Synchronized Time Stamp Support

Jim Kowalkowski

1.0 Introduction

1.1 Purpose

New software has been added to IOC core to maintain time stamps. The new software has the ability to maintain time stamps over all IOCs on a network. The purpose of this paper is to explain how EPICS will synchronize the time stamps. In addition, this paper will explain how to configure and use the new EPICS time stamp support software.

1.2 Definitions

Time Stamp: Two long words representing the time in seconds/nanoseconds past an epoch. The first long word represents the seconds past the epoch, the second long word represents the nanoseconds within the second. The EPICS epoch is currently January 1, 1990. A commonly used Unix epoch in January 1,1900.

Event System: A hardware based subsystem for delivering events to all IOCs on a network. Typical events are the “tick” event and the “reset counter” event.

Event System Synchronized Time: Time stamps maintained using an event system. The clock pulses which increment the time stamps are provided by a single source, all IOCs increment time stamps using the single source pulse.

Soft Time: Each IOC increments the time stamps using it’s own internal clock.

Master Timing IOC: The IOC which knows the actual time and is the source of the actual time to all those who inquire.

Slave IOC: An IOC which relies on a master IOC to provide the actual time. This IOC will keep it’s time stamp in sync with a master.

Soft Slave: An IOC which uses Soft Time and synchronizes it’s time stamp with a master.

Event Slave: An IOC which uses Event System Synchronized Time to maintain it’s time stamp. This type of IOC uses the master to verify that it’s time stamp is correct.

Soft Master: A Master Timing IOC that maintains it's time stamp using a private clock.

Event Master: An IOC which uses Event System Synchronized Time and is a Master Timing IOC. In addition, this IOC is the source of the clock pulses.

1.3 Overview

Time stamps are maintained across IOCs using UDP with a master/slave relationship. A master timing IOC is responsible for knowing the actual time. A slave IOC uses the master to verify it's own time stamps. Time stamps are maintained in two fashions: using an event system or using IOC tick counters. Both differ radically and require some explanation.

1.3.1 Event System Support

The time stamp software has special requirements in order to provide event system synchronized time. The time stamp software assumes that hardware is present which has finite counters for counting pulses or ticks delivered to it from some master tick generator. The software further assumes that every so other, before the finite counters overflow, a signal or event will occur to reset the counters to zero. The software requires access to these counters at any time and notification when the "counter reset" arrives. The combination of hardware tick counters and reset event notification gives the software the ability to maintain time. If the event system supports other events, it is required that the software be notified of the occurrence of each event. The event time will be recorded in a table which holds one time stamp for each possible event so that a user can inquire as to when the event occurred.

1.3.2 Soft Time Support

If no event system is present, the time stamp software can operate in a software timing mode. In this mode, no events are available, only the current time can be retrieved. An IOC using soft time will increment a time stamp using the system clock (usually updated at 60Hz). The master/slave relationship allows a slave to periodically ask a master for the correct time. At this point the slave's time stamp can be adjusted to match the masters.

1.3.3 Role of a Master Timing IOC

A master timing IOC has difference responsibilities depending on if it is an event master or a soft master. An event master will be an event generator (create "tick" and "counter reset" events). When the event master detects a "counter reset", it broadcasts the time the event occurred to all event slaves on the network. The slaves can use the time stamp data to verify their own time. A soft timing master does not have any events, so no broadcasting is done. Both soft and event masters have the ability to be queried at any time by slaves for the current time.

1.3.4 Role of a Slave Timing IOC

A slave timing IOC also has difference responsibilities depending on if it is an event slave or a soft slave. An event slave contains hardware to manage “tick” and “counter reset” events. The time stamp support software uses this information to maintain time. An event slave will listen for broadcasts from the event master timing IOC and use the information to verify it’s time. A soft slave periodically queries a master timing IOC to get the current time.

1.4 Design Specifications Summary

- Event System Synchronized Time

    All IOCs will have identical event times. In addition, all IOCs will maintain the same current time.

    The minimum events which must be supported is two. One must be used as a “tick” event and one must be used as a “reset hardware tick counter” event. The second will be used to update the time stamp representing the current time.

    An optional event can be used as a “heartbeat” event. This event can be used to signal errors.

