Updating IOC Time Capabilities

Timo Korhonen & Babak Kalantari

PSI, Villigen, Switzerland

1. drvTS revisited
2. Wishlist
3. Current status
4. Proposal

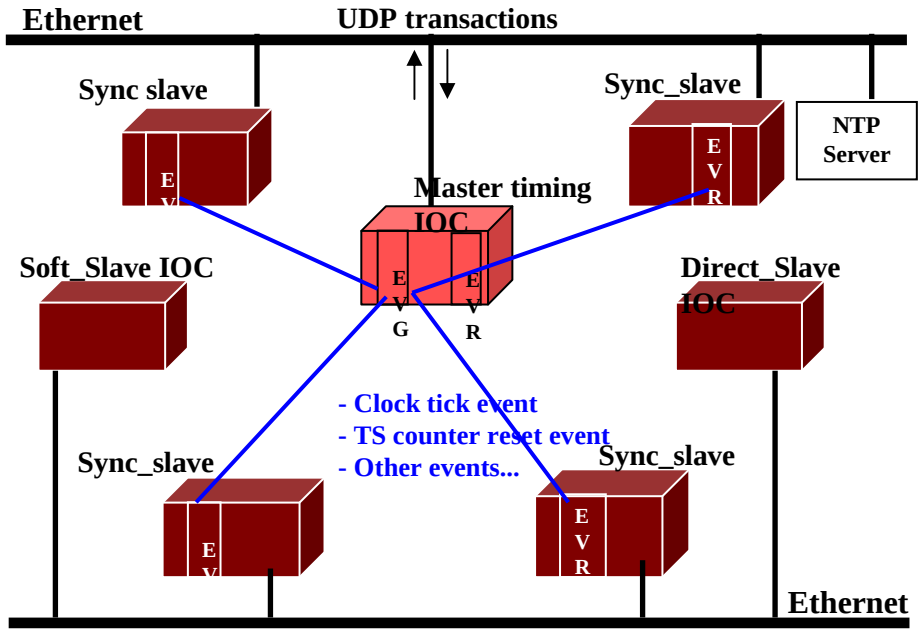
What is drvTS?

The drvTS was/is the default timestamping mechanism in EPICS prior to 3.14.
The main concepts:
•'soft' and 'hard' timestamping are supported in a **single monolithic package**.
•Hardware timestamping implementation was written to support APS event system. The same concept of hardware time support is continued in SLS and Diamond (Micro-Research) event systems; these are thus also supported.
•The (custom) soft time protocol in drvTS is a simple slewing algorithm that tries to keep the local clock in sync with a master timing IOC (or NTP)

We have complained a lot about drvTS but actually it has some great advantages:

•possibility to have perfectly synchronized timestamps across IOCs when hardware support (event system) is present
•possibility to timestamp data by the occurrence of an event (trigger)
    •this is supported in the records

Event System Synchronized timestamps



A system setup with event timing:

- Event and timestamp distribution over dedicated link
- Ethernet/UDP used for startup, transmitting validation data and soft timing control
- A mixture of ‘hard’, ‘soft’ and ‘direct’ timing (drvTS terminology)

Each event receiver has a timestamp counter that can count

- Timestamp events or
- Received event frames (after prescaling)

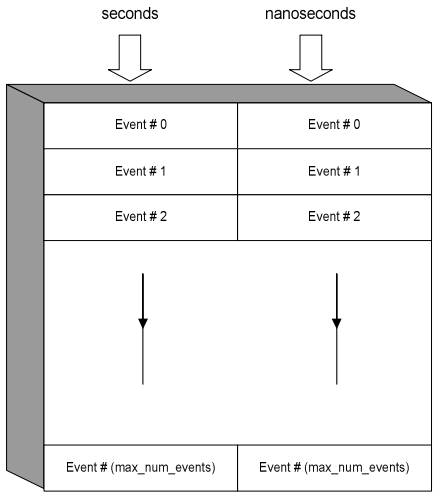
- The counters can be synchronously reset
- The timestamp counter is latched and put into a FIFO when an event is received

With this support it is possible to build a perfectly synchronized timestamp system (and it works!)

Event System Timestamp Support and drvTS

drvTS was built around the event system facilities:

- There is one master providing the 'real' time
  - The real time to slaves is provided at IOC boot time
  - For hardware timestamps the system in principle does not need the network afterwards. However, the IOCs receive time broadcasts to determine if they are still in sync.
- The IOC time is updated from the timestamp counter with the IOC clock rate.
- A table of the last occurrence of each event number (1-255) is kept
  - The latched counter value in FIFO is read and converted into real time (sec and ns after EPICS 'epoch' of 1.1.1990)
- Records can request the event time by placing the event number in the TSE field, or a link in TSEL (both in dbCommon)
  - There are a couple of special values for TSE:
    - 0 is the 'current IOC time' (default)
    - -1is the 'best available' time; usually would be reading from the timestamp counter
    - -2 means 'get timestamp from device support'



Advantage of synchronous timestamps

Follow this line of thought: the control system is our ‘100000-channel oscilloscope‘ to view the system.

