixed in version**
Title delete BIAS_HANDSHAKE **Status** Done
C code files **Owner** MT **SCAT** **BRAT** 0 **Page** 254 **Line** 25

Problem description

Delete from this table, (a) because it is a connection management constant and (b) is superseded by RECEIVE_OK_HANDSHAKE

Bug fix description

ID 226 **Balloter** Jim Skidmore **How submitted** Post ballot **Fixed in version** D103
Title C code tidbit in self_ID_grant_actions() **Status** Done
C code files self_id.c **Owner** JH **SCAT** **BRAT** 0 **Page** 340 **Line** 29

Problem description

In self_id_grant_actions, braces and additional test of active flag are needed to get correct arbitration signal sent on ports which are not the lowest_unidentified_child port.

Bug fix description

Change from:

```
for (i = 0; i < NPORT; i++) {  
    if (!all_child_ports_identified && (i == lowest_unidentified_child))  
  
        if (!proxy[i]) portTarb(i, GRANT); // Send grant to lowest unidentified child (if any)  
        else portTarb(i, DATA_PREFIX); // Otherwise, tell others to prepare for packet  
    }  
}
```

to:

```
for (i = 0; i < NPORT; i++) {  
    if (!all_child_ports_identified && (i == lowest_unidentified_child))  
    {  
        if (!proxy[i]) portTarb(i, GRANT); // Send grant to lowest unidentified child (if any)  
        } else portTarb(i, DATA_PREFIX); // Otherwise, tell others to prepare for packet  
    }  
}
```

ID 142 **Balloter** **How submitted** **Fixed in version**
Title Max. rise/fall time **Status** Done
C code files **Owner** EH **SCAT** **BRAT** 0 **Page** 102 **Line** 0

Problem description

Table 6-1 in Draft 1.00 lists max rise time at S800 as 600 ps. This rise time, if used, won't allow the eyd diagram to be wide enough. Comparing to FibreChannel, this should be 400 ps.

Bug fix description

ID 216 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title max. rise/fall time **Status** Done
C code files **Owner** EH **SCAT** **BRAT** 0 **Page** 102 **Line** 43

Problem description

Table 6-1 in Draft 1.00 lists max rise/fall time at S800 as 600 ps. This rise time, if used, won't allow the eye diagram to be wide enough. Comparing to FibreChannel, this should be 400 ps. The rise/fall times for the other speeds should be similarly scaled.

Bug fix description

ID 140 **Balloter** CWS **How submitted** Post ballot **Fixed in version**
Title Active nodes and ports **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 53 **Line** 31

Problem description

no such thing as an active node, only active ports.

Bug fix description

ID 145 **Balloter** **How submitted** Post ballot **Fixed in version**
Title misuse of "active" and "inactive" **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 141 **Line** 42

Problem description

Bug fix description

Reworded

ID 211 **Balloter** DW **How submitted** Post ballot **Fixed in version**
Title Clarification of scrambling **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 156 **Line** 9

Problem description

What is the significance of the D28.0 codeword? There is no place in the spec or here where I can find anything that indicates that the D28.0 codeword has any reason to occur during training. There is no explanation of how a continuous stream of Training requests (00000000b) will ever be scrambled into 00111000b (D28.0) so that it can be turn into a K28.5. Also missing is a description of how the synchronization of the scrambler works (lsb of the transmit represents the lsb of the scrambler and the decoded word is supposed to be 0 so if you get something other than 0 you should change the bit, or something like that.)

Bug fix description

Extra sentence added to 10.2.6.1.4
Note added to 10.2.10 on how the descrambling works.

ID 200 **Balloter** **How submitted** **Fixed in version**
Title table 10-15 inappropriate **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 165 **Line** 47

Problem description

Table 10-15 has very little to do with the port state machines

Bug fix description

Replace with a table which describes the variables used in the port state machines

ID 210	Balloter DW	How submitted Post ballot	Fixed in version
Title	POR values for PHY registers	Status Done	
C code files	Owner CWS	SCAT <input checked="" type="checkbox"/> BRAT 0	Page 234 Line 0

Problem description

I need all values in the PHY registers to have some power-on value even if it is vendor dependent. Particular are PHY-ID and R bit. They don't have any power on values specified. I suppose that this is because they are not used until a bus reset is sent and they would get set there. However, with the PIL-FOP model, the PIL might use the values in the FOP before any other port comes up. So, I would like the PHY-ID to initialize to zero and the R bit to be set to 1.

Would also like some definition of the gap count sticky-bit in the spec and think that PHY Register Chapter is as good a place as any (we refer to the sticky bit in several places but we don't define what it is anywhere.) When we define it, we should state what its POR value is.

Bug fix description

B_link to vendor-dependent

Max_legacy_path_speed to 000 (also logging a bug that this is not initialised - needs to be reset in R0)

Physical_ID I think should be power reset to 63 (the bus is mal-configured until the first time it is configured!!) - this is different from your suggestion!

PS to vendor-dependent

R to 1 (PHY always comes up as an isolated node)

AStat and BStat as 00

Child 1 (interesting this, disconnected or disabled ports are always "children" - I'm clarifying the text on this)

Negotiated_speed as 000

Receive_OK as 0

ID 167	Balloter	How submitted	Fixed in version
Title	enab_accel and enab_multi	Status Done	
C code files	Owner CWS	SCAT <input checked="" type="checkbox"/> BRAT 0	Page 234 Line 19

Problem description

Power reset values of enab_accel (0) and enab_multi (1), are different. However description of both indicates they are only used for backwards compatability with legacy links, in which case they should fit the 1394a spec. The 1394a spec has them both power up '0' unless hardware strap options select otherwise. Don't see how 1394b came up with different power up values for these two bits.

Bug fix description

ID 174 **Balloter** **How submitted** **Fixed in version**
Title ref to 1394a **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 238 **Line** 14
Problem description
This should be a ref to 15.5.1, not 1394a.
Bug fix description

ID 184 **Balloter** **How submitted** **Fixed in version**
Title A0:A1 and A0:TX contention **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 270 **Line** 14
Problem description
When a collision occurs, arb_OK() returns true immediately. If immediate_request() is TRUE at the same time, both A0:A1 and A0:TX could test true.
Bug fix description

ID 185 **Balloter** **How submitted** **Fixed in version**
Title link_concatenation now unnecessary **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 270 **Line** 33
Problem description
link_concatenation is replaced with grant_self on all state transitions. Fixes problems with mutual exclusions and guarantees that a legacy link concatenation is not interrupted. See C code bug # 77.
link_concatenation is unnecessary since grant_self is now set for cases of concatenation. The TX:TX transition can now be written as end_of_packet && grant_self
Bug fix description

ID 186 **Balloter** **How submitted** **Fixed in version**
Title RX:A0 and RX:RX **Status** Done
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 270 **Line** 51
Problem description
RX:A0 and RX:TX are not exclusive. Outer parentheses are not conventional. Suggested replacement is:
!concatenated_packet && !(fly_by_OK || grant_self) && requesting_port == NPORT
Bug fix description

ID 217 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title incorrect comment in rx_sync_error declaration **Status** Recommend Accept
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 306 **Line** 30

Problem description

boolean rx_sync_error; // FALSE if failed to complete synchronization
should be // TRUE if failed to complete synchronization

Bug fix description

ID 214 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title Bport receiver misses first request. **Status** Recommend Accept
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 312 **Line** 23

Problem description

there is a case where the bport receiver will miss the first request sent by its peer.

The case is when the peer has been sending requests before the receiving port enters PRX4:Receive. The variables localR and last_localR have been identical for more than one symbol.

When the code evaluates these variables at the end of receive_actions it determines that there is no first occurrence and does not push the arb request onto the FIFO.

Bug fix description

Line added to top of bport_receive_actions().

ID 213 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title Misleading comments in end of bport_receive_actions() **Status** Recommend Accept
C code files **Owner** CWS **SCAT** **BRAT** 0 **Page** 313 **Line** 12

Problem description

bport_receive_actions() had some older comments regarding "control signal" and "config request symbol". Control and config requests have all been mapped into ARB_STATES, and ARB_REQUESTS remain independent.

 // Buffer first occurrence of each new control signal....

(stuff deleted)

 // Buffer first occurrence of each new config request signal....

Bug fix description

Code changed to:

 // Buffer first occurrence of each new arbitration state....

(stuff deleted)

 // Buffer first occurrence of each new arbitration request....

ID 209 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title Gap_count_reset_disable is not initialised **Status** Recommend Accept
C code files reset.c **Owner** CWS **SCAT** **BRAT** 0 **Page** 337 **Line** 10
Problem description
should be set to FALSE in arb_power_reset
Bug fix description

ID 68 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D103
Title gap_repeat_actions **Status** Recommend Accept in principle
C code files beta_arb_functions.c **Owner** CWS **SCAT** **BRAT** 492 **Page** 330 **Line** 15
Problem description
(Comment #42)
Table 16-18, pg. 330, function gap_repeat_actions
The token_receive_port variable must be reset when done.
At pg. 330 line 15, just before the terminating }, insert:

token_receive_port = NPORT;
Bug fix description
Jim's fix had a danger of clobbering token_receive_port too soon in the case of an ASYNC_START followed immediately by ARB_RESET*. So a different fix was implemented.