- Soft Time

    All IOCs of this type will maintain time stamps which are within two clocks ticks or 1/30th of a second of a master. The master may be a designated IOC or the Unix boot server. A Unix boot server master must have NTP available for polling.

2.0 User Level Interface

2.1 General Use Functions

Three functions exist in the synchronous time stamp support software for the user to retrieve time stamps:

- TSgetTimeStamp(event_number, struct timespec* time_stamp)
- TScurrentTimeStamp(struct timespec* time_stamp)
- tsLocalTime(TS_STAMP*)

To retrieve the time stamp which represents the time that an event occurred, use TSgetTimeStamp(). TScurrentTimeStamp() can be used to retrieve the time stamp which represents the time of day. TsLocalTime() is the function for returning the current time stamp in the old time stamp driver, the routine can still be used to retrieve this information.

2.2 Record Support

Record support will have the ability to tie the record processing time to an event. This means that a user can specify that processing of a record is due to an event (from the event system). When the record gets processed, the time in the TIME field of the record will be the time when the event occurred. In order to support the event times from record support, two new fields will be added to dbCommon. The fields will be Time Stamp Event (TSE), and Time Stamp Event Link (TSEL). The TSE field will be the actual event time the user is interested in. The TSEL field will be a link used to populate the TSE.

To facilitate the use of the time stamp support software, a new record support function will be added:

```
recGblGetTimeStamp((dbCommon*)prec).
```

This routine uses *TSgetTimeStamp*(prec->tse, &prec->time) to set the processing time of the record. The new recGblGetTimeStamp() will replace the existing call to tsLocalTime(). If the TSE field is zero (the default), then *TSgetTimeStamp*() will report the **current time**. It is important to remember that if a TSE field is set, then the processing time (in field TIME) will always reflect the last time the event occurred.

3.0 Driver Configuration

The synchronous time stamp support software is configured by calling TSconfigure() from the “startup.cmd” file. The parameters to this routine are:

master_indicator:	1=master timing IOC, 0=slave timing IOC, default is slave.
sync_rate_seconds:	The clock sync rate in seconds. This rate tells how often the synchronous time stamp support software will confirm that an IOC clock is synchronized. The default is 10 seconds.
clock_rate_hz:	The frequency in hertz of the clock, the default is 1000Hz for the event system. The value will be set to the IOC’s internal clock rate when soft timing is used.
master_port:	The UDP port which a master timing IOC will use to receive time stamp requests. The default is 18233.
slave_port:	The UDP port which a slave will use to receive time stamp information from a master.

This routine must be run before iocInit(). The synchronous time stamp support software is initialized as part of iocInit. Running TSreport() after iocInit() will produce a report which shows the current state of the driver.

4.0 Event System interface

The event system interface consists of seven function which can be provided by an event system. The synchronous time stamp support software uses a card number of zero on all functions that require a card number. The functions are as follows:

- ErHaveReceiver(int event_card_number)
- ErGetTicks(int event_card_number, unsigned long* ticks)
- ErRegisterEventHandler(int event_card_number, EVENT_FUNC event_function)
- ErRegisterErrorHandler(int event_card_number, ERROR_FUNC error function)
- ErForceSync(int event_card_number)
- ErGetTime(struct timespec* time_stamp)
- ErSyncEvent()

The definition are as follows:

ErHaveReceiver()	Returns -1 if no event (timing) hardware present, else returns the number of supported events.
ErGetTicks()	Returns the number of ticks since the last hardware tick counter reset.
ErRegisterEventHandler()	Informs the event system of a function to call when an event occurs, the format of the function will be defined below.
ErRegisterErrorHandler()	Informs the event system of a function to call when an error occurs, the format of the function will be defined below.
ErForceSync()	This function will force an event generator to generate a tick reset event and send it.
ErGetTime()	This function returns the actual time. The intention here is that this function will retrieve the actual time from GPS system or equivalent and return is in time stamp format.
ErSyncEvent()	Return the event number for the tick reset event.

All of these routines are checked to exist when the time stamp support code initializes. All have some kind of default routine provided if they are not found, most of which just return an error condition. The functions which the time stamp support software registers (event and error) have the following format:

- TSeventHandler(int Card,int EventNum,unsigned long Ticks)
- TSerrorHandler(int Card, int ErrorNum)

Here the *Card* is the event system board of interest (always zero), the *EventNum* is the event that occurred, and the *Ticks* is the number of ticks since the last board tick counter reset.

