802.1AS Slave Clock Interface Proposal (revised)

10 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 is defined as state machine relying on the Event Capture interface

- 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.indication { // mandatory
    // No parameters
}
Event Capture service interface

- Event Capture service interface
- Time client
- Time synchronization service

EVENT_CAP.request → EVENT_CAP.response

7654680.238402 sec
TRIG_GEN state machine

Start

T = 0.;

Wait

(TGRptr=dequeue(TGRQ)) != NULL
T = TGRptr->grandTime;

(now() == T) && T != 0.
issue(TRIG_GEN.indication);
T = 0.;

Note: the conditional for this transition is TRUE only instantaneously. This is considered acceptable for an abstract Mealy state machine.
Clock Generator Interface

CLK_RATE.request {  // mandatory
    clockRate,  // cycles per second (0 = never)
    grandTimePhase  // CLK_GEN phase specification
}

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

Behavior of this interface:

A CLK_GEN.indication is generated for every time $t$ at which

$$(t - \text{grandTimePhase}) = n \times \frac{1}{\text{clockRate}}$$

for some integer $n$. 
CLK_GEN state machine

Start

Rate = 0.;

Wait

(CRRptr=dequeue(CRRQ)) != NULL
Rate = CRRptr->clockRate;
Tphase = CRRptr->grandTimePhase;

(frac( (now()-Tphase)*Rate ) == 0.) && (Rate != 0.)
issue(CLK_GEN.indication)

Note: the conditional for this transition is TRUE only instantaneously. This is considered acceptable for an abstract Mealy state machine.
Cross Timestamp Interface

```java
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, // Timestamp of eventNumber’th
        // XTS_EVENT.request
    eventNumber // value of event counter at time of
        // XTS_POLL.request
}
```

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.
Example: Media clock cross-stamp

Media clock = XTS_EVENT.request

Global clock

Event counter

dQCK

Clock latch

dQCK

Data returned by XTS_POLL.response

Invoke XTS_POLL.request = Media timescale

Event counter = Media timescale

7654680.238402 sec

183 184 185 186 187 188

186

Invoke XTS_POLL.request
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 so-called “underlying media clock” may be a precision low-jitter (i.e. PLL filtered) time-of-day clock.
- 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.
TIME_DISC.indication {
    disruption // boolean
}

This primitive is generated whenever there is a change in the value of the disruption parameter. The disruption parameter is set to TRUE if:

- an event (e.g. change of GrandMaster ID) occurs which constitutes a potential timescale discontinuity, or
- the 802.1AS layer detects a nonuniformity in the progression of time greater than a <TBD> threshold (e.g. the currently active GrandMaster is manually set or newly acquires lock to an external reference)

The disruption parameter is set to FALSE otherwise.
Optional/Mandatory recap

- All five interfaces are optional
  - Example: a device may expose time only as programmatic availability of a `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 useful 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 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.
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 behavioral text
- Clock Generation interface
  - Consensus in principle, verify details of primitives
  - Needs behavioral text
- Discontinuity interface
  - Consensus in principle, verify details of semantics
  - Nonuniformity detection needs review
  - Needs State Machine and text