Frequently Asked (and sometimes wrongly answered) Questions about Timing

By MIrcea Neacsu

1. Can you explain the latency issue in HYPACK®?

Time is extremely important in HYPACK® because each piece of data received from a device is immediately stored in the raw data files. In order to correlate it with data coming from other devices, we need to know the time the measurement was taken. The great majority of devices do not give the time of the measurement as part of the data string so we are forced to assume that the arrival time of the message is the actual time of the measurement.

To compensate for the difference between the arrival time of the data (that we use as our time-tag) and the real time of the measurement, we can subtract a ”latency” time which is exactly that: the time interval between the measurement and the arrival of the data string with that measurement.

Let's take a simplified example of a GPS receiver: On the integer second, the receiver measures the signal propagation time from each of the satellites and starts calculating its position. After solving the many equations involved and feeding the results through some Kalman filters and what not, it finally generates a position value that is sent out through a NMEA GGA message. By the time we receive the GGA string, the boat has moved from where it was at the integer second and the depth values coming from the echosounder are also quite different.

The timeline would be:

TABLE 1. *Sample Time Line*

Computer time	GPS time	Action
12:30:50.000	12:30:40.000	GPS receives satellites signal and begins calculation
12:30:50.010		HYPACK receives echosounder string saying: depth=10.3
12:30:50.110		echosounder: depth=10.5
12:30:50.210		echosounder: depth=10.7
12:30:50.310		echosounder: depth=10.8
12:30:50.400	12:30:40.400	GPS string $GPGGA,123040.0,1213.1415,N,7020.567,W,....
12:30:50.410		echosounder: depth=10.9

Looking at this time line we can see a number of things:

- Most importantly, the depth at the integer second position changed from 10.3 to 10.9. When the GPS did the measurement of ranges to each satellite and started computing its position the echosounder was reporting a depth of 10.3. By the time the GPS sent the data to HYPACK®, the echosounder depth had changed to 10.9.

  This is a simplification as HYPACK® does not assign depth values to positions; instead it assigns positions to depth values, but the effect is the same for our purposes.
- The GPS position string contains the time of validity of the position (12:30:40.0), but we cannot use it because the computer clock is not synchronized to the GPS clock.

- Data coming from the echosounder does not contain any timing information so we must continue to use the computer clock for it and we can only hope that we get the data quickly.

If we knew that the GPS latency is 0.4 seconds we could just **subtract** this latency from the arrival time of the GGA message and everything would be fixed.

2. How can I determine the latency?

The simplest method to determine the latency is to run the latency test described in our manual: you run the same line over a sloping bottom in opposite directions and feed the two data files to the LATENCY TEST program. The program goes through the data files, adds a latency value, recalculates sounding positions with that latency value, then calculates the depth differences at each point between the two files. It then changes the latency value and repeats the whole process. With a bit of luck, you end up with a U-shaped curve of average depth differences and the bottom of the curve indicates the best fit between the two profiles. That is the latency value indicated by the LATENCY TEST program.

The resulting latency is, not only the latency of the GPS but more of a global system latency involving the GPS, computer and echosounder. All gets combined in the calculated latency value, but we choose to call it GPS latency because, traditionally, the GPS is responsible for most of it. Once a latency value is determined it can be entered into the HYPACK® HARDWARE configuration (as GPS latency) and that value will be **subtracted** from all time tags for that device.

3. Is it true that latency is subtracted from time tags? I thought all corrections are added…

Yes it is true: as opposed to all other corrections, the latency value is **SUBTRACTED** from the time tags. This is one of the rare cases where we write “corrected” values in raw data files.

Why it is like that: What fun would be a rule that doesn’t have exceptions? Besides, now it is too late to change anything.

4. Why should I bother with synchronization when I can correct my data with a latency correction?

Errors affecting the calculated latency value:

- Not running the two lines exactly one on top of the other
- Changes in depth correction factors (tide, draft, sound velocity)
- Jumps in position due to poor GPS positioning or reflections (bridges, piers, other vessels, etc.)

Also, the latency value can change depending on factors such as the number of satellites.

The solution: We can to try to synchronize the computer clock with the GPS clock and take advantage of the time of applicability contained in the GPS messages. If we can synchronize the two clocks, we can use the time tags provided by the GPS receiver for data coming from it and the synchronized computer clock for data coming from other devices like the echosounder. There is a NMEA message that seems almost perfect for this purpose: the ZDA message contains just the GPS time and date.