ID 143 **Balloter** **How submitted** **Fixed in version**
Title misuse of 1394-1995 and 1394a **Status** Done
C code files **Owner** ?? **SCAT** **BRAT** 0 **Page** 126 **Line** 3
Problem description
This bullet is wrong, the PMD does not support either 1394-1995 or 1394a!
Bug fix description

ID 144 **Balloter** **How submitted** **Fixed in version**
Title Mis-use of the word Active **Status** Done
C code files **Owner** ?? **SCAT** **BRAT** 0 **Page** 126 **Line** 4
Problem description
(in two places)
Bug fix description

ID 77 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title link_concatenation, grant_self, and TX:A0 and TX:TX mutual exclusion **Status** Recommend Accept
C code files various **Owner** **SCAT** **BRAT** **Page** 0 **Line** 0

Problem description

If TX:A0 = TRUE && grant_self = TRUE, then TX:TX is true as well. While investigating, it became clear that the distinction between link_concatenation and grant_self wasn't clear. In BEOP, link_concatenation was being set if the link was to be granted, but grant_self was not being set. As a result, the RX:TX transition would not be taken as intended if BEOP was called from within receive_actions.

Also, it was noted that a con_mgmnt_request could be granted during a true legacy link concatenation. It is unclear if this would confuse legacy links.

Bug fix description

The proposed fix is detailed and included in the modified C code. In essence, link_concatenation will only be used/set when a legacy link uses the hold cycle to request a concatenation. transmit_actions will then prefer a link_concatenation to other transmit requests. Grant_self will replace link_concatenation on the state machines for transitions to and from TX/RX.

Link_concatenation is no longer required as parameter to be passed into beop or con_mgmnt_granted.

Note State machine changes required as per above

ID 79 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title Initialization of portT, introduction of dsport_active **Status** Recommend Accept
C code files dsport_rx.c, dsport_tx.c, port.c, shared.h **Owner** **SCAT** **BRAT** **Page** 0 **Line** 0

Problem description

portT is not initialized anywhere for a DS port ... all of the upstarts code initializes just for beta ports. Furthermore, dsport_transmit_actions() runs all of the time after bus reset, even if the port hasn't yet been marked as active, connected whatever. In simulations, the combined consequence is that at the instance the port is marked connected, D/S bits come rushing out of port immediately.

bports don't suffer because portT is properly initialized, and because the bport_tx machines stay in an inactive state until bport_active is set true.

Bug fix description

A symmetric solution is proposed. Introduce dsport_active which starts and stops dsport_transmit_actions. This is easily done by changing the opening while to "while (power_reset or !dsport_active) "

(I used the word "or" above rather than the vertical bars because the bug tracker complained.)

Then, in every circumstance in port.c where bport_active is assigned, also assign dsport_active as required symmetrically. And just prior to asserting dsport_active, make sure portT is set to a valid value.

ID 86 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title dsport_active and PMD...(SELECT_DS_PORT) not symmetric with bport_a **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 0 **Line** 0

Problem description

dsport_active not being set and cleared symmetrically with bport_active. Both are intended to control operation of FIFO and data/arb transmitter. Neither affects toning or bias/connection detection.

Bug fix description

adjust to make symmetric - see also #79

ID 96 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title revamp of loop healing **Status** Recommend Accept
C code files various **Owner** **SCAT** **BRAT** **Page** 0 **Line** 0

Problem description

Skid raised a number of issues with loop healing, Wooten added some features as part of DevCon presentation, and Hauck uncovered some possible errors in related pieces of code. The C code is being updated to match the revised DevCon presentation and does include optimizations for single port PHY's and isolated nodes.

Bug fix description

1) in_test_interval is no longer required. Effectively, the node controller can simply check !need_new_LTP && test_interval_timer >= TEST_INTERVAL to track if a particular holding register value has been in effect for the required time.

2) In initialize_arbitration(), it is incorrect to clear restore_request and con_mgmt_request. Doing so causes these flags to be reset with each bus reset, possibly before they have been serviced. For example, if we still need to process a restore when an unrelated bus reset occurs, we will forget that we have a restore to do later. Nothing at that point causes restore_request to be set again.

3) con_mgmt_request has been renamed to loop_test_request. And rather than a set/clear flag used as a semaphore between otherwise async processes, it is asserted to continuously by the loop_tester until the need to test is removed (either by completion of the test or by loss of the test port). Consequently, it is only driven in loop_test_interval_actions().

4) tx_format and tx_speed are presumed to hold the format and speed of the current or last packet transmitted. This is important so that the rule re: S100 downshifts can be implemented correctly in boss_end_packet_actions(). So any time we transmit _any_ packet on the bus, we should set these correctly to enable the downshift test. Consequently, tx_format and tx_speed are now set within con_mgmt_granted().

5) restore_request is now dealt with independently from loop_test_request. A restore_request will cause arbitration without needing to also set loop_test_request. It was much simpler to handle them as separate cases.

6) Processing of received ATTACH_REQUESTS is now done differently. Each port implements an attach variable which is set when the port is ready to attach for any reason, including receipt of an ATTACH_REQUEST. This ATTACH flag "arms" the port and is used in reset actions to begin propagation the reset, much like resume is used. The node controller will ask to release the bus as a direct or indirect result of any ATTACH bit being set. So con_mgmt_granted simply waits for the indication from the loop_test_interval_actions() before releasing the bus. received_attach is obsolete.

7) Skid's comment #12 is basically accepted.

8) As discussed in Skid's #13, test_port_selector had several race conditions. It was rolled into loop_test_interval_actions(). However, the part that monitored whether the arb state machine was stuck in a loop was retained but renamed to loop_detector().

9) loop_test_interval_actions() now uses a new function attach_pending() to determine if there are unfinished attaches to complete.

10) loop_test_interval_actions() completely revamped. Based on skid's #14 and David's DevCon presentation. All asynchronous handshakes strengthened. Special support for single port PHY's added per DevCon.

11) Untested_actions() completely revamped per Skid's #18 and the Devcon presentation. The concept of a single_port_node was introduced and allowed to head straight to attach_request (no LTS sent). And isolated_node arms the attach immediately and then lets the node controller release the bus. This fixes a bug in which we were tying up the PHY/Link interface for >> a max packet time. Most asynchronous handshakes strengthened to avoid feared race conditions. An the concept of an attach flag was introduced to arm the port such that reset.c would propagate the joining reset on those ports (use to be just active or resuming ports).

12) In process_req.c, separately specified the requests for connection management: now loop_test_request and restore_request. D1.01 code didn't properly cause a request to be asserted for restores.

13) For loop testing, single_port_phy's and isolated_nodes no longer hold onto their local bus after sending an attach_request. Much like for resume, reset_detected() has been modified to detect the incoming reset on the test port in these cases.

14) To propagate resets on newly attaching ports, reset_start_actions() and reset_wait_actions() were modified to work on ports with the attach flags set. There is still as sort of race condition here ... once a port sees bus_initialize_active, it will clear attach and head off to active. So for a brief period, both attach and active are false. But before reset_start_actions can complete, we should make it to active. As an observation, we shouldn't ever find ourselves in reset_wait with the attach flag set.

15) transmit_actions() was modified to account for canceled link requests and loop_test_requests. It was observed that Beta link requests could be cancelled at any moment such as just after boss_end_packet_actions decided to jump to TX, or after start_tx_packet, etc. So we have to be careful not to use our entry into TX for the wrong packet type. (Like we entered for an immediate_request which got canceled and then tried to do a loop test packet. Invalid because we didn't wait for a subaction gap.) So at entry into TX and at each wait thereafter until the link is granted, the various possible transmit requests are prioritized and considered. This mechanism also allows the removal of loop_test_request by the node controller without worrying about what the arb state machine is doing. If we came into TX because of a loop_test_request which is no longer present, we gracefully terminate with a null packet if we can't make use of the visit to TX for anything else.

16) process_requests would only clear in_packet if the last symbol had set packet_ending AND there was something else in the FIFO. As a result, the clearing could be delayed for some time and, if the port status changed from !active to active, the FIFO would start throttling itself preventing the reset machine from seeing the bus reset which would follow an attach request, for example.

ID 23 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D103
Title DS arbitration signal encodings **Status** Recommend Accept
C code files data_structures.h, dsport_rx.c, dsport_tx.c **Owner** **SCAT** **BRAT** 415 **Page** 274 **Line** 7

Problem description

DS arbitration signals are overloaded, and the encodings do not reflect this

These will have to be assigned numerical values to deal with the overloading
Decode does not reflect overloading, encode with have to be updated

Overloading of ds arb states can not be easily resolved at the cable port; it needs to be at the arb state machine side of the FIFO per skid's recommendation. Additionally, there is no RX_DATA arb state ... a stopped clock detector is needed.

Bug fix description

Also BRAT #s 416, 460, 463

Process_req has been modified to use PHY_state to filter and resolve the overloading. And a caution about filtering of spurious RX_DATA_PREFIX indications has been added to dsport_rx.c. Also, RX_DATA_PREFIX kicks off the extract clock circuit and a stop clock detector is used to determine when to push the ending arb state into the FIFO.

ID 24 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title RESET_WAIT **Status** Recommend Accept
C code files data_structures.h **Owner** **SCAT** **BRAT** 418 **Page** 274 **Line** 17

Problem description

(Comment #11)
Add RESET_WAIT to enumerated list of wait_times.

Bug fix description

Agreed. Note, while not strictly referenced in the C code, RESET_WAIT is referenced on the state machine. However, it is not used directly ... it is summed with a variable named reset_duration.

