FROM:	Jerry Hauck, jerry_hauck@ccm.sc.intel.com
TO:	P1394a Working Group
DATE:	October 8, 1997
RE:	Isochronous period too long

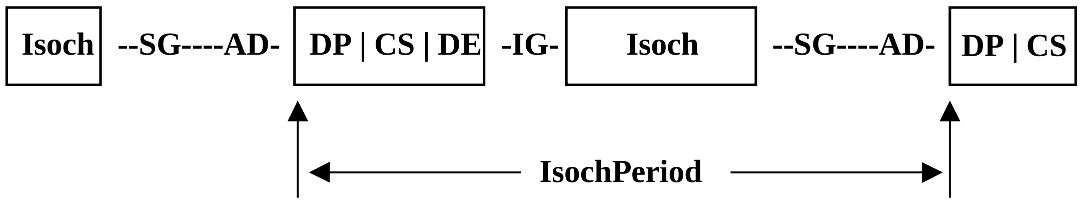
Subclause 9.20 of the P1394a Draft 1.1 states:

> If the cycle master detects an isochronous period that exceeds the maximum time, it shall clear its own *cmstr* bit and cease transmission of cycle start packets. The cycle too long condition is defined as the failure to detect a subaction gap within 125 µs of the transmission of a cycle start packet.

This subclause seems to imply that the "maximum time" allowed for isochronous traffic (including cycle starts) is 125 µs and that adherence to that maximum time would allow isochronous transmissions to continue indefinitely and on time. I propose that the 125 µs limit as defined is not sufficient to allow indefinite, on-time isochronous transmission. Furthermore, I believe there is a legitimate motivation to ensure that non-cycle start asynchronous packets can make forward progress even in the presence of maximum-sized isochronous periods.

Analysis

Consider the following limiting model of bus behavior:



Legend:

SG	= Subaction Gap
AD	= Arb Delay
DP	= Data Prefix
DE	= Data End
IG	= Isoch Gap
CS	= Cycle Start
Isoch	= Abstract collection of all Isoch stuff including DP/DE, etc.

Note: For this example, I've defined the IsochPeriod to include the Arb delay which follows a subaction gap.

Consider three possible durations for the IsochPeriod described above:

i. **IsochPeriod > 125 µs:** This is a trivial case ... after a finite number of back to back isoch intervals, there will be at least one interval during which two cycleSync's occur within the same period. Consequently, cycles are lost because the bandwidth is over subscribed. No asynch packets (other than cycle start) are ever sent.

ii. **IsochPeriod = 125 µs:** This is the boundary case. Whatever phasing the cycleSync interval has with the cycle start packet will remain constant over all cycle start packets. There is an exact one-to-one ratio of cycleSync's to IsochPeriod's. This case can repeat indefinitely without loss of isochronous data or cycles, but no asynchronous packets other than cycle start can ever be sent.

iii. **IsochPeriod < 125 µs:** This case guarantees that asynchronous packets can be sent with *some* frequency. This does **not** guarantee that an asynch packet can be sent each isochronous period, however. Since the cycleSync period = 125 µs and the IsochPeriod < 125 µs, the phasing of cycleSync's relative to the end of the isoch period will "walk". Said differently in each successive isoch interval, the next cycle sync will occur later in time relative to the end of the last isoch packet. After a finite number of back-to-back isoch periods, there will be at least one phase alignment in which the PriReq to the PHY resulting from the cycleSync occurs after the subaction_gap + arb_delay interval. When this occurs, the PHY will have to wait until an arb_reset_gap + arb_delay interval elapses before granting the bus for a cycle start. While waiting for this arb_reset_gap, the PHY is guaranteed to recognize and grant any arbitration request from any node which was triggered by the subaction gap. Voila ... an asynch packet gets a chance. The number of back-to-back isoch cycles which occur before an asynchronous packet sees an opportunity becomes a function of how close the IsochPeriod is to 125 µs.

It is my understanding that Draft 1.1 of P1394a intended to allow case ii above; i.e., isochronous transmission could continue forever without any guarantee for asynchronous delivery. To allow case ii:

IsochPeriod = 125 µs
DP+CS+DE+IG+Isoch+SG+AD = 125 µs

The description in 9.20 implies that a timer (which I'll call cycleTooLongTimer) starts counting with the start of the cycle start packet and ends with the subaction gap indication. Therefore, the proper trigger point to allow case ii would be given from the above as:

cycleTooLongTimer + AD > 125 µs

So, it appears that the "proper" time-out should be 125 µs – AD, not the 125 µs limit noted in 9.20. With a gap count of 63, AD ~ 2.6 µs, giving a time-out of ~122.4.

However, my analysis was based on bus timings on the wire. We need to consider PHY and LINK delays. For example, the subaction gap indication from the PHY takes a finite propagation time as does the GRANT indication for the start of the cycle Start packet. The delays are offsetting and, if they are equal, the analysis is unchanged. (To be precise, we'd also have to look at the variations in the elapsed time between detecting cycleSync and getting that indication latched in the PHY. I'm assuming the jitter on this is quite low and in the noise.)

Modified Proposal

After some consideration, I think case iii is a compelling target for P1394a. Consider the case when two independent buses are joined and the isochronous bandwidth is instantaneously overallocated. Normally, this situation corrects itself by requiring each isoch talker to reaffirm channel and bandwidth allocations. Unfortunately, such reaffirmations are asynchronous packets and might never get sent when the isoch cycle length falls in case i or ii. Triggering "isochronous period too long" for both case i and ii ensures that a bus with oversubscribed isochronous bandwidth can *gracefully* recover.

Adopting case iii as the only valid operating regime, the ideal P1394a trigger point would be given as:

> cycleTooLongTimer >= 125 µs – AD, or
> cycleTooLongTimer >= 122.4 µs (with a gap count of 63)

Of course, each implementation would have to adjust their time-out depending on when they actually start and stop the period timer. If there are differing pipeline delays for the timer's start and stop events, adjustment of the trigger would be required.

Since the specification of a presise time-out isn't called for in this application, I suggest that a reasonable detection range be allowed for by the spec in order to ease implementation. This also allows us to be a bit sloppy with regards to the issue of PHY/LINK notification delays, precise start/stop timing definitions, etc. Consequently, I propose the following replacement text for subclause 9.20:

> If the cycle master detects an isochronous period that exceeds the allotted time, it shall clear its own *cmstr* bit and cease transmission of cycle start packets. The cycle too long condition shall be detected whenever the duration of the current isochronous period, if maintained in subsequent periods, would starve transmission of asynchronous packets other than cycle starts.
>
> To meet this requirment, implementations with a discrete PHY and a dedicated period timer should trigger no less than 115 µs and no more than 120 µs after sending a cycle start unless a subaction gap or bus reset indication is first observed.

Implementations without a dedicated period timer should observe the cycle_offset fields of consecutive cycle start packets and trigger when, in the absence of intervening asynchronous traffic, a subsequent cycle_offset either remains constant or increases relative to a previous cycle_offset. Bus events which constitute asynchronous traffic in such an implementation include:

a) bus reset indication,
b) arbitration reset gap indication, or
c) reception or transmission of an asynchronous packet other than cycle start.