5. What is this VERITIME thing?

Many years ago we used a fairly naive approach to synchronization where we would take the ZDA message and set the computer clock to the time indicated in that message. The problem with this approach is that the two clocks can (and will) drift in time and, unfortunately, you cannot redo the synchronization in the middle of logging a data file because it would wreak havoc with all the time tags.

Enter the VERITIME clock model: This is a software loop (those with electronics background would call it a PLL loop) that constantly monitors the difference between the GPS clock and the computer clock and slowly changes the computer clock to match the GPS clock. If you turn on synchronization and you run the same latency test you should obtain a latency very close to 0.

6. Should I still run a latency test when using synchronization?

Yes, it's a darn good idea (keep reading).

7. Is the latency always zero when using synchronization?

Sometimes the resulting latency is still quite appreciable and this has to do with how the ZDA message is sent.

In the NMEA standard there is no, specification that the ZDA message should be exactly on the integer second or about the accuracy of the time indicated in the message. If we look at a typical ZDA message:

```
$GPZDA,163448.014,09,03,2011,00,00*57
```

we see that the GPS unit bothered to output the time to milliseconds resolution so we can hope that it started to send the ZDA message at that exact moment. This is however not the norm for all GPS receivers. Other receivers could send the same message as:

```
$GPZDA,163448,09,03,2011,00,00
```

(without any milliseconds) and send it somewhere during that second but without paying much attention to its timing.

8. What happens if the GPS sends ZDA messages more often than once per second?

Complete mayhem! VERITIME will jump off its rails and very soon you will see the dreaded "SYNC FAIL" error in the SURVEY program. For fans of electronics, each PLL loop has a certain bandwidth and sending ZDA messages more often than once per second falls outside the VERITIME bandwidth.

...What about sending ZDA messages less often once per second (once every 2 or 5 seconds)?

Yes, that works but it probably doesn't do you any good. Because VERITIME adjusts the clock model only when it receives a ZDA message, you are allowing the two clocks to drift apart for a longer interval.

9. Can I make it better?

The next improvement is to **use the pulse per second output from the GPS receiver**. This is an electrical pulse that we bring through a small level converter to a RS-232 pin (the CTS signal) and use it as reference to our PLL loop instead of the ZDA message. The ZDA message is still needed because we need to know the clock reading when the PPS pulse occurred. This is the most accurate method one can currently use to synchronize the GPS and computer clock.

As a rule of thumb from the data I've seen so far (and I probably get to see more bad data than good) the latency errors you can expect are:

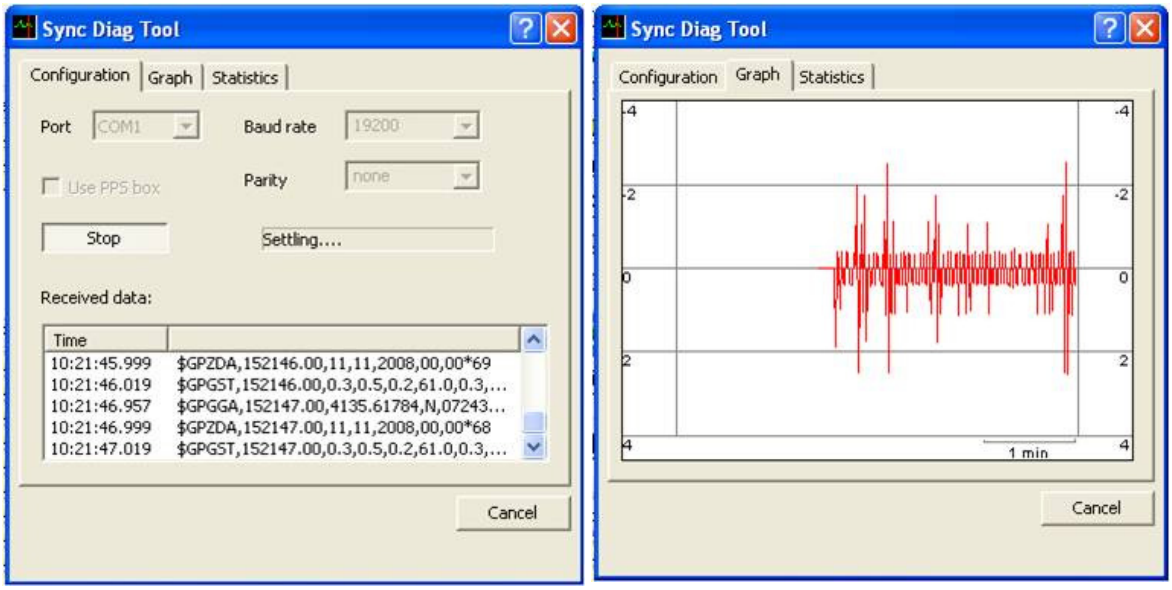
- **Using LATENCY TEST program:** 50 to 100ms but can change unexpectedly.
- **Using ZDA message for time sync**: 20-30 ms; good enough for most single beam surveys
- **Using PPS signal for time sync**: 1-5 ms; good for any multibeam work.