ID 25 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D103
Title Arb_state enum values **Status** Recommend Accept
C code files data_structures.h **Owner** **SCAT** **BRAT** 417 **Page** 274 **Line** 27

Problem description

These values will have to be re-assigned, to deal with the overloading which is used for DS arbitration line states. Also one or two new ones will have to be introduced to deal with the fact that in DS mode, different encodings are used for RX and TX versions of the same arbitration states

Bug fix description

See # 23 (BRAT #415)

ID 22 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title PMD services need defining properly **Status** Recommend Accept
C code files data_structures.h, phy_services.c, port.c **Owner** **SCAT** **BRAT** 419 **Page** 276 **Line**

Problem description

A proper service interface should be used for PMD, rather than the use of variable shared between several modules and the PMD.

It needs to be made clear which port machine is controlling the PMD at any given time (i.e. the control signal for a PMD mux).

Bug fix description

Also BRAT #s 434, 435, 436, 437, 438, 440, 442, 447, 449, 450, 451, 453, 454
Various C code modifications proposed for "to PMD" direction, similar changes also required for the "from PMD" direction.

ID 26 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title reset_notify clearing **Status** Recommend Accept
C code files phy_services.h, port.c **Owner** **SCAT** **BRAT** 422 **Page** 277 **Line** 23

Problem description

Deal more sanely with the issue of reset_notify clearing on successful transfer to the link,

Bug fix description

Also BRAT #s 456, 457, 495

insert

```
void PH_EVENT_response(boolean event_OK);
```

amend port.c

```
PH_EVENT_indication(missed_reset ? PH_RESTORE_RESET : PH_RESTORE_NO_RESET, 0, 0);  
// PHY/Link interface is required to provide PH_EVENT_response of event_OK  
// to clear the missed_reset flag  
}  
in_standby = FALSE;  
}
```

```
void PH_EVENT_response(boolean event_OK) {  
if (event_OK) missed_reset = FALSE;  
}
```

amend comment in reset_start to

```
PH_EVENT_indication(PH_BUS_RESET_START, 0, 0); // Optional upon reentry to R0 from R1  
// PHY/Link interface is required to provide PH_EVENT_response of event_OK  
// to clear the missed_reset flag
```

ID 32 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title sync_error_signal and sync_lost_signal variables **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** 423 **Page** 278 **Line** 36

Problem description

(Comment #02)

The sync_error_signal and sync_lost_signal variables are per-port-variables, and should be declared as arrays, or moved to the port.c module (sec 16.2.2, pg. 290).

Bug fix description

Not appropriate in port.c since shared among modules ... #ifdef subscribed region added to shared.h

ID 35 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title bport_active **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** 425 **Page** 279 **Line** 19

Problem description

(Comment #03)
The bport_active variable is a per-port-variable, and should be declared as an array, or moved to the port.c module (sec 16.2.2, pg. 290).

Bug fix description

Not appropriate in port.c since shared among modules ... #ifdef subscripted region added to shared.h. Also corrected for newly added dsport_active signal.

ID 33 **Balloter** Clay E Hudgins **How submitted** Ballot comment **Fixed in version** D1005
Title bport_active **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** 424 **Page** 279 **Line** 20

Problem description

Consider replacing "should" with "shall."

Bug fix description

Note, this is a mechanism, not a requirement
Amend comments to:-
boolean bport_active; // Set to true to instruct the Beta port to commence operating
 // Set to false to instruct the Beta port to cease operating

and similar change made to the dsport_active signal

ID 8 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title inconsistent use of "the" and "main" in comments **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** **Page** 279 **Line** 22

Problem description

"the" and "main" are used inconsistently in section comments

Bug fix description

amend appropriately

ID 4 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title loop_to_detect declaration **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** 426 **Page** 280 **Line** 28

Problem description

loop_to_detect declaration is alphabetically out of place

Bug fix description

move down four lines

ID 5 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title bportT declaration **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** **Page** 280 **Line** 47
Problem description
bportT is not used in process_req.c
Bug fix description
change the comment
// shared between main arb state machine, port.c, node.c, bport_tx.c and dsport_tx.c

ID 6 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title reference to process_req.c **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** **Page** 280 **Line** 53
Problem description
references are made to the c code file name process_requests.c
Bug fix description
change the comments (several places) to process_req.c

ID 7 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title arbreqTdeclaration **Status** Recommend Accept
C code files shared.h **Owner** **SCAT** **BRAT** 427 **Page** 282 **Line** 25
Problem description
arbreqT is used in port.c
Bug fix description
add port.c to the comment

ID 34 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title restore_port **Status** Recommend Accept
C code files node.c **Owner** **SCAT** **BRAT** 428 **Page** 284 **Line** 23
Problem description
(Comment #04)
Table 16-6, pg. 284, function restore_port
For-statement incorrect.
Bug fix description
Change pg. 284 line 23 from:

for (j = 0; j++; j < NPORT)
to:

for (j = 0; j < NPORT; j++)

ID 84 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title send_LTP() does not produce LTP format defined in 15.5.4 **Status** Recommend Accept
C code files node.c **Owner** **SCAT** **BRAT** **Page** 285 **Line** 3

Problem description

15.5.4 shows 3F in the PHY_ID position for an LTP. However, send_LTP() fills that field with zeros.

Bug fix description

Insert
tx_phy_pkt.phy_ID = 0x3F;

ID 90 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title "standby_reset" should be "missed_reset" in comment **Status** Recommend Accept
C code files node.c **Owner** **SCAT** **BRAT** **Page** 286 **Line** 19

Problem description

Bug fix description

change it

ID 133 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1006
Title False dominance detection in loop healing **Status** Recommend Accept
C code files node.c **Owner** **SCAT** **BRAT** **Page** 287 **Line** 0

Problem description

If HR_MODE == ATTACH_REQUEST because a remote controlling node is performing an attach, chances are that a local test port will be selected and stay selected because it thinks it is dominant if a full 8 bit compare is done with the received LTS. Then when the HR_MODE is changed back to LTP_TEST, the selected test might not be dominant any longer, but will take a full TEST_INTERVAL to determine that.

Bug fix description

Fixed in revised C code

ID 114 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1006
Title Sending ATTACH_IN_PROGRESS LTP can negate short bus reset benefit **Status** Recommend Accept
C code files node.c **Owner** **SCAT** **BRAT** **Page** 288 **Line** 29

Problem description

When a PHY decides to send an ATTACH_REQUEST, it does so at the same time it requests to send the ATTACH_IN_PROGRESS LTP. The time to send the LTP in the current spec is 160ns + 660ns + 160ns. Additionally, before a BUS_RESET could be sent, another call to start_tx_packet for 160ns is required. As a result, once ATTACH_REQUEST is scheduled, the PHY can't head off to R0 for another 1.14 us.

If the node receiving the ATATCH_REQUEST responds with a short bus reset quickly, it will expect the peer PHY to be in R0 within 1.26 us. Otherwise, the short RESET turns into a long reset.

This leaves 100 ns margin at best. Consequently, the ATTACH_REQUEST should be delayed until the LTP has been sent.

Bug fix description

Fix was slightly different than proposed. The send_LTP and con_mgmnt_granted loop routines were rewritten such that send_LTP always concludes with a call to start_tx_packet. As a consequence, when a node is ready to send the ATTACH_IN_PROGRESS LTP, it will only take 660ns + 160ns + 160ns before R0 can be visited. This leaves >260 ns margin which seems sufficient.

ID 18 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title definition of autocrossover_supported **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 433 **Page** 289 **Line** 17

Problem description

This is an implementation dependent constant

Bug fix description

Change from boolean to a #define

ID 89 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title crossover can now be a local variable within connection_status **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 289 **Line** 32

Problem description

Only used locally, now that we have a PMD service for it.

Bug fix description

move definition

ID 19 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title Min_port_speed comment **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 432 **Page** 289 **Line** 44

Problem description

Min_port_speed was deleted from the port register map during SCAT review. This comment needs to be updated to reflect this decision.

Bug fix description

update comment

ID 27 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title Delay in activate_connect_detect **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 439 **Page** 290 **Line** 16
Problem description
Clarify use of delay in activate_connect_detect
Bug fix description
add comment
// also ensures handshake times for both modes

ID 20 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title use of Beta_mode_only **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 290 **Line** 46
Problem description
comment does not reflect the name change of Beta_mode_only to Beta_mode_only_port
Bug fix description
update comment

ID 40 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title signal_detect race condition **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 444 **Page** 291 **Line** 30
Problem description
(Comment #15)
Table 16-7, pg. 291, function signal_detect_OK
Eliminate possible race conditions with signal_detect_monitor, avoids setting
sd_detected FALSE for a short period (which adversely affects other code which
monitors this flag).
Bug fix description
Change line 30 from:
if (x) sd_detected = FALSE;
to:
if (x) sd_detected = signal_detect;
[JH] Agreed, but changed comment as well.

ID 39 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title Toner **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 443 **Page** 291 **Line** 47

Problem description

(Comment #05)
Table 16-7, pg. 291, function toner
For-statement incorrect.

Bug fix description

Change pg. 291 line 47 from:

for (i = 0; i++; i < 3) { // send three bits or spaces
to:

for (i = 0; i < 3; i++) { // send three bits or spaces

ID 115 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1006
Title Compatibility of increased speed toning with future standards **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 291 **Line** 47

Problem description

For compatibility with future inteconnects (serial ATA), the current spec increased the number of speed tones which would be collected for the received speed from 3 to 6. However, the number of speed bit tones sent before signaling EVT_SENT_SPEED is still 3. As a consequence, it is possible for a node to send its last speed handshake with the ACK bit and, after the 3 speed bit, call start_port(). This would cause the receiver to see garbage in the last 3 bits of the expected 6 bit speed field. Seems undesired, so perhaps EVT_SENT_SPEED should not be sent until the 6th speed bit (defined to be zero in our standard) is sent.