5.0 Operation

An IOC can be configured to run in one of four ways: synchronous-master, soft-master, synchronous-slave, soft-slave. Each mode maintains time differently. When the time stamp support code initializes, it determines the mode which it will operate in based on configuration parameters and event system function information.

5.1 Synchronous Time

Synchronous timing is determined by the presence of event system hardware. The function ErHaveReceiver() gives this information. All IOC's using synchronous time will have the exact same time stamps. The "tick" event increments all IOC's event system tick counters, the "reset tick counter" resets the counter to zero and posts the event to the time stamp support software. The time stamp support software system knows the current time by looking at the last "tick reset event" time and querying the event system with ErGetTicks() for the number of ticks which have elapsed since the last reset. Since the event handler function gets called with the "ticks since last reset", an event time is the last "tick reset event" time plus the ticks count in the call.

5.1.1 Master

A synchronous master is determined by the first parameter to TSconfigure() and the presence of event hardware. A master must have an event generator and an event receiver. Since the event system is configured from EPICS database records, the database must have records in it to initialize the event system. The master is responsible for providing the "tick" event and the "reset tick counter" event.

When the time stamp support software's event handler function on a master timing IOC receives a "reset tick counter" event, a time stamp message is broadcast out on the slave_port (configured by TSconfigure()). The master timing IOC is also listening on the master_port (also configured by TSconfigure()) for incoming requests for time stamp information.

At boot time, a master will set the vxWorks clock from the Unix boot server time. The time is retrieved from the boot server using NTP or the time protocol. The GPS module takes an unknown period of time to sync to the correct time, so it can not be used until it's time is valid. The event system is not up and running until record support initialization is complete, therefore the event system time stamps are not ticking until the event system initializes. At the time of the first sync, the event system is known to be up, and the "reset tick counter" or "sync" event is set to the current time (vxWorks clock).

5.1.2 Slave

A synchronous slave is determined by the first parameter to TSconfigure() and the presence of event hardware. This mode is automatically selected if no TSconfigure() call is used in the "startup.cmd" file. This type of slave must have an event receiver. The EPICS database must be configured to initialize the event system.

When a synchronous slave configuration is determined, the IOC broadcasts a request for master on the master_port port. If a master is found, then all current time, sync rate, and clock are extracted from the master's response (the sync rate and clock rate from the TSconfigure() are overwritten to match the master's configuration). If a master is not found, the slave IOC goes into a polling mode to try to find a master every two minutes. While a slave has no master, The IOC's clock is initialized from the Unix boot server and TSgetTimeStamp() returns the vxWorks time clock value.

The slave experiences the same problems as the master upon boot.

5.2 Soft Time

Soft time IOCs use the 60 hertz clock available from vxWorks to maintain a time stamp. The IOC will increment a time stamp at a 60 hertz rate. Soft time is determined by the absence of event system hardware. All soft timing IOC's will not have the same time stamps.

5.2.1 Master

A soft master is determined by the first parameter of TSconfigure() and the absence of event hardware. Upon boot, the master soft timing IOC retrieves the current time from the Unix boot server. The IOC sets up a soft time-stamp counter using a one tick watch dog and sets the vxWorks clock. From this point on, the master runs using the soft time-stamp counter. The master listens on the master_port port for requests for the master's current time.

5.2.2 Slave

A soft slave is determined by the first parameter of TSconfigure() and the absence of the event hardware. This mode is automatically selected if no TSconfigure() is present in the "startup.cmd" file. The basic operation of a soft slave is to synchronize the time stamp with a master when possible or to the Unix boot server if no master is available.

Upon initialization the time stamp support code determines if a master is present on the network, and if NTP support is available from the Unix boot server. The slave will request time stamps from a master or unix server at sync_rate_in_seconds rate from TSconfigure(). If the time stamp on the slave is found to be off, the clock will be incrementally adjusted so that by the next sync, the clock will be corrected to match the master's time stamp. A soft slave will automatically switch between a master and the unix boot server depending on if the master is available or not. In order to sync with the unix boot server, NTP must be available there for query only.