802.1AS Slave Clock Interface Proposal

2 July 07
Chuck Harrison
cfharr@erols.com
Notice of copyright release

• Notice:
  – This document has been prepared to assist the work of the IEEE 802 Working Group. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.

• Copyright Release to IEEE:
  – The contributor grants a free, irrevocable license to the IEEE to incorporate material contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE’s name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE’s sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by the IEEE 802 Working Group.
Slave Clock behaviors

• “Capture” timing behaviors
  – Event timestamp
  – Cross timestamp to another timescale

• “Generate” timing behaviors
  – Clock gen (e.g. 1PPS, 44.1kHz, 24.576MHz)
  – Single trigger out at specified time

• “Status” behavior
  – Warn client of timescale discontinuity
Clock timing behavior abstract logic

• **Fundamental capabilities:** application independent
  – Event capture
  – Trigger generation
  – Both require only two very simple primitives:
    • Event (in or out): zero parameters
    • Global time (out or in): one parameter

• **Derived capabilities:** more application specific, perhaps more directly useful
  – Cross timestamp
  – Clock gen
  – Both require more complex primitives
Proposal: Five Easy Pieces

• Define 5 interfaces in 802.1AS for slave clocks:
  – Event Capture
  – Trigger Generate
  – Cross Timestamp
  – Clock Generate
  – Discontinuity

• Cross Timestamp and Clock Generate are defined as state machines relying on the Event Capture and Trigger Generate interfaces, respectively

• All five interfaces are Optional in PICS
  – If implemented, each has mandatory & optional prims.
Fundamental Interfaces

EVENT_CAP.request {  // mandatory
    // No parameters
}
EVENT_CAP.response {  // mandatory
    grandTime // Time when request received
}

TRIG_GEN.request {  // mandatory
    grandTime // Time when trig to be generated
}
TRIG_GEN.response {  // mandatory
    // No parameters
}
Clock Generator Interface

CLK_RATE.request {  // mandatory
    clockRate // cycles per second (0 = never)
}

CLK_PHASE.request {  // optional
    grandTime // Time when a CLK_GEN event will occur
}

CLK_GEN.indication {  // mandatory
    grandTime // Time of this event (optionally NULL)
}

Behavior of this interface is defined by an adaptation layer state machine which generates periodic TRIG_GEN.request primitives and passes each TRIG_GEN.response primitive through to client as a CLK_GEN.indication.
Cross Timestamp Interface

XTS_EVENT.request {  // optional
    // No parameters }

XTS_JAM.request {  // optional
    newCount    // Value to jam into event counter
}

XTS_POLL.request {  // mandatory
    // No parameters }

XTS_POLL.response {  // mandatory
    grandTime,   // Time corresponding to eventNumber
    eventNumber  //
}

Behavior of this interface is defined by an adaptation layer state machine which passes each XTS_EVENT.request primitive to the underlying layer as an EVENT_CAP.request while also counting the requests.
Cross Timestamp Interface II

• If `XTS_EVENT.request` is driven by a media clock, `eventNumber : grandTime` is the cross-timestamp required for many synchronization algorithms (e.g. RTP).
  - `XTS_EVENT` and `XTS_JAM` are optional, as the interface remains very useful even when the underlying media clock is maintained by another application interface.

• The “underlying media clock” may also be the `stationTime` of dvj presos, or 61883 SYT clock

• If `XTS_EVENT.request` is driven by individual arbitrary events, this interface provides the integrity check offered in earlier dvj and ch proposals by the `frameCount` field.
Discontinuity Interface

```java
TIME_DISC.indication { // mandatory
    synchronized // Enum: true, false, or unspecified
}
```

This primitive is generated whenever
- an event (e.g. change of GrandMaster ID) occurs which constitutes a potential timescale discontinuity, or
- there is a change in the value of the `synchronized` parameter

This interface provides both “event” and a “status” services

Question:
- is the “true” value of `synchronized` meaningful?
Optional/Mandatory recap

• All five interfaces are optional
  – Example: a device may expose time only as programmatic availability of a \textit{stationTime : grandTime} cross-stamp.
  – Example: a device may expose time only as the availability of a 1 kHz squarewave.
  – Standardizing the fundamental interfaces (Event Capture and Trigger Generate) is essential for defining the behavior of the derived interfaces even if the fundamental interfaces are not exposed.

• Within each interface specification there are primitives which are mandatory \textit{if} claiming PICS compliance with that interface spec.

• All five interfaces are abstract.

• Why define interfaces if they are all optional & abstract?
  – Reduce the probability of “stupid” implementations by newbies = increase the chance of successful early deployment of AVB.
Event Capture service interface

Time client

EVENT_CAP.request

EVENT_CAP.response

Time synchronization service

EVENT_CAP.request

EVENT_CAP.response

7654680.238402 sec
Task Status: Slave Clock Interface

- **Event Capture interface**
  - Well understood, has consensus, editorial only
- **Cross Timestamp interface**
  - New, needs socialization
  - Needs adaptation State Machine and text
- **Trigger Generation interface**
  - New, needs socialization
  - Needs State Machine and text
- **Clock Generation interface**
  - Consensus in principle, verify details of primitives
  - Needs adaptation State Machine and text
- **Discontinuity interface**
  - “event” vs “status” semantics need review
  - Needs State Machine and text