On a similar note, the received_speed variable in port.c is defined to be of type speedCode which, in turn, is an enumerated type with 7 values. However, the code attempts to load received_speed with 6 bits. Don't know if this is legal in C, but it isn't legal for VHDL. Seems like received_speed would have to be declared as an int, or that speedCode be defined to have upto 2^6 values.

Bug fix description

Fix was to wait until after all 6 bits were sent before EVT_SENT_SPEED or EVT_SENT_ACK are issued. Nothing done about the C code issue.

ID 42 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title received_speed_indication **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 448 **Page** 292 **Line** 22

Problem description

(Comment #06)
Table 16-7, pg. 292, function receive_speed_indication
For-statement incorrect.

Bug fix description

Change pg. 292 line 22 from:

for (i = 0; i++; i < 6) { // now look for six speed code bits
to:

for (i = 0; i < 6; i++) { // now look for six speed code bits

ID 28 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title receive_speed_indication **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 446 **Page** 292 **Line** 25

Problem description

receive_speed_indication looks for a start bit. If it finds one, then it samples the next 6 bit positions and constructs "rs" which holds the decimal equivalent of the binary tones. If a start bit tone was followed by no speed encoding, then rs = 0 after the 6 samples. As a result, received_speed is set to -1 (received_speed = rs - 1;) and EVT_RECEIVED_SPEED is signaled.

It may be argued that listening_for_speed will never be true unless both ends of a connection have already shifted into sending a speed_code. If so, then we should never run into this problem (i.e., at least one speed bit will always be set when we are listening). But the proof escapes me and it seems real implementations may end up sending a few empty tone frames before getting their speed signals out.

Bug fix description

Obviously correct solution would be to bracket the last two executable lines of received_speed_indication with if rs > 0 . That is, only calculate received_speed and trigger EVT_RECEIVED speed if some real speed bits were found.

ID 88 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title BIAS_HANDSHAKE -> RECEIVE_OK_HANDSHAKE **Status** Recommend Accept
C code files data_structures.h, port.c **Owner** **SCAT** **BRAT** **Page** 292 **Line** 45

Problem description

One occurrence of BIAS_HANDSHAKE remains in port.c and data_structures.h which should be replaced/removed. Also, Table 11-3 should have the comments merged and BIAS_HANDSHAKE removed. Also, remove from Table 15-8

Bug fix description

replace/remove

ID 102 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title negotiated_speed - no power_reset value given **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 293 **Line** 38

Problem description

No power_reset value given for negotiated speed. C code / comments seem ambiguous on whether: value should start at 000, and go upwards as port is initialized, or value should start at max_port_speed, and go downwards as port is initialized

Bug fix description

Power reset value is implementation dependent, and negotiated_speed is set only once speed negotiation is complete. See definition of port_speed on p280 I35-37

Propose clarifying by

1) adding comment

p293 I28

// set Negotiated_speed in port register map to port_speed

and

2) possibly also add some text into the description of S3:S0 on p266 I34 and S4:A0b on p267 I14 to say that this is the point at which Negotiated_speed is set to port_speed[xxx]

ID 80 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title Loss of DC Connection Not Noticed, never tones again **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 294 **Line** 30

Problem description

Consider Node B which is connected to Node D with a DC path. Furthermore, assume Node D is off. Node B will attempt toning. After debouncing the DC connection and hearing no return tones, it will try bias. After a return bias is not received, the current C code leaves Node B listening for tones, but not sending tones. (Is this intentional ... I can't remember?)

Now if the DC path is removed, I don't think the C code takes note. It will still not initiate toning. Now a new connection (DC or otherwise) can be made to Node B without any notice and again without cause Node B to tone. If the new connection were made to another Node in the same state as Node B, then neither would initiate toning and the connection wouldn't become active until a power reset.

Seems like either Node B needs to revert to toning if bias is unsuccessful, or the loss of a DC connection should be noted and somehow cause toning to be turned back on.

All of this assumes local_plug_present was not implemented.

Bug fix description

Insert two lines before line 30 on p294

```
if (!dc_connected) toning = TRUE; // might have been waiting for dc_connected
// peer to wake us up
```

ID 43 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title connection_status race condition **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 452 **Page** 294 **Line** 45

Problem description

(Comment #17)
Table 16-7, pg. 294, function connection_status
Avoid race condition in transition to untested state.
At pg. 294 line ~45, insert following just before return statement:

```
while (!untested_state) // wait till in untested_state
;
```

Bug fix description

[JH] Accepted. Removed comment about race condition from code as well.

ID 97 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title receive_ok late when taking P0 -> P11 -> P2 **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 295 **Line** 51

Problem description

For single port PHY's, the time spent in P11 is quite short (just the time to achieve synchronization and to send the ATTACH_REQUEST). If two such peer ports enter P11 at maximally different times (one port goes to P11 after sending the last bit of it's speed, the other port is waiting for all 6 speed bits before declaring EVT_RECEIVED_SPEED ... so diff is 3 tone bit times or so), then receive_ok may not yet be set on when P11:P2 is taken.

Note that receive_ok is set when the local port heads to P11, but depending on the time constants, etc. receive OK may shut off before the far port starts signaling. And the start up time for receive_OK once signaling is heard was specified to be 2 calls to signal_detect_OK.

This might only be a sim problem when connectivity time constants are shortened (allowing receive_ok to be cleared quickly), but isn't obviously correct as written.

Bug fix description

Basic change was to connection_status(). When in the active, untested, resuming states where we have already determined we have (or had) a continuous signal, let receive_ok simply be a function of bport_sync_ok. This means it will be set properly before heading to P2:Active and also means that loss of sync will quickly send us out of P2.

ID 91 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title con_mgmnt_request set for restore? **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 298 **Line** 9

Problem description

Couldn't figure out how a restore would cause con_mgmnt_request to be set. Seems like at the time restore_request is set by restore_actions, con_mgmnt_request would need to be set as well??

Bug fix description

See #12 in C Code bug #96

ID 29 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title error in comment in standby_initiator_actions() **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** 455 **Page** 298 **Line** 22

Problem description

comment says suspend_target instead of standby_target

Bug fix description

amend comment

ID 135 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title Remove declaration of time_expired from untested_actions() **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 298 **Line** 43

Problem description

time_expired no longer used

Bug fix description

ID 99 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title No need to check for ATTACH_REQUEST after sending one anymore **Status** Recommend Accept
C code files port.c **Owner** **SCAT** **BRAT** **Page** 299 **Line** 32

Problem description

Can't think of any scenario which would result in a multi-port PHY sending an ATTACH_REQUEST on the test_port and then receiving an ATTACH_REQUEST on the test_port?

Seems to me that before an ATTACH_REQUEST is sent by a non single port phy, dominance is first established. And if it is, then by definition the neighboring PHY connected to the test port could _not_ have established dominance and scheduled an ATTACH_REQUEST.

So why in the code/algorithm have we put the effort into checking for either BUS_RESET or ATTACH_REQUEST to return after we sent and ATTACH_REQUEST?

Sure, it might be more robust, but I can't think of a valid topology which would let me test it.

Bug fix description

Test for ATTACH_REQUEST removed

ID 1 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title Speed filtering frequency error **Status** Recommend Accept
C code files speed.c **Owner** **SCAT** **BRAT** 465 **Page** 304 **Line** 51

Problem description

Currently waits for 10 ns

```
for (i = 0; i < (1<<DS_PHY_SPEED); i++)  
    wait_event(PH_DS_BIT_CLOCK); // Wait for 50 MHz clock
```

Bug fix description

```
for (i = 0; i < (2<<DS_PHY_SPEED); i++) // Wait for 50 MHz clock  
    wait_event(PH_DS_BIT_CLOCK);
```

ID 21 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title FIFO overflow test **Status** Recommend Accept
C code files dsport_rx.c **Owner** **SCAT** **BRAT** 467 **Page** 306 **Line** 54

Problem description

Consider the starting condition of fifo_rd_ptr = fifo_wr_ptr. This means the FIFO is empty, yet the code refuses to push onto the FIFO.

Bug fix description

The proper test should be:

```
(FIFO_DEPTH + fifo_wr_ptr - fifo_rd_ptr) % FIFO_DEPTH < FIFO_DEPTH - 1
```

The left hand side is the expression for how many elements are currently in the FIFO, and the right hand side is the maximum number of elements the FIFO can hold.

ID 9 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title encoding of BORDER **Status** Recommend Accept
C code files bport_tx.c, bport_rx.c **Owner** **SCAT** **BRAT** 469 **Page** 308 **Line** 11

Problem description

encoding and decoding of BORDER is missing

Bug fix description

Also BRAT # 476
replace 0b111 (INVALID) with BORDER

ID 131 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title variables used by rx_character not properly initialized **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** **Page** 309 **Line** 11

Problem description

rx_character uses a number of variable/signals including pkt and pkt_prefix before they are initialized.

