SYNCHONOUS SAMPLING FOR DISTRIBUTED EXPERIMENTS

Markus Wittkamp and Josef Ettl

Deutsches Zentrum für Luft- und Raumfahrt (DLR), Mobile Rocket Base, markus.wittkamp@dlr.de

ABSTRACT

Sounding Rocket payloads, especially for atmospheric research, often consists of several independent sensors or experiments with different objectives. The data of these sensors can be combined in the post processing to improve the scientific results of the flight. One major requirement for this data-correlation is a common timeline for the measurements of the distributed experiments.

Within this paper we present two ways to achieve absolute timing for asynchronously working experiments. The synchronization process is using the Global Positioning System (GPS) and a standard serial communication protocol for transport of timestamps and flight-states.

Key words: Timing, GPS-Synchronisation, Distributed Clocks, Distributed Sensors, Time Synchronisation.

1. TIME SYNCHRONISATION

Traditional PCM[1] based telemetry systems are controlling the sampling process by sending special synchronisation signals to the analogue-/digital converters (ADC) of the experiments. This forces a specific timing for all devices of the payload. Consequently all instruments must be able to read these signals. The advantage of this technique is the absolute synchronous sampling of all connected devices. If an already existing device should be reused together with a different telemetry system, usually the electrical interface has to be adjusted to the new telemetry unit.

Modern sensors often using embedded processors for internal operations like controlling the experiment and preprocessing of data. Usually these operations are based on internal timing and the data forwarding to the telemetry is taking place whenever the routines are finished. Beside of the need for buffers and special data handling in the telemetry system, a way of time-synchronisation is needed, if data of this kind of sensors should be combined or fused with data of other devices.

[1]Pulse Code Modulation

Fortunately these units often offers communication capabilities like the UART[2] protocol on RS-232 or RS-422 electrically interfaces. This serial protocol can also be used to synchronise the internal processing, sending information about the flight-state, flags, timestamps and more. It is fully deterministic and therefore suitable for real-time applications.

The techniques described below have already been used on the two WADIS [1, Page 7] payloads and are intended to be used on MAIUS [2]. They were first introduced in [3].

1.1. Generating a Reference Clock

In order to send synchronisation messages to the experiments, a reference clock is needed. Some GPS-receivers are providing a Pulse of Second (SP) with a fixed phase alignment to full integer seconds. This SP can be used to derive such a clock. The reference clock will keep the phase of the SP, similar to a phase locked loop.

The algorithm for generating the reference clock is based on a counter to build up a resettable clock divider. This counter is driven by a local oscillator, running on a much higher frequency than the desired frequency of the reference clock (f_{rc}) should be. The local oscillator frequency (f_{lo}) doesn't need to be synchronized to any other oscillator or the GPS. The counter is running from zero up to the value of $M - 1$, except it was reset by the GPS SP. Finally the frequency of the reference clock is

$$f_{rc} = \frac{f_{lo}}{M} \tag{1}$$

Due to the reset to zero at the the rising edge of the SP, the counter stays phase aligned to the integer second of the GPS. A possible frequency offset of f_{lo} will also be adjusted at the end of the full second. The consequences of clock and phase errors due to the free running oscillator are discussed later in section 3.

To transform the counter value into a square wave signal for the reference clock, the output level is set to logically high if the count is less than $M/2$ and to low if it is equal or above this value. Since this divider is designed to be

[2]Universal Asynchronous Receiver Transmitter

part of a larger FPGA[3] design, the reference clock can also be used for other purposes inside or outside of the chip.

1.2. Generating Trigger Words

Beside of internal timing, the reference clock is used to generate synchronisation messages with an experiment dependant frequency. They are called the trigger words or trigger messages. Within this paper the term *trigger word* is used for single word messages, whereas trigger messages containing more than one word. Formally trigger words are also trigger messages. These messages are to be send out to the experiment devices using the UART protocol, keeping their phase aligned to the reference clock and the original GPS-SP. Since the trigger words are serial messages of at least one word length, they can also be used to carry additional information.

If trigger words are used to carry flags without any other data, they are just marking timestamps. This will generate precise relative timing, but the resulting internal experiment time is still ambiguous. In order to achieve absolute timing, a unique event is needed to serve as a reference point. For sounding rockets this event might be *Lift-Off*. Lift-Off is generated when the umbilical is extracted due to the movement of the vehicle on the launch rail.

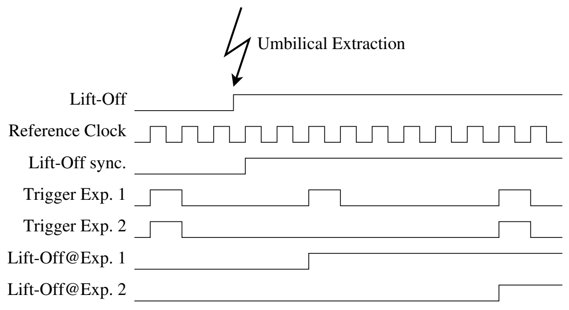


Figure 1. Sampling of the Lift-Off signal and forwarding the information to the experiments by using trigger words.

