To: IEEE P1394a

From: Steve Finch
Silicon Systems Inc.
14351 Myford Rd.
Tustin, CA 92780-7068
(714) 573-6808
Fx: (714) 573-6916
email: steve.finch@tus.ssi1.com

Revision: 0.1

Date: Feb 3, 1997

---

Dear members,

I voluteered to help Peter Johansson with the 1394a document by pulling together the input for clause “9. Clarifications and corrigenda”. To that end, I requested input via the reflectors, and captured any responsed that I received. I then “organized” them into common subjects (as much as that was possible) and have put them into this document.

What I think we should do next is to review this material to see:

1) if the issue needs to be included in P1394a
2) if the issue should be included in clause 9 or whether some other clause is a better spot
3) if we have an answer/fix is it the correct answer/fix
4) if we have just a question/problem, how do we go about generating an answer/fix

Also, I am willing to continue to collect things. So if you know of an issue and you don’t see it on the list, get it to me and I’ll add it to my collection and bring it to the groups attention.

steve

## “Issues” Table of Contents

# 1. Isolation

## 1.1 Required or optional

We have this problem with the way Annex A is currently worded with respect to PHY-LINK isolation. Some people have incorrectly interpreted it to say that PHY-LINK isolation is required (normative). We need to modify this so that it is clear that PHY-LINK isolation is **NEVER** ‘required'.

A simple solution to this problem might be to just make Annex A informative like is was for a long time anyway.

## 1.2 50 uA requirement still correct?

Isolation: The isolation requirement is that the sum of the currents through all wires shall be less than 50 uA. Is that stable, or has there been proposals for changing this?

<<editor's notes: >>
On Isolation:
1. 1394:
a. I could find no statement of either requirement or option in this regard. Since Annex A is normative and no statement of it being an option, it would appear to be a requirement.
b. Annex A contains isolation at both the link/phy interface and phy/phy interface.

2. 1394A
a. Contains statements which indicate that link/phy may not be required, but doen't state so explicitly.

i. Clause 5, contains:

The clock rate of the signals at this interface remains constant, independent of speed, to support galvanic isolation for implementations where it is desirable.

and

Table 5-1 — Signal description contains

| Name | Driven by | Description |
|---|---|---|
| Direct | Neither | Indicates direct connection or isolation barrier |

and

The Direct signal is used to disable the digital differentiator on the D, Ctl, SCLk, LPS, and LReq signals, indicating that the two chips are directly connected, rather than through an isolation barrier.

ii. Clause 5.3, contains:

The differentiator is disabled by the "Direct" signal when the chips are directly connected.

b. I could find nothing in 1394A about phy/phy isolation.

<<end notes>>

## 2. Shield connections

### 2.1 Meeting rules and FCC compliance not possible

The specification makes normative shield connections that are not possilbe in many systems. It is not possible to follow some of the normative rules that are in Annex A and have the system pass FCC. For example, the insistance on DC isolating the shield from logic ground simply does not work in most of our systems. The _only_ way we are able to get through FCC is by keepiong chasis and logic grounds tied _firmly_ together (i.e. they are the same thing.)

### 2.2 Signal screens connected to outter shield

The main one I can remember is that the diagram of the cable showed the pair
screens being connected to the outer shield, which was definitely wrong.

## 3. Cable Power

### 3.1 Voltage supplied: differences between TA and standard

Power Voltages

(Referencing: David Wooten's Power Distribution Slides.)

a) If there a formal (near standard) document version of these? If so, where is the pdf draft downloadable from?

b) On slide #9, it states that the new voltage limits are 20V min and 33V max. Are these for the producer (it should supply this) or the consumer (it should be capable of using this).

### 3.2 Consistancy between producer and consumer

Previous 1394-1995 specs were: Producer: 8-40V; Consumer: 7.5-40V (4.2.2.7), or perhaps 8-40V (A.3.2, para 3)

Is the producer spec tighter (such as 24V-to-33V), or is the consumer spec lighter (such as 15V-to-33V)?

Producer and consumer specs clearly have to be different, to account for modest IR drops.

Why was the upper range dropped from 40V (1394-1995) to the current 1394TA-proposed value of 33V. Rumors are that this makes the consumer's DC-to-DC converters cheaper. Is this true? What are the specific part numbers and breakdown voltages -- this is useful information is one tries to encourage related systems to use the same voltage-distribution standard.