Bug fix description

bport_rx scrubbed to see if all global signals within the file were initialized before use. Several were not and were added to rx_inactive_actions. Additionally, rx_scam_req was not being reset between each character. Consequently, update_scrambler might think a received character was a request symbol when it really was a control symbol. Probably doesn't happen in a real system ... but it felt safer to initialize the lot of rx_scam_x at the start of each character decode.

ID 51 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title rx_character **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** 470 **Page** 309 **Line** 27

Problem description

(Comment #25)
Table 16-11, pg. 309, function rx_character
Parentheses required (as written, rx_scam_ctrl is ex-or'ed with just one bit of descram).

Bug fix description

Change pg. 309 line 27 from:

```
rx_control=(rx_scam_ctrl^((descram&0x80)>>4) | ((descram&0x20)>>3) |  
((descram&0x8)>>2) | (descram&0x2)>>1);  
to:
```

```
rx_control=(rx_scam_ctrl^(((descram&0x80)>>4) | ((descram&0x20)>>3) |  
((descram&0x8)>>2) | ((descram&0x2)>>1)));
```

For aesthetic consistency with other code, change line 34 from:

```
} else if((rx_scam_req=rx_comma_decode(character_in, rx_rd))>=0 && !pkt) {  
to:  
  
} else if(((rx_scam_req=rx_comma_decode(character_in, rx_rd))>=0) && !pkt) {
```

ID 52 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title train_character_sync **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** 472 **Page** 310 **Line** 29

Problem description

(Comment #26)
Table 16-11, pg. 310, function train_character_sync
The rx_bits variable must be kept to just 16-bits of significant data.

Bug fix description

Change pg. 310 line 29 from:

rx_bits = rx_bits << 1;
to:

rx_bits = (rx_bits << 1) & 0xFFFF;

ID 53 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title rx_sync_lost_actions **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** 473 **Page** 310 **Line** 55

Problem description

(Comment #27)
Table 16-11, pg. 310, function rx_sync_lost_actions

The char_check variable must be initialized prior to calling the character_sync function.

Bug fix description

Add following at pg. 310 line 55 just after "rx_rd=negative_rd;" statement:

char_check = 0; // initialize char_check count

ID 54 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title scrambler_sync_actions **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** 474 **Page** 311 **Line** 28

Problem description

(Comment #28)
Table 16-11, pg. 311, function scrambler_sync_actions
The check for inverted polarity (by looking for either of two special characters) also requires that the rx_req_code variable be TRUE (i.e., the characters are request codes).

Bug fix description

Change pg. 311 line 28 from:

} else if((rx_req == 0xF8) || (rx_req == 0x98)) {
to:

} else if(((rx_req == 0xF8) || (rx_req == 0x98)) && rx_req_code) {

ID 121 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title Extraneous code in bport_rx.c for setting context **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** **Page** 312 **Line** 17

Problem description

The routine rx_sync_actions() in bport_rx.c contains a while loop with the comment "Do not report DATA_END symbols that have been sent for context info only"

I don't believe we are "setting context" anymore and, consequently, this while loop can be removed.

Bug fix description

while loop removed

ID 55 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title pad_count **Status** Recommend Accept
C code files bport_rx.c **Owner** **SCAT** **BRAT** 475 **Page** 313 **Line** 21

Problem description

(Comment #29)
Table 16-11, pg. 313, function bport_receive_actions
Incrementing of the pad_count variable is incorrect.

Bug fix description

Change pg. 313 line 21 from:

```
if (pad_count==rx_speed_ratio) pad_count=0;  
else pad_count++;  
to:
```

```
pad_count = (pad_count + 1) % (1 << rx_speed_ratio);
```

[JH] Used Dave Scott's suggestion to rename rx_speed_ratio to rx_speed_diff to avoid some of the confusion. And then used Skid's mod operator for the calculation.

ID 56 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title bport_transmit_actions comments **Status** Recommend Accept
C code files bport_tx.c **Owner** **SCAT** **BRAT** 477 **Page** 316 **Line** 5

Problem description

(Comment #30)
Table 16-12, pg. 316, function bport_transmit_actions
Extraneous spaces in comment.

Bug fix description

Change pg. 316 line 5 from:

```
int tx_speed_ratio; // number of symbols per byte for given pkt_speed and port_speed  
to:
```

```
int tx_speed_ratio; // number of symbols per byte for given pkt_speed and port_speed
```

ID 83 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title Bad comment in phy_response_actions() **Status** Recommend Accept
C code files decode_phy_pkt.c **Owner** **SCAT** **BRAT** **Page** 325 **Line** 4

Problem description

comment at head of routine is "overloaded to transmit LTP packet"

I don't think this is correct anymore. There is a send_LTP() routine which calls tx_quadlet directly.

Bug fix description

delete comment

ID 66 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title bestReq **Status** Recommend Accept
C code files beta_arb_functions.c **Owner** **SCAT** **BRAT** 490 **Page** 327 **Line** 10

Problem description

(Comment #40)
Table 16-18, pg. 327, function bestReq
Parentheses required to group elements correctly (> operator higher priority than & operator).

Bug fix description

Change pg. 327 lines 10-12 from:

```
*in_phase_isoch_request = (iso_cycle && (best_req.isoch & *ipm > ISOCH_NONE & *ipm));  
*in_phase_async_request = (!iso_cycle && (best_req.async & *apm >= CURRENT & *apm));  
to:
```

```
*in_phase_isoch_request = (iso_cycle && ((best_req.isoch & *ipm) > (ISOCH_NONE & *ipm)));  
*in_phase_async_request = (!iso_cycle && ((best_req.async & *apm) >= (CURRENT & *apm)));
```

ID 13 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title use of accelerating **Status** Recommend Accept
C code files process_req.c **Owner** **SCAT** **BRAT** 493 **Page** 332 **Line**

Problem description

accelerating variable is not set on legacy ISOCH_REQ

Bug fix description

```
set it  
case PH_ISOCH_REQ:  
  accelerating = TRUE;  
  breq = ISOCH_REQ;  
  req_speed = phyArbReq.speed;  
  break;
```

ID 132 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1006
Title Wrong port speed for reset_start_actions() **Status** Recommend Accept
C code files reset.c **Owner** **SCAT** **BRAT** **Page** 338 **Line** 21

Problem description

reset_start_actions() schedules a bus reset on all active ports by calling portTarb on all ports. portTarb sets the portT speed to DEFAULT for all active ports. Since reset_start_actions() must also schedule a reset on resuming and attaching ports, it sets portT "by hand" for those ports. However, it currently is using a speed of S100 rather than DEFAULT.

Bug fix description

For consistency with the active ports, the speed for resuming and attaching ports should be DEFAULT as well.

ID 70 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title Child ports in Tree_ID **Status** Recommend Accept
C code files tree_id.c **Owner** **SCAT** **BRAT** 496 **Page** 339 **Line** 11

Problem description

(Comment #10)

Table 16-21, pg. 339, function tree_ID_start_actions

Child ports are not always marked as child ports. Consider the case of a node receiving PARENT_NOTIFY on all active ports (the node should become root). As written, the code will not mark the last active port as a child, causing an unnecessary root-contention cycle. The code that checks the child count should be moved outside of the for loop. I suggest the following.

Bug fix description

Change pg. 339 lines 11-24 from:

```
do {  
...  
} while (!(reset_detected() || ibr || isbr|| arb_timer == CONFIG_TIMEOUT));  
  
to:  
do {  
  children = 0; // Count the kids afresh on each loop  
  for (i = 0; i < NPORT; i = i + 1)  
    if (!active[i] || portRarb[i] == PARENT_NOTIFY) {  
      child[i] = TRUE; // Child if disabled, disconnected, suspended,  
      children++; // or if other PHY asks us to be parent.  
    }  
  if (children == NPORT - 1 && (!force_root || arb_timer >= FORCE_ROOT_TIMEOUT))  
    return; // Only one port left as the parent.  
  else if (children == NPORT)  
    return; // We are the root.  
} while (!(reset_detected() || ibr || isbr|| arb_timer == CONFIG_TIMEOUT));
```

ID 71 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title lowest_unidentified_child **Status** Recommend Accept
C code files self_id.c **Owner** **SCAT** **BRAT** 497 **Page** 340 **Line** 13

Problem description

(Comment #07)
Table 16-22, pg. 340, function self_ID_start_actions
The lowest_unidentified_child variable is not computed correctly (its always comes out as the lowest numbered child port).

Bug fix description

Change pg. 340 lines 13-20 from:

```
for (i = 0; i < NPORT; i++) {  
    if (child_ID_complete[i]) // Tell identified children to prepare to receive data  
        portTarb(i, DATA_PREFIX);  
    else  
        portTarb(i, IDLE); // Allow parent to finish  
}  
for (i = 0; i < NPORT; i++)  
    if (child[i] && active[i]) { // If active child  
        if (all_child_ports_identified)  
            lowest_unidentified_child = i;  
        all_child_ports_identified = FALSE;  
    }  
}
```

to:

```
for (i = 0; i < NPORT; i = i + 1)  
    if (child_ID_complete[i]) // Tell identified children to prepare to receive data.  
        portTarb(i, DATA_PREFIX);  
    else {  
        portTarb(i, IDLE); // Allow parent to finish  
        if (child[i] && active[i]) { // If active child  
            if (all_child_ports_identified)  
                lowest_unidentified_child = i;  
            all_child_ports_identified = FALSE;  
        }  
    }  
}
```

Agreed ... i = i + 1 replaced with i++.