Figure 1 is showing how the timing of trigger words works:

1. The umbilical extraction event is sampled with the rising edge of the reference clock, generating an internal, synchronized Lift-Off signal. This sampling is also important for the internal processing of the signal on the FPGA implementation. However, the resolution in time and consequently the maximum uncertainty is the period of the reference clock.
2. Individual trigger words are generated for each experiment unit, carrying the Lift-Off-State. Each experiment device is getting these trigger words on an individual frequency and with its own settings for the UART transmission, e.g. Baudrate, usage of Parity, etc. The uncertainty of the trigger time for an individual device is the sampling period. From the experimenters point of view, this corresponds to the rate of the trigger words.
3. Finally the experiment devices are reading Lift-Off and generating their internal time reference.

[3]Field Programmable Gate Array

This method is generating an incremental and absolute timing system for the experiment data. It is getting fixed to a time-reference using an unique event. Even if the devices didn't start on the same time, due to the event and the phase alignment to the GPS, the time axis can now be compared globally, if the internal timestamp and the timestamp of the unique event becomes part of the telemetry messages. It is to note, that the precise timing information, carried by the trigger words, is not in the contents of the trigger word, but only in its arrival! So the devices have to connect their internal timing as close as possible to the arrival of the trigger word.

2. ABSOLUTE TIMESTAMPS

The previous section described how to obtain relative and absolute timing using trigger words for synchronisation. This is easy to implement for systems on a single payload, containing devices which are able to read these trigger messages. Sometimes sensors doesn't accept synchronisation at all, but delivering absolute timestamps, like devices with embedded GPS receiver. In other scenarios subsystems are located on different places and can't communicate directly. One of those scenarios might be the forwarding of position data to a remote tracking station in order to initiate a handover of control. Another one might be the cooperation of two independent vehicles. If all stations are equipped with their own GPS receivers, it is possible to solve this kind of problem of distributed clocks by extending the method described above.

Since the reference clock is phase-locked to the SP and the frequency is known by design, it is possible to drive a timestamp-counter with this signal. This counter needs adjustments with external information, because the SP alone doesn't provide any absolute time information.

The solution consists of a hardware part, containing the reference clock generator, the timestamp counter and a program to adjust the counter with timing information from of the GPS messages. This hardware is also designed to be part of a FPGA design.

2.1. Hardware

Figure 2 is showing a simplified structure of the hardware for generating timestamps. As for the trigger word generator described above, the GPS receiver is sending its SP

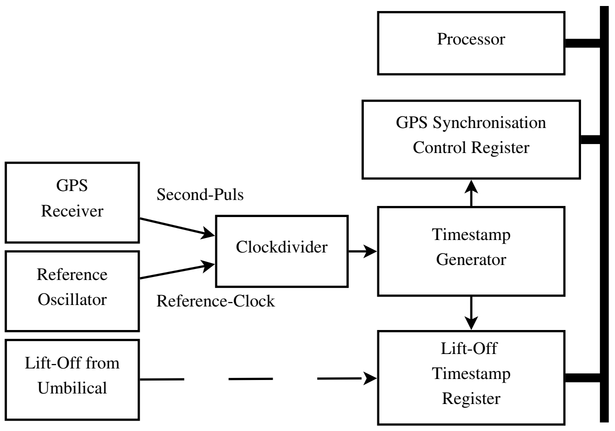


Figure 2. Structure of the GPS-synchronisation hardware. The dashed lines showing external devices.

to the clock divider, running on a local oscillator. The clock divider is then generating the reference clock. It is still possible to generate trigger words from the output of the clock divider. But here the reference clock is used to drive the timestamp counter, which might be designed as a loadable counter.

The processor will access the counter via a control register. This register provides the interface for the software-driver to load the counter and adjust the timestamp. If the design supports further adjustments, e.g. setting message- or bit-rates of the serial output, the register also provides access to these functions.

The Lift-Off event is still an important information for a sounding rocket flight. So another register is showing a snapshot of the timestamp counter, taken when the event occurs. But for the process of generating precise timing, the Lift-Off information is not essential any more, because the timestamp by itself is providing absolute time.

As like above, a serial message can be derived from the reference clock. It contains the current timestamp counter value and maybe flags, similar to the trigger words. The message is starting phase aligned to the reference clock. Like before, the precise timing information is in the arrival time of the message, and the timestamp is always connected to this event.

It makes sense to use a timestamp width sufficient to store absolute values since a well known reference date in the past. If for example a design wants to distribute the milliseconds of the current day, the counter-width must be equal or greater than 27 Bits[4]. In this case the reference clock frequency would be set to 1 kHz.

2.2. Adjusting the Timer

Basically the driver software on the processor has to read the current time from the GPS messages, calculate the value for the hardware timestamp counter and write the result to the counter. In detail, the algorithm consists of six steps (implementation independent version of [3, Page 48]):

1. read the current time from the GPS messages
2. calculate the counter value for the next integer second
3. wait for the SP interrupt
4. load the counter within the service routine
5. wait for the next SP interrupt
6. read and check the timestamp within the service routine

The last step is to check if the hardware is really generating GPS-synchronous timestamps now. It must show the counter value of the previously loaded value plus one second.