To use the PPS pulse you need a "PPS box" that does the electrical adaptation of the GPS signal to the RS-232 levels.

10. How can I check my synchronization or compare the PPS pulse and the ZDA message?

Under the "UTILITIES -CALIBRATION" menu there is the "ZDA TEST" program. It is a simple-to-use program that shows a graph of the synchronization error between the GPS clock and the computer clock.

FIGURE 1. *Sample ZDA TEST*



11. HELP! I CHANGED MY COMPUTER AND NOW SYNCHRONIZATION DOESN'T WORK.

Take heart in knowing that you aren't the only one. We have seen this happening on a fairly regular basis. Recently, we have found out about a possible hardware cause: if your computer has an AMD core the following information found in the Microsoft knowledge base might apply to you.

Article ID: 821893 - Last Review: January 7, 2009 - Revision: 5.0

The system clock may run fast when you use the ACPI power management timer as a high-resolution counter on Windows 2000-based, Windows XP-based, and Windows Server 2003-based computers

View products that this article applies to.

Expand all | Collapse all

SYMPTOMS

When a Microsoft Windows 2000-based, Windows XP-based, or Windows Server 2003-based computer runs in Advanced Configuration and Power Interface (ACPI) mode and uses a high-resolution counter, the system clock may run fast.

Back to the top

CAUSE

This issue may occur if the time increment in a program changes and the Hardware Abstraction Layer (HAL) cannot measure the time interval between successive clock interrupts. This causes the system clock to lose a short period of time. When the HAL misses many time-interval measurements in quick succession, the time loss may be significant.

Note

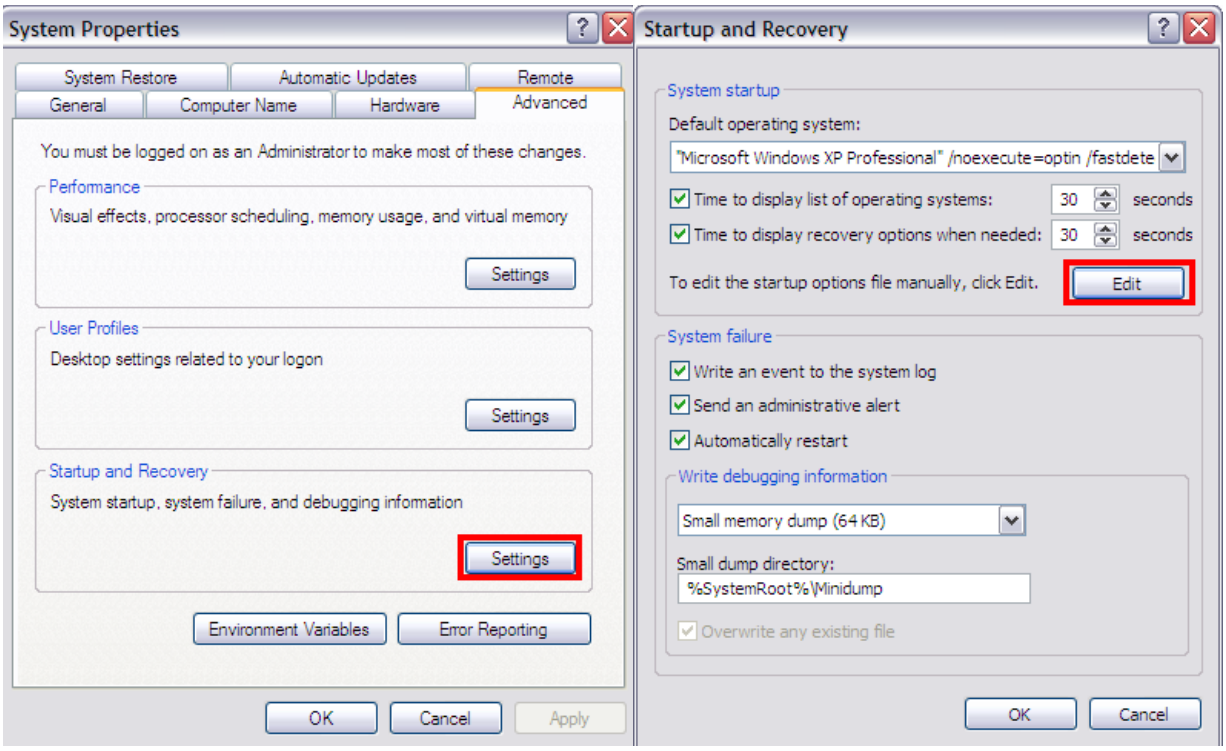
- This issue may occur on a computer that is running Halaacpi.dll (UP, ACPI, or APIC), Halmacpi.dll (MP, ACPI, or APIC), and Halmps.dll (MP, non-ACPI, or legacy) because these DLLs use the Real Time Clock (RTC) to generate clock interrupts.
- This issue does not occur on a computer that is running Halacpi.dll (UP, ACPI, or PIC) or Halx86.dll (UP, non-ACPI, or legacy) because these DLLs use the 8254 Programmable Interval Timer (PIT) to generate clock interrupts.