ID 130 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1006
Title grant_received not being cleared in self-id **Status** Recommend Accept
C code files seld_id.c **Owner** **SCAT** **BRAT** **Page** 340 **Line** 28

Problem description

S0:S1 relies on receiving a GRANT or SELF_ID_GRANT from the parent_port. The receipt of a grant causes process_requests.c to set grant_received. Consequently, when a non-root node completes self-id and heads off into A0, grant_received may be stale and set to TRUE, causing undesirable visits to TX (possibly after a quick visit to RX to receive more self-ID packets).

Bug fix description

clear grant_received at top of self_ID_grant_actions()

ID 81 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D1005
Title idle_bus() is incomplete ... missing ALL:R0x terms **Status** Recommend Accept
C code files idle.c **Owner** **SCAT** **BRAT** **Page** 344 **Line** 13

Problem description

idle_bus() must return false if any of the exit conditions from state A0 are true. The current idle_bus() code includes all explicit A0:xx transitions, but is missing the three implicit ones into R0.

Bug fix description

Suggestion is to add:
return (!((power_reset) or
 (reset_detected()) or
 (ibr) or
 ((proxy_root ...

ID 74 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title idle_actions parens **Status** Recommend Accept
C code files idle.c **Owner** **SCAT** **BRAT** 501 **Page** 346 **Line** 10

Problem description

(Comment #44)
Table 16-23, function idle_actions, pg. 346
Parentheses required to group elements correctly (== operator is higher priority than ? operator).

Bug fix description

Change pg. 346 lines 10-11 from:

```
if (!OK_to_grant && proxy_root &&  
    (arb_timer == STORES_GAP_COUNT ? subaction_gap: BOSS_RESTART_TIME)) {
```

to:

```
if (!OK_to_grant && proxy_root &&  
    (arb_timer == (STORES_GAP_COUNT ? subaction_gap: BOSS_RESTART_TIME))) {
```

ID 15 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title reset of did_arbrst **Status** Recommend Accept
C code files idle.c **Owner** **SCAT** **BRAT** 500 **Page** 346 **Line** 28

Problem description

need to clear this on exit even in iso_cycle, to be consistent with the way that 1394a behaves (any time that there are no requests to process after the bus has been busy causes a gap)

Bug fix description

remove the test on iso_cycle
did_arbrst = FALSE; // only relevant to a B_bus

ID 17 **Balloter** CWS **How submitted** Ballot comment **Fixed in version** D1005
Title Isoch concatenated packets **Status** Recommend Accept
C code files transmit_actions.c **Owner** **SCAT** **BRAT** 503 **Page** 348 **Line** 25

Problem description

Hold protocol is not used for concatenated iso packets from beta link

Bug fix description

```
Remove C code for this
} else if (data_to_transmit.reqType == PH_REQ_DATA_PREFIX) {
    // concatenated packets from Legacy link
    end_of_packet = link_concatenation = TRUE; // End of packet indicator
    stop_tx_packet(DATA_NULL, tx_speed, CONCATENATION_PREFIX_TIME, tx_format, NPORT);
    // MIN_PACKET_SEPARATION
    req_speed = data_to_transmit.speed;
```

ID 109 **Balloter** Biva **How submitted** Post ballot **Fixed in version** D1005
Title extra wait in receive_actions **Status** Recommend Accept
C code files receive_actions.c **Owner** **SCAT** **BRAT** **Page** 349 **Line** 52

Problem description

In Draft 1.01 P.349, L.52 the statement "wait_next_symbol(tx_speed);" should be removed, since it causes 2 PBT wait in the do-while loop per byte. Already one PBT duration is taken up by the tx_byte() function in Line 44.

Bug fix description

ID 31 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title ds_portR, rx_S200, and rxS400 variables **Status** Recommend Accept in principl
C code files phy_services.h **Owner** **SCAT** **BRAT** 421 **Page** 276 **Line** 40

Problem description

(Comment #01)
The ds_portR, rx_S200, and rxS400 variables should be arrays.

Change line 40-41 from:

```
RX_signal ds_portR;
boolean rx_S200, rx_S400;
```

to:

```
RX_signal ds_portR[NPORT];
boolean rx_S200[NPORT], rx_S400[NPORT];
```

Bug fix description

Noted that connect_detect and bias_detect were also per-port variables. #ifdef unsubscribed ... #else ... #endif region added to phy_services.h

ID 36 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title loop healing - in_test_interval **Status** Recommend Accept in principl
C code files node.c **Owner** **SCAT** **BRAT** [^] 429 **Page** 286 **Line** 28

Problem description

(Comment #12)
Table 16-6, pg. 286-287, function con_mgmt_granted
The in_test_interval variable not needed. The test_interval_timer variable should only be reset upon sending a new LTP packet. If a short-bus-reset is to initiated, there should be no packet-end sequence.
Change pg. 286 lines 28-55, pg. 287 lines 2-3 from:

```
} else { // request for loop test
...
}
to:

} else { // request for loop test
if ((link == Legacy_Link) && (breq == FAIR_REQ || breq == PRIORITY_REQ)) {
breq = NO_REQ; // Cancel the request
PH_ARB_confirmation(PH_LOST, 0, 0); // And let the link know
}
PH_DATA_indication(PH_DATA_PREFIX, 0, 0); // Send notification of bus activity

for (i = 0; i < NPORT; i = i + 1)
portTarb(i, DATA_PREFIX); // Ensure the bus is held
with DATA_PREFIX.

max_occupancy_timer = 0; // Start the occupancy timer.
release_bus = FALSE;

in_control = TRUE;
while (!release_bus)
if (need_new_LTP) {
send_LTP;
test_interval_timer = 0;
need_new_LTP = FALSE;
}
received_attach = FALSE; // Clear flag if set

start_tx_packet(S100, LEGACY); // Send null packet to clean up,
in_control = FALSE; // then release the bus.
if (!isbr_OK) // The isbr_OK flag is set in untested_actions.
stop_tx_packet(DATA_END, S100, DATA_END_TIME, LEGACY, NPORT);
}
```

Bug fix description

JH] Basically agreed. Need to add a PH_DATA_indication of DATA_END before the call to stop_tx_packet, and the received_attach flag is no longer required. release_bus became loop_test_request and is only tested, never set. Also, the port sets isbr and allows this routine to set isbr_OK. The port can't know the arb phases and understand if isbr_OK is allowed.

See #96

ID 37 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title test_port_selector - reselection of aborted port **Status** Recommend Accept in principle
C code files node.c **Owner** **SCAT** **BRAT** [^] 430 **Page** 287 **Line** 15

Problem description

(Comment #13)
Table 16-6, pg. 287, function test_port_selector
There's no mechanism to keep an aborted port from being re-selected.

There's a code race condition which could result in the while loop getting stuck. When a port completes the loop-test, its port_under_test flag is cleared by loop_test_interval_actions. At the same time, the release_bus flag is set, which causes con_mgmnt_granted to terminate and the arb-state-machine to transition to the R0 reset-start state. But this, in turn, causes the con_mgmnt_request flag to be cleared.

Now suppose that this code (test_port_selector) executed before the arb-state-machine transitioned to the R0 state. A new test_port would be selected and con_mgmnt_request would again be set. But con_mgmnt_request would be cleared upon entry into R0 (see initialize_arbitration), and there's no mechanism for it to be set again. The following while loop would be stuck. It's waiting for port_under_test to be cleared by loop_test_interval_actions, which is waiting for in_control to be set, which is set upon entry into con_mgmnt_granted, but which won't be entered because the con_mgmnt_request flag was cleared by reset_start_actions (see initialize_arbitration).

Also, considers what happens if a node has more than one untested port, and while the selected test-port is waiting for the test-interval to elapse, one of the other untested ports receives an attach-request symbol? This will cause the bus to be released and a short-bus-reset to be started, but the port_under_test flag won't be cleared, so the test_port_selector routine will be stuck in the while loop.

To fix this, add code to exit the while loop if a bus-reset is started.

Change lines pg. 287 15-25 from:

```

for (i = 0; i < NPORT; i++) { // selects ports in turn - important if testing a particular port
    // is aborted
    ...
}

```

to:

```

for (i = 0; i < NPORT && !bus_initialize_active; i = i + 1) {
// Selects ports in turn - important if testing a particular port is aborted.
// Select unique port to test if none already selected (do not need to search from port 0),
// but don't select a port that is currently receiving a dominant LTS value.
if (untested[i] &&
    !((test_interval_timer >= TEST_INTERVAL) && !need_new_LTP &&
    ((portRarb[i] == DATA_BYTE) &&
    ((current_data[i] & 'h3F) > HR_test_value) || ((current_data[i] & 'h80) != 0))) {
    received_LTS = 0; // once in 64, this will cause a phantom potential collision
    port_under_test[i] = TRUE;
    test_port = i;
    con_mgmnt_request = TRUE; // and request the bus
    while (port_under_test[i] && !bus_initialize_active)
        ; // and sit here while the port is tested
    test_port = NPORT;
    port_under_test[i] = FALSE;
    con_mgmnt_request = FALSE; // don't need to request the bus now
}
}

```

Bug fix description

[JH] Agreed in principle, but implementation has radically changed. The test_port_selector routine has been merged in with the loop_test_interval_actions() routine and the asynchronous handshakes revised. Consequently, I believe the race conditions have been corrected. Specifically, loop_test_interval_actions() selects a valid test port based on some refined criteria to make sure we don't select a port which will have to be aborted. port_under_test is then set true which allows the given test_port to know it has been addressed by any send_attach or loop_disable commands. The port, once commanded to attach, is responsible for scheduling the attach in a fashion that is synchronized with the rising edge of bus_initialize_active. Only after the port has successfully synchronized with the reset can loop_test_interval_actions continue and select a new test port. The request to gain control (loop_test_request) is now a level signal, not a set/clear semaphore. So as long as the test port needs service, con_mgmt_granted will eventually be visited. No more deadlock. And in all cases, testing of a given port is guaranteed to conclude with clearing port_under_test.