3.3 Current Ripple

Ref: 4.2.2.7

The "Maximum peak-to-peak ripple" is stated to be 100 mA *(Iload/1.5A). Is this the same as Iload/15? If not, what is the relevance of the 100mA and 1.5A numbers?

3.4 Measurement of max ripple and max change in load

Ref: 4.2.2.7

The maximum change in current load is spec'd to be less than Iload in 100us. I'm a bit confused about how ripple and max change are measured. Is it thus OK to have a 100% ripple, linearly connecting: 0A at t=0; Iload at 100us; 0A at t=200us; (...). Seems like there is some intent to have the max-change and ripple spec's be applied over certain frequency ranges. A set of test conditions or test-jig illustrations would certainly help, as is often done for IC-specification tests. Or, am I simply missing something obvious?

I'm interested in what are the final (or near-final) results of the 1394TA work, if the 1394TA assumptions have effectively revised implementaters operational assumptions.

3.5 Power management when insufficient power available to operate

Self-powered repeater concerns

Suppose self-powered repeater is on, and you turn on a remote device that consumes 10W. The repeater power is then lost, making its phy non-operational. Then, this repeater is logically disconnected (you can't reach the control registers on the other side), but the remote (and now not accessible) device is consuming 10W.

Question: How is this problem avoided, or if unavoidable what is the recommended recovery strategy when that 10W of power on the not accessible device is needed for another accessible device?

4. Necessity for separate packet queues for requests, responses, and isochronous data

By the way, I was dismayed to learn yesterday, in a private communication, that many who are working on 1394 products don't understand that it is absolutely necessary to have separate packet queues for requests, responses, and isochronous data!!

I'd suggest that the Trade Association start an education campaign ASAP to remedy this misunderstanding, or Serial Bus products will have deadlocks as soon as it starts to be used seriously, and of a kind that can't be fixed without redesign.

That kind of product recall situation could bring an undeserved bad reputation to the whole technology. Apparently this problem was caused by the relevant parts of the spec being "informative" instead of "normative", which is a pretty strange situation for something so fundamental to proper operation. This should be fixed as part of the P1394A corrections, of course, but that will be finished too late to avoid disaster. Geez, people are going to think we've learned nothing, putting our vendors through the equivalent of the PCI-Bridge deadlock nightmares all over again. Arrgh!)

# 5. Clearification Requests

## 5.1 Dribble bits

<<editor's note: This message was a response to my request for "items to be fixed" and got some side bar discussion. I tried to capture it and put together here. I don't know if it is anything but confusing.>>

Original message:

When a PHY is receiving a packet (repeating) does it pass the "dribble bits" on to the LINK? My intuition says no. Careful reading of the C (C++?) code in tables 4-47 and 4-41 says it does. However, the same careful reading reveals syntax errors which make me dubious that this code has ever passed muster with a compiler, let alone actually been run. Am I wrong about that? Other sections refer to the LINK checking for certain packet sizes. REF clause 6.3.3.2 page 168, State L4; Transition L4:L0b; Transition L4:L5; Do these references include/exclude dribble bits?

Related message:

> Your assumption about the dribble bits is correct -- they are NOT sent to the link. The PHY creates them and strips them off prior to sending the data to the link. The dribble bits are used to precondition the serial bus at the end of the data into either DATA_END or DATA_PREFIX (for concatenated packets). DATA_END is (from transmitter's point of view) DATA = 0, STRB = 1.

Thanks so much for your reply. I would not be surprised at all to find that most knowledgable people out there, and anyone who has implemented a PHY would agree with you. However, can you refer me to anything in the spec, or even related documentation, that states such?

The spec states to the contrary as far as I can tell: Table 4-41 code (section 4.4.1.2) includes the decode_bit() function which shows any transition - data bits, dribble bits, or end symbols - going into the FIFO. It also includes the rx_bit(*,*) function which shows how the end of "data" is located, so that the dribble bits will be read out of the FIFO. Table 4-47 (section 4.4.2.4.2) shows code including the receive_actions() function. This function clearly shows rx_bit() being used to grab the dribble bits from the FIFO and send them to the LINK via PH_DATA.ind( ).

