

A proved model for the synchronization of a ClockSlave

Rodrigo Ferreira Coelho, Siemens AG

Dragan Obradovic, Siemens AG

Günter Steindl, Siemens AG

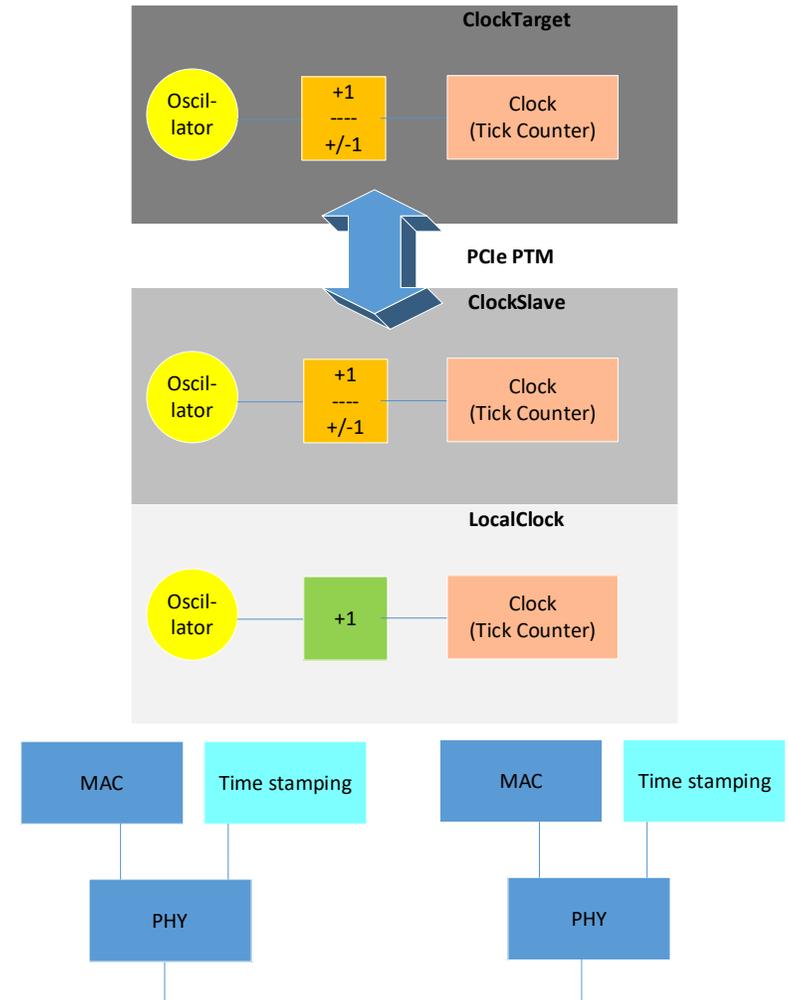
V02

Controlling the ClockSlave Log

V01: Initial version

V02: Sync states and coupling control

Implementation model assumed for this presentation



The statement of this presentation are in general independent from this implementation model.

The error sources and thus the error contribution may differ depending on the chosen implementation.

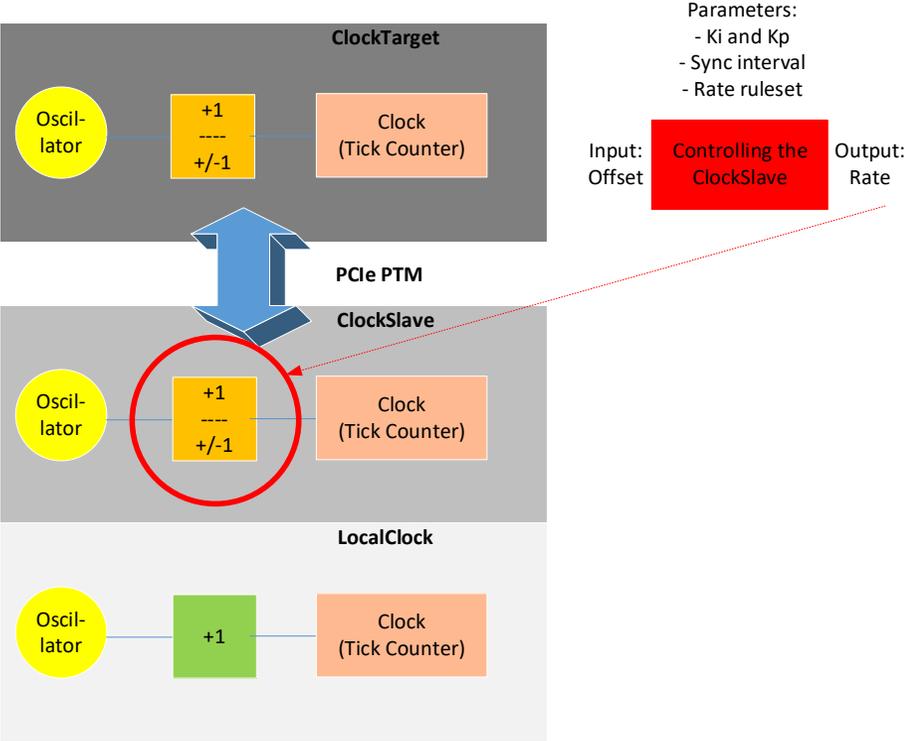
Example 1:
LocalClock, ClockSlave and ClockTarget may share an oscillator.