What does an oscilloscope do?

-acquire data synchronously (needs trigger)

-record and display it (needs a timebase)

Oscilloscope users implicitly expect the acquisition to be synchronous.

A distributed system like a collection of EPICS IOCs is not able to do these actions without help:

-trigger and sync distribution for acquisition

(for event system, handled with drivers)

-time distribution as a common timebase

(handled in drvTS)

With hardware assistance, we can guarantee exact synchronism

Timestamp distribution is an indispensable part of the system.

Good, Bad and the Ugly

- Good things in drvTS
  - Support of synchronous timestamps
  - The way how the events are integrated (fits our system well)
  - Many of the concepts are worth of keeping
- Bad (or not that good) things
  - Custom soft time protocol (which was of course required when drvTS was written – then there were no real alternatives to it!)
  - Lack of flexibility
  - Complex, has lots of hidden ‘features’ that cause surprises
  - Not extensible, lack of external interfaces

- It has been our wish a long time to rewrite the code
  - However, just rewriting is not a solution.
  - drvTS was an **architecture** and thus needs to be replaced by **another architecture** that fulfills a wider spectrum of needs
  - Together with other sites and complementary requirements we should be able to come up with a good solution
  - Need to clarify the way to go

So, what do we want to have?

- A robust system that is able to deliver timestamps even in the case of
  - Timing distribution failure
  - NTP server (temporary) failure
- In failover, jumps in time should be avoided, (especially jumps backward)
  - Causes problems in archiving and (later) analysis of data
- System whose behavior is specified, controllable and intuitive
  - drvTS tries (or tried) to be clever and guess things for the user
  - Behavior should be configurable (automatic switching to a ‘better’ source is not always what we want. The timing for instance may be online but not fully configured. The old drvTS tried to switch automatically as soon as it saw the timing master alive.)
- A system that can support a mixture of platforms (reality of today)
  - VME IOCs, PCs, embedded/micro IOCs, SoC (FPGA) IOCs…
  - vxWorks, RTEMS, Linux, Windows,….
- Should be able to support different time sources
  - Live together with standard computing infrastructure (NTP)
- Should be able to support real-time and pulse-based timestamps
  - For non-pulsed machines we need real time!

What have we done so far?

- Fixed a lot of bugs in drvTS (startup problems,…)
- Added a fallback mechanism that avoids jumps in time
  - The drvTS soft timer is always activated and kept in sync with the HW timing
  - If there is a problem with the event stream, switch to soft time
- Removed unwanted automatism, added record support to access modes
  - Automatic fallback to soft time, record access to return to HW time (could be automatized)
- Implemented a PLL algorithm to keep master clock IOC in sync with NTP
  - New idea: modulate the clock at the source; slaves can be kept in sync without any extra communication or computation on the slave side.
  - This allows us to overcome the problem with the broadcasts and subnets
- Provided a clean interface to routines for timing boards (like a device support)
  - Dropped the ugly symbol table search for function names
- Implemented monitoring & control of clock parameters via soft device support
- Added a possibility to force an IOC to maintain a direct clock (NTP sync)

Where are we, where do we go?

- Most of our ideas and requirements are common to those addressed by General Time
- We (too) feel it necessary to have a common, clearly defined strategy to support timestamps
  - A common base service
  - Support for many (if not all) platforms
  - User (meaning site) configurability, defined exception handing (fallbacks)
- It makes no sense to develop something for ‘ourselves’ only
  - Many sites are using the (Micro-Research) event system and could benefit from the work. So far we have not distributed what we did with drvTS
- A clear strategy to proceed is required. We have not been sure of the way to go
  - When the road is not clear, this work tends to be overrun by other, allegedly more urgent things.
- However, as soon as we agree which way to go, we could produce results pretty quickly
  - We have had the improved drvTS more than 2 years in operation and believe that we understand the issues pretty well
  - However, when trying to move to 3.14, we were puzzled of how to go on

Our Proposal

- Adopt General Time as the unified approach
  - We (SNS & PSI) share the code and develop the first version
  - Distribute that to other interested in a couple of months
- This has to be supported on multiple platforms (anything else would be plain stupid)
  - We need to enlarge the community: we cannot test all possible combinations
  - The specs should be finalized and distributed (when the direction is clear)
- We (PSI) would like to have a few modifications and enhancements to the API
  - Possibility to control the switching (we do not always want to automatically switch back from fallback to the 'better' timing; both ways should be possible)
  - We have some considerations about how to implement the switching
- We strip the soft timing protocol away from what is now called drvTS (and try to figure out a better name for it.)
  - drvTS will be made available with a driver interface, compliant to General Time API or enhanced version of it
  - We need to check how we best modify our failover
- We want to take an analytical look at the system-wide synchronization issue in the mid term (but after the first version is out)

Distributed systems

drvTS is (for historical reasons) available for vxWorks and VME-based IOCs only. In our opinion, this is not good. Reasons:

-hardware that supports synchronized timing is being used in other configurations:
RTEMS&VME, RTEMS & PMC, Linux & PMC, even Windows & PMC,…could be anything
-this implies: anything other than OSI does not make any sense.