I believe you are correct, and therefore the spec is wrong here, but how do we substantiate such a position? Anyone else want to jump in and help?

Perhaps you were trying to keep things simple for me, but part of your explanation still leaves a slight hole in my understanding.

> If we are doing a non-concatenated packet (ie. ending with DATA_END) AND the last Data bit of the packet was DATA = 1, the dribble bits provide the means for arriving at the required DATA_END value.

No problem with that. However, given a DATA_PREFIX at packet start, and an even number of data bits, only 2 possible line states are possible at the end of packet data: TX_DATA_PREFIX and TX_DATA_END. In either case, we can established the desired ending line state with only 2 additional bits. A "00" sequence will get the line state to TX_DATA_END. A "11" sequence will get the line state to TX_DATA_PREFIX. The last of these bits is clearly identified as the "end symbol". I think it is arguable that the 2 bit sequence is actually the end symbol, but if it isn't then what we have is 1 dribble bit followed by an end symbol. So in ANY case a single dribble bit is sufficient to accomplish the purpose you rightly identified. The spec requires a speed specific number of multiple dribble bits. Why? What is their purpose?

I am afraid that part of your explanation went right by me.

>We cannot add any transitions to the DATA_END because each DATA/STROBE transition indicates a bit of data, hence the need for "dribble bits".

Each dribble bit and the end symbol itself are transitions, are they not? So aren't we already adding transitions, and indicating "a bit of data" for stuff beyond the payload data bits? So I'm not sure what you are saying here.

Two explanations (of my confusion with the spec regarding dribble) seem plausible to me:
1) An assumption has been made about the data path width and number of pipeline stages between the transceiver/port and the resync buffer. The speed dependent number of dribble bits is to ensure a clock edge on the extracted receive clock to get the last packet data through that pipeline and into the resync buffer. Therefore, the dribble bits and end symbol never get to the resync FIFO.
2) No such pipe exists. The bits do get to the FIFO. Packet end detection does not occur until they are read from the FIFO and sent to the ANNEX-J interface. The control sequencing is such that the last ANNEX-J "word" does not get sent. Note that the number of dribble bits plus end symbol is exactly 1 ANNEX-J "word".

Both explanations represent reasonable designs. Neither explanation is supported by the spec (IMHO).
1) All description of the port interface - drawings, C code, text - indicate it is serial and un-pipelined. Little text description of the resync buffer is given. The C code for it also precludes deserialization or registering.
2) While Annex-J seems to be a very popular suggestion, thats what the spec calls it. A "suggested implementation", "informative", "example". It is not binding, therefore compliant operation cannot depend upon it.

## 5.2 Resync Buffer

More details regarding the Resync Buffer would be useful. Those I have so far spoken with know less in this regard than I, and are equally confused. I understand its clock regime crossing and clock speed matching functions. I even understand how to calculate 1.6 clocks (S200), 3.2 clocks (S200), and 6.4 clocks (S400) of speed matching need. I am suspicious that this behavior relates to "dribble bits" since if you round the numbers up you have very nearly the dribble + ending bit length. But why should that be so? Wouldn't the Buffer go empty without dribble bits, and everything get received and repeated? The satement about flushing the last bit through the receiver (clause 4.4.1.1) is curious. It seems to suggest something about the structure of the receiver decode and sync logic, but is unclear about what.

## 5.3 Transmit LINK under-run

It is not clear what is acceptable and necessary behavior when a transmitting LINK has an under-run. If local FIFOs are less than a full packet, and streaming mode operation is being used, it is possible that mid packet the LINK could run out of available data. It seems suggested that the LINK-PHY interface - particularly ANNEX-J - cannot tolerate a stall. Even if it could, we would probably trigger an early buffer empty in the receiving nodes. So some sort of packet termination is called for. It is probably important to ensure an even bit length. But what more beyond that? Must it be quadlet multiple length, as normal? Make it the actual length, but force some other (CRC?) error? What confirmation code to use? This case is not well defined in the spec and I think it is an important one.