Example 2:
ClockSlave and ClockTarget may share an oscillator.

Additionally, the model is independent whether a PTP relay is implemented or not.

Controlling the ClockSlave

Controlling the ClockSlave Control loop



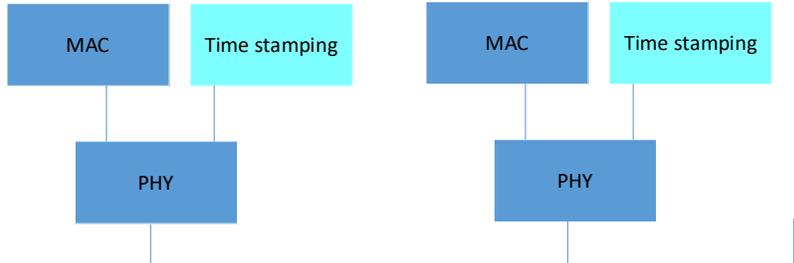
Each calculated Offset value (time interval e.g. in nanoseconds) is used as an input for a PI controller which calculates the “new” rate controlling the ClockSlave time.

The PI controller ensures that the rules defined for the changes of the value of the ClockSlave time are kept.

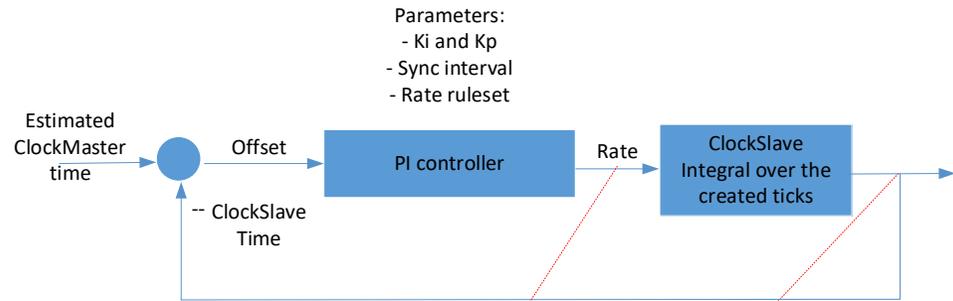
Example:
To simplify hardware implementation, this rate is expressed as the interval (number of ticks) after which a tick is added (count faster) or omitted (count slower).

If the ClockSlave was too “slow”, then for each interval an additional “tick” will be added to ClockSlave time.

If the ClockSlave was to “fast”, then for each interval the “tick” will be skipped and thus the ClockSlave time is not incremented.



Controlling the ClockSlave Control loop



Offset

Estimated ClockMaster time – ClockSlave time when receiving a sync message

- The PI controller is only executed when a sync message is received. Thus, the configured Rate stays constant until the next sync message is received.