Back to the top

Possible workarounds:

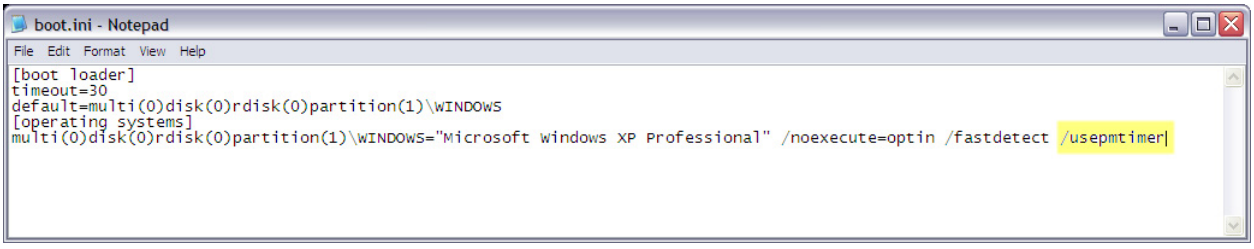
- **For Windows XP systems**
  a. **Open the System properties dialog box** (Control Panel -> System) and select the “Advanced” tab. (Figure 2 right)
  b. **Under ‘Start up and Recovery’, click [Settings]**. (Figure 2 left)

FIGURE 2. Accessing System Startup and Recovery Settings



c. **Add the string "/useptimer" at the end of the line with the operating system**.

FIGURE 3. The Results: Default Operating System



d. **Save the file and reboot your system**.

- **For Windows Vista and Windows 7** try to disable "Intel SpeedSetp Technology" in your BIOS settings.

We are still investigating this issue and will keep you posted on any developments.

12. WHAT IS THIS OPTION "USE GPS TIME-TAGS EVEN WHEN NOT SYNCHRONIZING"?

The full wording of this option should be something like "*record all data coming from GPS using the time-tags provided by the GPS unit"* but this is too long to fit in any decently sized dialog box.

Let's start with a big warning:

Beware! *The option applies only for data coming from that particular device*. It doesn't affect the data coming from other devices.

Getting back to the sample time line in Table 1, if the echosounder would be smart enough to send UTC time-tags, we could take the GPS time tags from the GGA message, the echosounder time-tags from the echosounder string and HYPACK® wouldn't have to worry about any of these latency issues. This is the lazy approach to synchronization: let someone else do the work!

This situation occurs mostly in multibeam surveys where those hyper-expensive echosounders are also fed the PPS pulse from the GPS. It's a process similar to using HYPACK® VERITIME but it sends the data with UTC time-tags. There are similar options for the POS/MV and iXea F180 drivers.

Important!! The option can be used only if **all** devices are synchronized outside of HYPACK®. Maybe I wasn't clear enough: ***ALL*** devices have to be synchronized outside of HYPACK®. And by "all" I mean *really all*.