## 6. Psuedo code errors

### 6.1 Table 4-47

1. I think code in table 4-47 is in error regarding PHY packet detection? In the "while (~test_end )" loop, there is an if clause commented "// accumulate first 64 bits". This clause causes bit_count to only be incremented when it is < 64. Therefore any packet >= 64 bits in length will have bit_count == 64 upon exit of the loop. This condition is the test condition later used to determine "// PHY packet received". I believe simply making the bit_count++ a seperate, and unconditional statement, after the if clause will solve the problem. If I'm wrong about this, then I miss-understand something, and am in need of enlightenment.

2. While we're on code, the same table 4-47 code, just above the " if (bit_count == 64) // PHY packet received" test has a call "stop_rx_packet(ending_data};". Note the typo - ending "}". Also, unless I missed a second prototype, the call uses an undeclared signature. Table 4-43 is the definition of stop_rx_packet. It is declared returning void, taking no arguments. So what is the above call supposed to do?

3. Same table 4-47, very top line on page 112, is an assignment involving 2 undeclared varaibles "phy_addr = received_bits". Also missing ";".

4. Back to table 4-47. Right near the end in a switch statement there are calls to stop_tx_packet(argument). Table 4-40 declares a stop_tx_packet taking 2 arguments. Unless I missed a prototype.....

### 6.2 Table 4-41

Table 4-41, definition of rx_bit. Assignments are made to "end_of_data" and "rx_data". These are declared as pointers, and the assignments are values. Assignments should be to "*end_of_data" and "*rx_bit".

7. When to sample speed signal

When speed signaling, when is the common mode voltage considered stable? (It seems to me that from the standard alone one could latch in the speed signal anywhere between -0.02us and 0.10us. I would think that somewhere around 0.04us the common mode would be stable and a valid compare could be had.) Should the standard spec the compare time?

8. Jitter and skew budget: need review?

Per table 4-24 Cable environment jitter and skew, and annex E.2; is it believed that the jitter and skew budgets are allocated correctly and are not overly conservative, or is there room in these numbers for adjustment and reallocation now that real silicon and cables and connectors are available?

9. Overshoot restrictions too restrictive?

In section 4.2.2.1 ,Signal amplitude, it is mentioned that an additional 10% of overshoot is allowed beyond the differential output signal amplitude specified in table 4-12. At 400Mb/s and even 200Mb/s is it realistic to allow only 26.5 mV with rise and fall times of 1.2 and 2.2 ns respectfully?

10. Link Request state machine figure

ref: figure J-7

What happens in the case of an Imm or Iso request when the LOST or WON states have been reached? There is no description of how the state machine should return to the idle state from these states for an iso or imm request, but this is surely required, at least in the next cycle (i.e when the cycle sync event has occured) so that new iso or imm requests can be sent ? Also, within a (125 us) cycle it is surely necessary that the idle state be reached after an iso or imm request so that a possible subsequent imm request (or iso or fair | pri for that matter) can be sent ?

Proposal :

Instead of the condition 'Reqsent' for the transition R2:R0 (LOST->IDLE), we could have : 'Reqsent || cyc_sync && (iso || imm)',

(It would also be necessary to have the whole request logic, like in the IDLE state, duplicated in the LOST state, so that a request can be sent, and therefore a reqsent can be achieved in the LOST state after an imm | iso lost request) and similarly for the transition R4:R0 (WON->IDLE), instead of 'ctl==Idle && (fair || pri) we could have 'ctl==Idle && (fair || pri) || cyc_sync && (iso || imm)'

11. Editorial corrections

11.1 Figure 8-20, field size incorrectly indicated

Figure 8-20 shows the node_vendor_id as 16 bits. This should be corrected to indicate that the field is 24 bits in length.

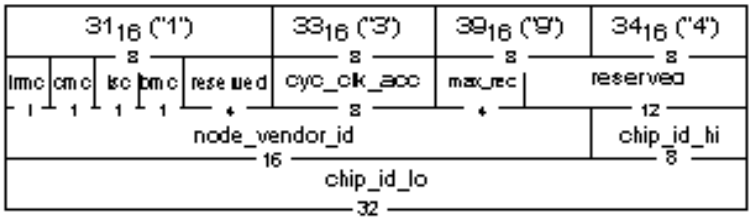


Figure 8-20 — Bus_Info_Block format

**Proposed text for inclusion in 1394A:**

Near clause 8.3.2.5.4, an error in figure 8-20 indicates the size of the node_vendor_id field is 16 bits. The size of the node_vendor_id field is 24 bits.