[JH] The desire to keep an aborted port from being reselected was addressed by Wooten's DevCon presentation. So this is accepted in principle. As written, the proposed fix relies on current_data[i] holding the last received LTS. This inspiration allowed us to always use current_data[i] for the comparison rather than relying on received_LTS to be set (the source of other race conditions). Consequently, received_LTS is removed.

See #96

ID 38 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title loop_test_interval_actions **Status** Recommend Accept in principle
C code files node.c **Owner** **SCAT** **BRAT** [^] 431 **Page** 287 **Line** 44

Problem description

(Comment #14)
Table 16-6, pg. 287, function loop_test_interval_actions
Loop-healing logic needs to wait for the test_interval_timer to increment to at least TEST_INTERVAL time before comparing LTS values. If an attach-request is received on any port, then processing in this function should be skipped (the processing in untested_actions should control the release of the bus, etc.).

I can't see any need for the in_test_interval flag--all references to this flag may be deleted.

I suggest the following to replace the loop_test_interval_actions function in its entirety.

```
void loop_test_interval_actions() { // continuously running
    static int collision_count;

    if (power_reset) {
        collision_count = 0;
        in_control = FALSE;
        HR_test_value = new_test_value();
        while (power_reset)
            ;
        return;
    }

    if ((test_port != NPORT) && port_under_test[test_port]) {
        // Delay testing until the test port selector has initialized a new port to test.

        if (received_attach) // Attach-request symbol rcv'd on some untested port (not necessarily the test_port).
            ; // Do nothing

        // Check if the test-port is receiving a dominant LTS from its peer port.
        else if ((test_interval_timer >= TEST_INTERVAL && !need_new_LTP &&
            (received_LTS & 'h3F) > HR_test_value) || (received_LTS & 'h80) != 0) {
            // We've waited long enough for the last xmitted LTP to reach all
            // nodes in the network, and the received LTS is dominant, so abort
            // testing on this port (go to next).
            port_under_test[test_port] = FALSE;
            release_bus = TRUE;
            while (in_control)
                ;
            return;
        }

        // Check if there's a collision between the xmitted LTS and the rcv'd LTS.
        else if (in_control && test_interval_timer >= TEST_INTERVAL && !need_new_LTP &&
            (received_LTS & 'h7F) == ((HR_G << 6) | HR_test_value)) {
            // There's a collision between xmitted LTS and received LTS.
            if ((USING_EUI) && (collision_count == 0))
                EUI_sequence = 0; // reset the EUI sequence
            collision_count = collision_count + 1;
            if (collision_count == COLLISION_LIMIT) {
                loop_disabled[test_port] = TRUE; // place port in loop disabled state
                port_under_test[test_port] = FALSE;
                release_bus = TRUE;
                while (in_control)
                    ;
                collision_count = 0;
                return;
            }
            HR_test_value = new_test_value(0);
            HR_G = (HR_G == 0) ? 1 : 0;
            need_new_LTP = TRUE;
            return;
        }
    }
}
```

```

}

// Check if the test-port can be attached.
else if (in_control && test_interval_timer >= TEST_INTERVAL && !need_new_LTP) {
    // Test interval has expired, and we're in control, without a collision
    collision_count = 0; // Clear any bogus collisions
    HR_mode = ATTACH_IN_PROGRESS; // Should not receive attach-request on any port
    // once this is set.
    need_new_LTP = TRUE;
    while (need_new_LTP)
        ;
    send_attach = TRUE; // Flag for the test port
    while (send_attach) { // Wait for the port to complete
        if (!(untested[test_port] || (!in_control)) {
            // If the port has shut down for some reason or on bus reset...
            send_attach = FALSE;
            port_under_test[test_port] = FALSE;
            release_bus = TRUE;
            while (in_control)
                ;
            return;
        }
    }
    release_bus = TRUE;
    while (in_control)
        ;
    port_under_test[test_port] = FALSE;
    return;
}
} else {
    collision_count = 0;
    send_attach = FALSE;
}
}

```

Bug fix description

[JH]Disagree that we need to wait so long to compare LTS to HR. Generation bit was included so that we could quickly detect collisions without having to wait for full intervals. However, can't abort too quickly or assume no collision too quickly. Basically, a new_ltp guarantees that the generation number has changed and that the old HR <> the new HR. So if we get LTS=HR, we note a collision immediately. But as long as LTS <> HR, we have to wait a full test interval before drawing a conclusion. This change on aborting is captured in the DevCon presentation and included in the latest implementation.

[JH] Agreed on processing outstanding attaches before selecting a port to test. No need to test if attach is going to complete and force a new test interval anyway.

[JH] Agreed on in_test_interval. It has been removed.

See #96

ID 41 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title toner - code imporvement **Status** Recommend Accept in principl
C code files port.c **Owner** **SCAT** **BRAT** [^] 445 **Page** 291 **Line** 42

Problem description

(Comment #16)
Table 16-7, pg. 291, function toner
A suggestion for code simplification which (IMHO) better illustrates the intended purpose.

Change lines 42-54 to:

```
while (toning || send_speed) {
    t_ack = we_agree && send_speed;
    t_speed = send_speed ? (PHY.port_speed[pn] + 1) : 0;
    for (i = 0; i < TONE_INTERVAL; i = i + 1) {
        if (i == 0 ||                // start bit
            i == 1 && t_ack ||        // ack bit
            i >= 2 && i <= 4 && (t_speed & (1 << (i - 2)) != 0)) // speed bits
            send_tone;
        else
            wait_time(TONE_DURATION);

        wait_time(SPEEDCODE_BIT_INTERVAL); // inter-bit gap

        if (i == 4) {
            if (t_speed != 0) // speed code bits sent
                signal(EVT_SENT_SPEED);
            if (t_ack) // ack bit sent
                signal(EVT_SENT_ACK);
        }
    }
}
```

Bug fix description

[JH] Agreed in principle, but the specific code above doesn't meet the intent of waiting until the last possible point to sample the latest we_agree and port_speed signals. So the spirit of the change was accepted, but the actual code is slightly different. Parens also added to send_tone().

ID 105 **Balloter** Mamezaki-San **How submitted** Post ballot **Fixed in version** D103
Title LTS speed **Status** Recommend Accept in principl
C code files port.c **Owner** **SCAT** **BRAT** [^] **Page** 298 **Line** 55

Problem description

In untested_actions() function, LTS symbols are transmitted at port speed, but receiver will receive it at S100 speed because of no speed code.

Bug fix description

Fix checked in as PRN #366. bport_rx.c was also updated such that when an LTS is being received, only changed data bytes are pushed into the FIFO. This solves the elasticity problem as well ... An LTS sequence looks like a packet >> max packet size, so the FIFO could easily overflow. Doesn't really have any significant consequence if the FIFO is changed to always accept the most recent push as has been suggested, but it feels cleaner to specifically document that LTS's are long and should be treated, for FIFO pushing reasons, as deletable symbols.

ID 44 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title untested_actions - various **Status** Recommend Accept in principle
C code files port.c **Owner** **SCAT** **BRAT** [^] 458 **Page** 299 **Line** 3

Problem description

(Comment #18)

Table 16-7, pg. 299-300, function untested_actions

1) Not enough time given for current S100 symbol (already in portT) to be transmitted; the data-prefix symbols may not be transmitted.

At pg. 299 line ~3, insert following just after "portT.arg = ARB_STATE;" statement:

```
wait_next_symbol(S100);
```

2) The con_mgmnt_request signal should be set only when node is not already in control of bus.

Change pg. 299 line 46 from:

```
con_mgmnt_request = TRUE; // request the bus  
to:
```

```
if (!lin_control)  
con_mgmnt_request = TRUE; // request the bus
```

3) The while-loop (waiting for node to gain control of bus) should exit when sync is lost.

Change pg. 299 line ~48 from:

```
while (lin_control && !seen_reset && !bport_sync_ok) {  
to:
```

```
while (lin_control && !seen_reset && bport_sync_ok) {
```

4) Need to clear port_under_test flag is attach-request received.

At pg. 299 line ~43, insert following just after "untested = FALSE;" statement:

```
port_under_test = FALSE;
```

Need to make sure that port_under_test flag is cleared if we lose sync.

At pg. 300 line ~13, insert following just before return statement:

```
port_under_test = FALSE;
```

5) If port becomes loop-disabled, need to ensure that flags are cleared and bus released.

At pg. 300 line ~14, insert following just before "} // end of loop while untested":

```
if (loop_disabled) {  
untested = FALSE;  
port_under_test = FALSE;  
release_bus = TRUE;  
return;  
}
```

6) The untested_fault flag is redundant. Replace all occurrences of untested_fault with loop_disabled. Affects the P11:P12 transition in Figure 11-2, pg. 176, as well.

Bug fix description

1) [JH] Point is taken, but preferred approach is to not schedule S100 arb symbols in start_port() or start_resume_port(). Instead, starting arb indication will be given a speed of DEFAULT, marking the starting arb states as purely elastic.

