# *Timing System Observations*

**John Winans**
Argonne National Laboratory
Advanced Photon Source
Accelerator Systems Division/Controls Group
March 1, 1994

## 1. Purpose

The purpose of this document is to augment *Synchronized Time Stamp Support* by Jim Kowalkowski.

## 2. Soft Timing

### Soft Timing Configurations

If there is no event system in place, the possible configurations for a specific IOC are:

- Soft Slave With UNIX Master (NTP)
- Soft Master
- Soft Slave With Master IOC

By default, the "Soft Slave With UNIX Master" mode is used. The other options are only possible upon using the `TSconfigure`() command in the `startup.cmd` file.

### Soft Slave With UNIX Master (NTP)

This mode of IOC timing operation will initialize by opening a socket to the UNIX master that it booted from. And it will be used to monitor NTP time stamps and synchronize the IOC's internal 60Hz clock to it. The IOC does not use the full implementation of the NTP client protocol. It uses the technique described in RFC 1361 "Simple Network Time Protocol."

Note, however, that incremental clock adjustments are made to the internal clock (this is an enhancement of RFC 1361). Use of this service typically results in the slave being out of sync by no more than one click tick.

### Soft Master

This is *ONLY* appropriate if the IOC contains a better time source than the UNIX NTP time service can provide. This mode is enabled by the use of `TSconfigure`(). When employed, the IOC will use the accurate local time source and respond to slave polls that are used to search for a soft master timing IOC. The result will be that soft slaves will choose this IOC as a timing choice over the use of the UNIX NTP service. Use of this service will result in time as accurate as the local source.

### Soft Slave With Master IOC

This mode is entered only if a soft slave finds a soft master IOC on the network. It locates the soft master by broadcasting for it periodically. If a soft master is found, a simple time sync protocol (not NTP) is used where the soft slave periodically polls the master for the time. This typically results in the slave being out of sync by no more than one clock tick.

## 3. Synchronous Timing

### Synchronous Timing Configurations

If an event system is located during initialization, the IOC will be configured as a synchronous slave. Options for synchronous operation are:

- Synchronous Master
- Synchronous Slave

By default, the "Synchronous Slave" mode is used. "Synchronous Master" mode is enabled by the use of the `TSconfigure`() command in the `startup.cmd` file.

### Synchronous Master

This mode is entered when `TSconfigure`() requests it at init time. In this mode the time source of the IOC is defined by the presence of the `ErGetTime`() function. If it is present, it is used to get the initial time. Presumably, GPS drivers will supply `ErGetTime`(). After the initial time has been set, it uses the clock tick mechanism of the event system to maintain the time (ie. `ErGetTime`() is never called again.)

In this mode, the IOC will broadcast resync messages when resync events are detected on the on the event system. This will maintain the exact same tick count value on all synchronous slaves that is accurate to the propagation delay present in the event system.

### Synchronous Slave

This is the default mode of operation when an IOC boots with event receiver hardware present. In this mode the IOC will get its time from the tick counter on the event receiver hardware. And resync with the master by receiving the resync event code followed by the resync time stamp from the synchronous master.

## 4. Additional Observations

### IocCore

DrvTS is actually part of the IocCore proper. It is not a regular driver because it is used internally by IocCore. (This is the same as the old time stamp support.)

### Globally Synchronized Time stamps

It is *NOT* possible to get synchronized time stamps on multiple records that are on different (or even the same) IOCs unless an event system is used to initiate the record processing. And those records refer to the event system as the source for their time stamps. (Event system in this context is not the same as the EPICS event records.)

### Dependancy on Event System

It is determined that an event system is in place by performing an init-time symbol search for the required support functions. The required support functions for an event system are described in the document "Synchronized Time Stamp Support." Should the required functions not be located by the symbol search, the IOC will be configured for 'soft' or 'async' operation.

### Compatibility

The timing driver provides `tsLocalTime`() so that any legacy code will still operate modification free. However, the AT-8 event system interface is incompatible for use as a time source. It would require the addition of event registry functions described in "Synchronized Time Stamp Support."

### The EPICS Time Stamp Format

The EPICS time stamp format is described in "Synchronized Time Stamp Support." However, it might do to mention it before proceeding to leapsecond handling. The timestamp is based on how many seconds have passed since an epoch. The timestamp uses the standard formula for leap years and assumes that all days have 60*60*24 seconds in them. This is not true for days that include leapseconds. Therefore the actual count of seconds that have passed since the epoch are calculated improperly. This, in itself, is not really a problem unless some IOCs, somehow, account for the leapseconds and others do not.

### On the Dealing With Leapseconds

There is no reasonable way to deal with leap seconds since the EPICS time stamp does not account for them. Therefore, any leap seconds that occur after January 1 1994 will not be accounted for (prior leapseconds have been hard-coded into the interpretation code -- same as the previous time stamp driver.)

The results of this is that leap-second sensative time sources may appear to have a different time than the IOC that is syncing to it (this will be the case for GPS based time sources.) The problem with this situation is that if a master timing IOC (that uses this type of time source) is rebooted after sustained operation across a leap second, the initial time stamp value it calculates will contain a different number of seconds from the epoch than any other master timing IOC will currently have, that sustained operation across the same leap second and has not yet rebooted. This will result in inconsistant values for the time stamps on the different IOCs.

The solution to this problem would be to require that all master timing IOCs that use leap-second accurate timing be rebooted, if ANY of them get rebooted (after a leap second event.) Or to only allow one single master timing IOC.