11.2 Note in clause 8.4.6.2 and Table 8-7 disagree

On page 237, in the note at the end of section 8.4.6.2, it says that the gap count can be set to 33 to allow for a worst case (16 hop) bus. The correct value is 42, according to Table 8-7 just above the note. I have also heard the value 42 from implementors.

**Proposed text for inclusion in 1394A:**

In clause 8.4.6.2, the note at the end of the clause conflicts with the values contained in Table 8-7. The table contains the correct value. Within the note the sentence which reads:

A gap count of 33 is sufficient for the worst-case 16 hops permitted by Serial Bus.

should read:

A gap count of 42 is sufficient for the worst-case 16 hops permitted by Serial Bus.

### 8.4.6.2 Gap count optimization

Serial Bus optimization by the bus manager is optional. Serial Bus performance may be optimized in three principal ways:

a) reconfigure the cable topology in order to reduce the number of hops, or
b) reconfigure the cable topology in order to arrange nodes of the same speed capability adjacent to one another, or
c) optimize the gap count for the current cable topology.

The first two methods, while desirable, are beyond the scope of this specification. The third method is the only one avail-able to the bus manager to optimize Serial Bus performance for a given cable topology. Although this optimization is optional, it is strongly recommended because Serial Bus performance is seriously degraded when the gap count is left at its default value.

Table 8-7 gives the values for gap_count for each maximum number of cable hops for a particular topology. The formulas used to calculate this table and their derivations are given in annex E.1.

| Max_hops | Total delay | Gap_count | Subaction_gap (µs) | Arb_delay (µs) | Total (µs) |
|---|---|---|---|---|---|
| 1 | 0.3295 | 1 | 0.6002 | 0.0814 | 0.6816 |
| 2 | 0.6589 | 4 | 0.9257 | 0.1628 | 1.0885 |
| 3 | 0.9884 | 6 | 1.2512 | 0.2441 | 1.4954 |
| 4 | 1.3178 | 9 | 1.5767 | 0.3255 | 1.9023 |
| 5 | 1.6473 | 12 | 1.9023 | 0.4069 | 2.3092 |
| 6 | 1.9767 | 14 | 2.2278 | 0.4883 | 2.7161 |
| 7 | 2.3062 | 17 | 2.5533 | 0.5697 | 3.1230 |
| 8 | 2.6356 | 20 | 2.8788 | 0.6510 | 3.5299 |
| 9 | 2.9651 | 23 | 3.2043 | 0.7324 | 3.9368 |
| 10 | 3.2945 | 25 | 3.5299 | 0.8138 | 4.3437 |
| 11 | 3.6240 | 28 | 3.8554 | 0.8952 | 4.7506 |
| 12 | 3.9534 | 31 | 4.1809 | 0.9766 | 5.1575 |
| 13 | 4.2829 | 33 | 4.5064 | 1.0579 | 5.5644 |
| 14 | 4.6123 | 36 | 4.8319 | 1.1393 | 5.9713 |
| 15 | 4.9418 | 39 | 5.3202 | 1.2614 | 6.5816 |
| 16 | 5.2712 | 42 | 5.6458 | 1.3428 | 6.9855 |

Table 8-7 — Calculated gap counts

If the bus manager performs gap count optimization, it shall first calculate an upper bound for the maximum number of hops of the current cable topology. This is done by examining the self-ID packets stored in the topology map in order to determine the total number of nodes and the number of leaf nodes. Once the value of Max_hops is calculated, the bus manager shall broadcast a PHY configuration packet with the t bit set to one and the gap_cnt field set to no less than the value obtained from table 8-7. After the bus manager has broadcast the PHY configuration packet, it may initiate a bus reset in order to confirm that all Serial Bus nodes are configured to operate with the new gap count.

NOTE: A bus manager or, in the absence of a bus manager, an isochronous resource manager, may effect significant Serial Bus performance improvements without the need to analyze bus topology and calculate the maximum number of hops. A gap count of 33 is sufficient for the worst-case 16 hops permitted by Serial Bus. This value is a substantial optimization of the default gap count of 63 in effect after a power reset.