2) [JH] This fix is no longer required. The port does not directly request the bus. Instead, attach is used to indicate when the node controller should cause a bus reset to happen.

3) [JH] Current implementation is different but agrees in spirit. If the port ever loses sync, untested_actions must conclude.

4) [JH] Current implementation uses a combination of attach and untested to communicate to the test controller when the port is "done". Basically, once a command is sent to the port, the controller waits for either attach to be true or untested to be false to conclude testing. A new test port can only be selected if any pending attach is also completed. So the test_port is completely done if attach and untested are both false.

5) [JH] Agreed in principle, but current implementation has a different structure. The node controller instructs the port that a loop has been found ... the port responds by clearing untested and sets loop_disabled. On seeing !untested, the node controller clears the port_under_test, etc. and releases the bus.

6)[JH] Not entirely true. Intent was that loop_disabled indicates if we are in P12 because of a loop we found. Untested_fault was intended to indicate if start_port() failed or we loss sync. This would happen, for example, if the far end was in the disable

state or could happen if the far end went to P12 first (initiated the loop disable). On a bus reset, we would only force ports out of P12 which were there because of a loop.

[JH] I was concerned with the asynchronous nature of clearing loop_disabled by reset.c. Would it be possible for port to set loop_disabled but, before the state machine transitions were evaluated, a reset cleared it? If so I would end up in P2 rather than P12! I decided not to worry about it. The found_loop signal from the node controller causes the port to set loop_disabled. found_loop in turn is only asserted when we have control of the bus, and we don't release control of the bus until the port acknowledges found_loop. At that point, there is a call to start_tx_packet() before we can get into R0. This should be plenty of time for the state machine to move to P12.

See #96

ID 110	Balloter Biva	How submitted Post ballot	Fixed in version D1005
Title	missing isbr_OK in untested_actions	Status	Recommend Accept in principle
C code files	port.c	Owner	SCAT <input type="checkbox"/> BRAT [^] Page 299 Line 27

Problem description

P.185, L.11-13 (Sec11.7.11) mentions "If,however,an LTP with a mode of ATTACH_IN_PROGRESS had been sent on the active bus,then a bus reset must be sent to the active bus (note:this bus reset is not sent on the port that had lost synchronization since it is no longer the test port.)". In order to ensure this we probably require that untested_actions() have a line "isbr_OK = TRUE;" inserted just after the if (!bport_sync_OK) condition near Line 27, Page 299. Should it be a short or a long bus reset ?

Bug fix description

This issue was addressed by the loop-breaking revamp. Basically, if we lose sync after sending an ATTACH_IN_PROGRESS, we either issue another LTP with mode of LTP_TEST in the case of a non-isolated node, or we simply set HR_MODE to LTP_TEST in the case of an isolated_node. Comments to this effect are in the latest node.c.

ID 46 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D103
Title Ds arb state decode **Status** Recommend Accept in principl
C code files dsport_rx.c **Owner** **SCAT** **BRAT** 461 **Page** 301 **Line** 16

Problem description

(Comment #20)
Table 16-18, pg. 301, function decode
Decoding of DS arb states is dependent upon the arb control state-machine state.
Change pg. 301 lines ~16-28 to:

```
// Note: During tree-ID, it is possible to receive short periods of
// RX_DATA_PREFIX, e.g., while in T1: Child_Handshake waiting for the
// peer port to transition to the S0:Self_ID_Start state.
switch(rx_arb) {
    case RX_DATA_PREFIX: return(DATA_PREFIX);

    case RX_DATA_END: return(PHY.PHY_state < S0) ? PARENT_HANDSHAKE : DATA_END;
    // RX_PARENT_HANDSHAKE is same as RX_DATA_END

    case RX_PARENT_NOTIFY: return(PHY.PHY_state < A0) ? PARENT_NOTIFY : REQUEST_CANCEL;
    // RX_REQUEST_CANCEL is same as RX_PARENT_NOTIFY

    case RX_SELF_ID_GRANT: return(PHY.PHY_state < A0) ? SELF_ID_GRANT : LEGACY_REQUEST;
    // RX_REQUEST is same as RX_SELF_ID_GRANT

    case RX_ROOT_CONTENTION: return(PHY.PHY_state < A0) ? ROOT_CONTENTION : GRANT;
    // RX_GRANT is same as RX_ROOT_CONTENTION

    case RX_IDENT_DONE: return(PHY.PHY_state < A0) ? IDENT_DONE : DISABLE_NOTIFY;
    // RX_DISABLE_NOTIFY is same as RX_IDENT_DONE

    case RX_IDLE: return(IDLE);

    case RX_BUS_RESET: return(BUS_RESET);

    case RX_CHILD_HANDSHAKE: return(CHILD_HANDSHAKE);

    case RX_SUSPEND: return(SUSPEND);
}
```

Bug fix description

See #23 (BRAT #415) Agreed in principle ... some of the above overloads are incorrect and RX_SUSPEND isn't isolated. Still, process_req has been modified to use PHY_state to filter and resolve the overloading. And a caution about filtering of spurious RX_DATA_PREFIX indications has been added to dsport_rx.c

ID 50 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title fifo error **Status** Recommend Accept in principl
C code files bport_rx.c **Owner** **SCAT** **BRAT** 468 **Page** 306 **Line** 54

Problem description

(Comment #24)
Table 16-11, pg. 306, function push
The if-statement condition is incorrect (it will not evaluate to TRUE if the fifo_rd_ptr is equal to the fifo_wr_ptr, i.e., the FIFO is empty).
Change pg. 306 line ~54 from:

```
if (((fifo_rd_ptr-fifo_wr_ptr + FIFO_DEPTH) % FIFO_DEPTH) > 1) {
```

Bug fix description

See 21 for equivalent fix

ID 100 **Balloter** CWS **How submitted** Post ballot **Fixed in version** D103
Title Stretching bug in bport_transmit_actions **Status** Recommend Accept in principle
C code files bport_tx.c **Owner** **SCAT** **BRAT** **Page** 316 **Line** 26

Problem description

I believe there is a bug in stretching request symbols. Unless portT.speed is assigned the value DEFAULT whenever a packet is not being transferred, the request symbols will be stretched according to the present value of tx_speed_ratio. While a request symbol is being stretched, portT can change at a higher packet speed rate and the state machine will miss the first of these new symbols.

I recommend the following change to the code at the end of bport_transmit_actions:

FROM:

```
tx_character(localT);  
for(i=1; i<tx_speed_ratio; i++)  
    tx_character(localT);            // stretch as required
```

TO:

```
tx_character(localT);  
if localT.tag == CTRL {  
    for(i=1; i<tx_speed_ratio; i++)  
        tx_character(localT);            // stretch control symbols as required  
}
```

Bug fix description

Preferred fix was to make sure portT.speed had been appropriately initialized before assigning to portT.tag. In many places, it turned out the assignment to portT.speed was redundant, but it was retained/added for clarity to emphasize that the speed field must be defined for each transmit symbol.

ID 112 **Balloter** Biva **How submitted** Post ballot **Fixed in version** D1006
Title self_ID and proxy **Status** Recommend Accept in principl
C code files seld_id.c **Owner** **SCAT** **BRAT** [^] **Page** 340 **Line** 18

Problem description

P.340, L.18-20 (self_ID_start_actions()) should include child_ID_complete[i] and proxy[i] flags too. Hence the condition should look like

```
if (!child_ID_complete[i]) {
    if (child[i] && (active[i] || proxy[i]))
        // or ((child && active) || proxy) ??
        lowest_unidentified_child = i;
    all_child_ports_identified = FALSE;
}
```

Bug fix description

[JH] Fix above doesn't appear to be complete. Skid's #7 (C code bug #71) dealt with what is documented above, but I expect that Biva is pointing out an additional bug related to proxy ports. Awaiting the complete fix before implementing ... Biva's little questioning comment is interesting (and, as it turns out, benign).

At this point, not all combinations of Child/Active/Proxy are possible. In particular, if Active is FALSE, then Child is TRUE. Equally, Active and Proxy cannot both be TRUE.

So the possible combinations are:-

Child	Active	Proxy	Set lowest_unidentified_child?
F	T	F	N
T	F	F	N
T	F	T	Y
T	T	F	Y

So Biva's fix is the "obvious" one, and his alternative is IMHO, less obviously correct (although it still happens to work)
 Bug fixed (format slightly different due to also fixing a bug from Skid)

ID 72 **Balloter** Jim Skidmore **How submitted** Ballot comment **Fixed in version** D1005
Title senior_border starting assumption **Status** Recommend Reject
C code files self_id.c **Owner** **SCAT** **BRAT** 498 **Page** 342 **Line** 11

Problem description

(Comment #08)
 Table 16-22, pg. 342, function self_ID_transmit_actions
 If node is a border node, the starting assumption is that it's the senior_border and that there's NO senior_port.
 Change pg. 342 line 11 from:

```
senior_port = parent_port; // working assumption, may be changed later
```

to:

```
senior_port = NPORT; // working assumption, may be changed later
```

Bug fix description

Further issue shown up by Biva - see C code bug #113 for complete solution

Tree ID establishes parent_port and root, and we presume senior_port and proxy_root settings are identical. Self ID establishes a true proxy_root and, if different from root, senior_ports are adjusted to always point to the proxy_root. Once in normal arbitration, root and parent_port are completely ignored. All arbitration is based on senior_port and proxy_root.