Rate

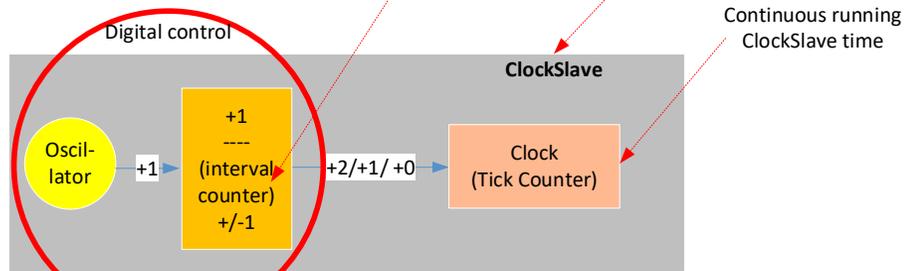
Offset correction factor calculated as an interval in ticks in which an additional +1 is added or a +1 is skipped.

PI controller

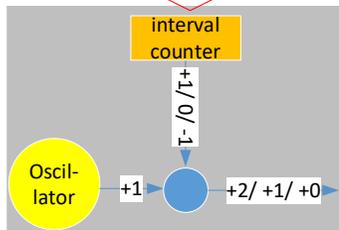
The PI controller applies

- Ki and Kp
- and
- uses the configured sync interval and the defined ruleset for allowed rate values.

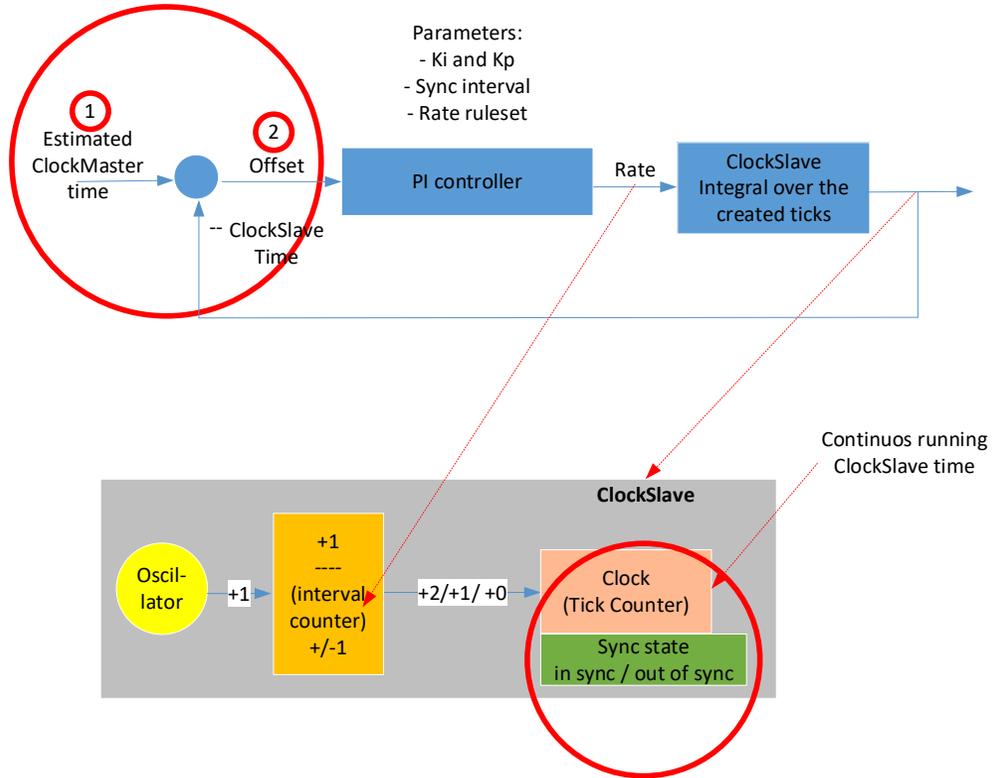
Digital control for the ClockSlave time counter is implemented as accelerated (+2 instead of +1) or slowed down (+0 instead of +1) counting.



Implementation can be expressed as this model



Controlling the ClockSlave Sync state



Sync state

The sync states (at least “in sync” and “out of sync”) of a ClockSlave is dependent on different inputs.

1. Receiving sync messages in time

Missing three consecutive sync messages change the sync state to “out of sync”

2. Offset checking

a) Out of sync

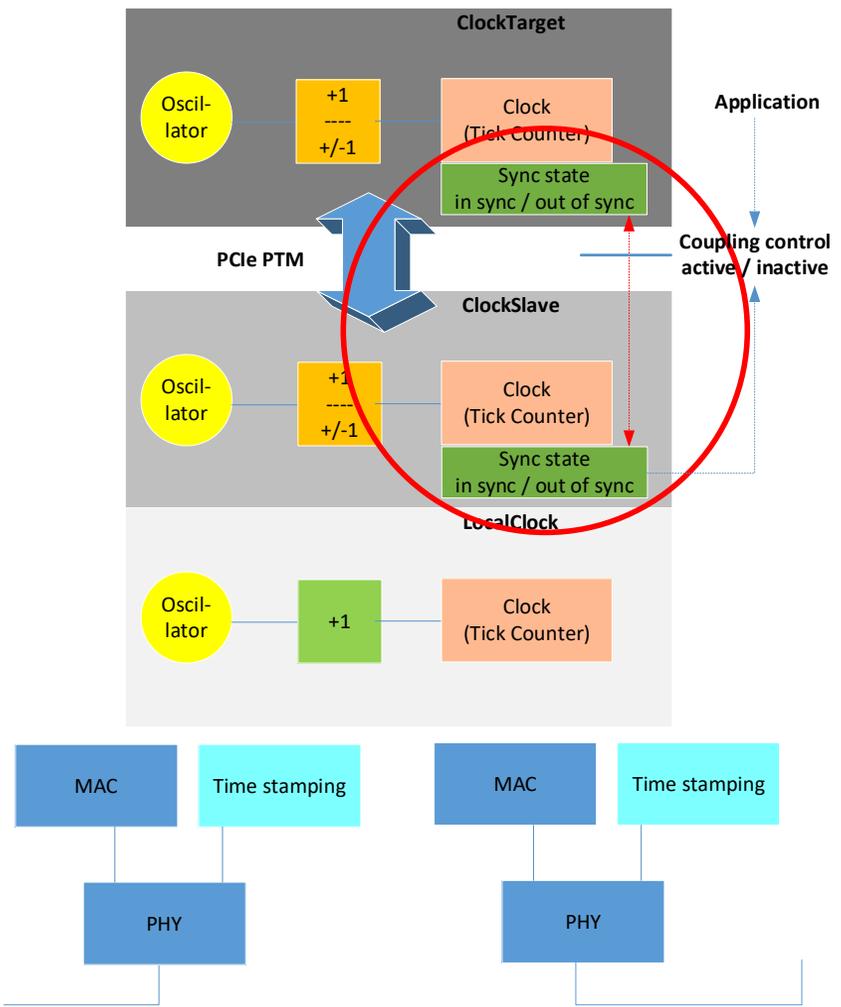
Getting three consecutive sync messages while the offset is greater than $1\mu\text{s}$ (in case of WorkingClock) changes the sync state to “out of sync”

b) In sync

Getting three consecutive sync messages while the offset is less than $1\mu\text{s}$ (in case of WorkingClock) changes the sync state to “in sync”

Controlling the ClockSlave

Sync state – ClockSlave vs. ClockTarget



ClockSlave and ClockTarget do have independent sync states (at least “in sync” and “out of sync”).

ClockTarget sync state

The ClockTarget sync state is “in sync” if the ClockSlave sync state is “in sync” and the coupling is active. The ClockTarget sync state is “out of sync” if coupling is inactive.

Coupling control

It is the responsibility of the coupling control to ensure a continuous running ClockTarget time.

The ClockTarget is decoupled from the ClockSlave as soon as the ClockSlave enters into the “out of sync” state.

An application can activate coupling if the ClockSlave sync state is “in sync” and a jump of the ClockTarget time is acceptable.

Controlling the ClockSlave

Conclusion

This control model doesn't calculate (and thus not use) a rate ratio between the ClockMaster and the ClockSlave oscillator for the control of the ClockSlave time.

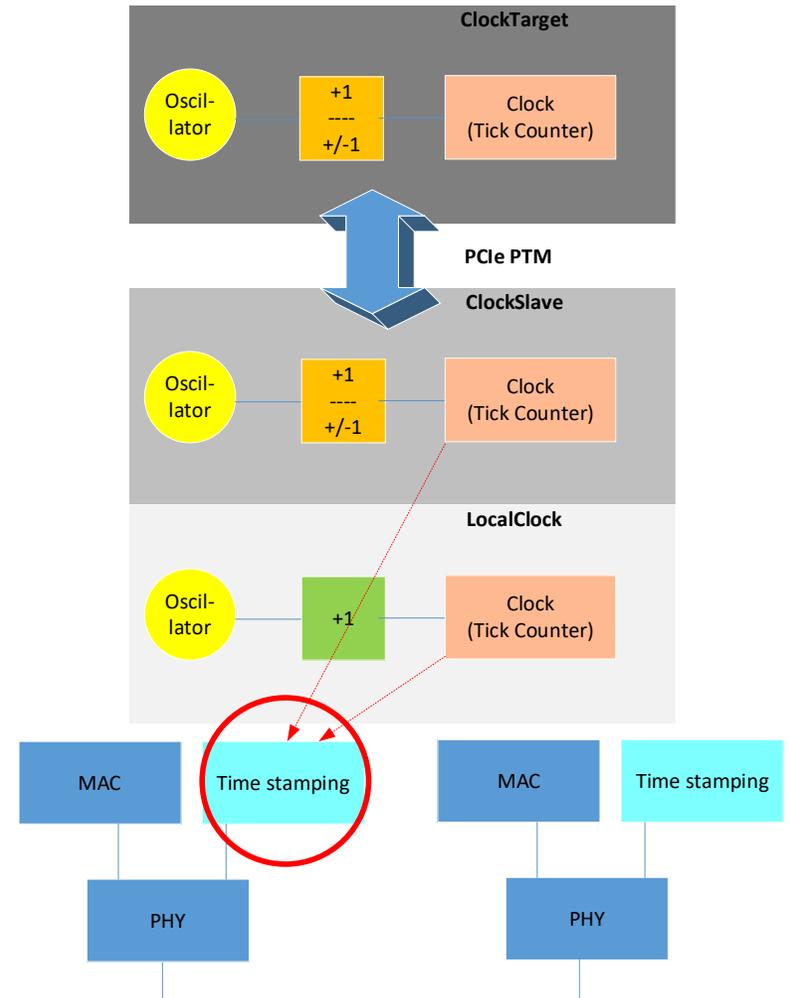
It just calculates the offset between the estimated ClockMaster time and the ClockSlave time and tries to minimize this offset to zero.

A threshold for the offset is defined to decide whether a ClockSlave is in the "in sync" or "out of sync" range.

Rate ratio calculation

Rate ratio calculation

Possible ways



The rate ratio between the ClockMaster and the ClockSlave can be calculated in multiple ways.

Example 1:
Neighbor rate ratio as specified in 802.1AS

Example 2:
Using the sync message receive timestamp similar to the concept explained at **slide 5**.

With each sync message, both, the receive timestamp and the estimated ClockMaster time are stored.

A sliding window of at least 200ms is used to calculate the time interval for the stored estimated ClockMaster times and associated the LocalClock times. The ratio between these intervals is used as rate ratio.

Example 3:
“Using the ClockSlave time while synchronized”
Use of synchronized ClockSlave time (implicit applied rate ratio) as hardware optimization in conjunction with residence times $\ll 1$ ms for the implementation of one step.

Rate ratio calculation

Filtering

The calculated rate ratio values are filtered in multiple ways:

1. Is the rate ratio value in an allowed range (e.g. values above 250ppm are assumed erroneous if +/-50ppm oscillators are used)
2. A median filter – median out of the last seven values – is applied

Rate ratio calculation

Conclusion

ClockSlave control

This model doesn't calculate (and thus not use) a rate ratio between the ClockMaster and the ClockSlave oscillator for the control of the ClockSlave time.

It just calculates the estimated offset between the estimated ClockMaster time and the ClockSlave time and tries to minimize this offset to zero.

Rate calculation (for pdelay and residence time compensation)

Rate ratio calculation based on the definition on **slide 12**.

It does calculate the rate ratio concurrently for pdelay and residence time compensation based on the stored timestamps / time intervals.

-> This model works if **no** pdelay messages are used due to exact measured pdelay value in case of aerospace or in-car too.

Constraints

The offset between ClockMaster and a ClockSlave
and
the rate ratio between them can be calculated and controlled in
multiple ways.
Not all of them may fit for industrial automation!

Conclusion

A solution is not only about synchronization accuracy.
It needs to fit to all of the industrial automation requirements.
Sync states, hot standby states and state machines AND
implementation simplicity need to fit, too.