The algorithm assumes that the processor has access to the SP as an interrupt, the decoded GPS messages and the frequency of the reference clock is known to calculate the value for a full second. There are some time-critical steps. If the processor is missing one of the interrupts or the service routine needs too long to read or load the counter, the hardware will be set to a wrong value. In case these delays are of a systematically fixed value, the check will fail to detect them. Instead it will show a correct value, because the timer was set late by a corresponding number of ticks.

3. TIMING ERROR SOURCES

Like all real-world systems, the described timing system contains various uncertainties and implementation pitfalls. This last section might serve as hints on how to avoid implementation errors and how to improve the precision of a actual implementation.

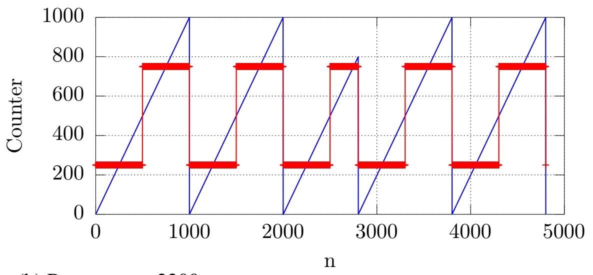
Sampling uncertainty Since trigger messages are sampling the lift-off event or other flags, the uncertainty of these flags timing cannot be less than the update rate of the trigger message (see Fig. 1).

Local Oscillator Frequency Offset Real-world oscillators are influenced by external effects like the temperature, accelerations, etc. These effects leads to an dynamic offset in frequency which is accumulating over the time until the next SP arrives. Figure 3 is showing the result of a simulation of too fast and too slow running oscillators.

To show the effects clearly, a poor oscillator with an accuracy of ±200 ppm was assumed. Depending on the sign

[4] $27Bits = \lceil ld(24h \cdot 3600000ms) \rceil$

(a) Reset at $n = 2800$.



(b) Reset at $n = 2200$.

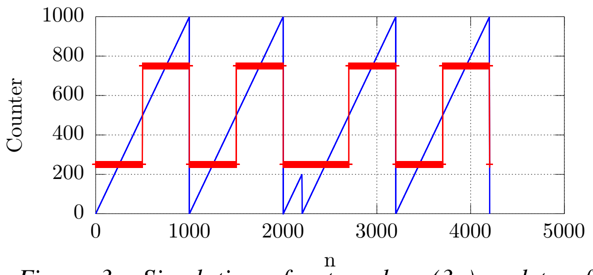


Figure 3. Simulation of a too slow (3a) and too fast (3b) running oscillator. The sawtooth signal shows the counter value, the rectangular wave is the logic level of the resulting, inverted reference clock, which is low below $M/2$ and high above.

of the offset, the width of the low or high period of the resulting clock is changing. The clock divider will work until

$$|err| < \frac{M}{2} \tag{2}$$

If the error increases above this value, e.g. due to missing synchronisation pulses from the GPS over enough time, the divider will generate too less or too much clock cycles. As result, the counter is showing wrong timestamps.

First Pulse of Second When the first SP arrives at the clock divider, the counter will synchronise to the edge very hard. However, at this time the timestamp counter is wrong anyway and need adjustments, as soon as the GPS receiver is tracking satellites. The situation is similar for the trigger word sending. The second in which the SP arrives first, will be wrong in frequency. If this happens after the occurrence of the reference event (Lift-Off), the internal timing of experiments will fail to show the right timestamp.

Jitter in Pulse of Second Considering a perfect local oscillator without any frequency offset and a jittering SP, the edge of the SP will reset the counter as it is designed to do. Except of the statistic nature of the jitter, the result for a single SP-period would be the same as if the oscillator frequency is showing offsets.

Delay due to UART Protocol Assuming a UART transmission with 8 data bits and no parity, a single UART word will need 10 bit periods for full transmission. Obviously there are several options to read the message and to synchronise the internal timing:

- waiting for the edge of the Start-Bit of the UART protocol
- waiting for the "data received"-Interrupt of the receiver UART

The first option doesn't add any latency to the data submission except of the delay introduced by the cable length. But this option requires extra hardware to detect the edge of the Start-Bit. Some microcontrollers are able to do this by using their shared-pin functions.

The second option doen't need anything else than a UART receiver, but add latency (t_{lat}) caused by the transmission of the 10 bits

$$t_{lat} = \frac{n \cdot 10Bits}{f_{Baud}Bits/s} \tag{3}$$

with the baudrate f_{Baud} and the number of words n until the interrupt is triggering the service routine. n would be one in most cases even if the message is longer than one word.

4. CONCLUSION

Within this paper two ways are shown to achieve precise timing, synchronous to the GPS clock. The methods are suitable for synchronizing sensors or other devices able to read trigger words or longer messages with absolute timestamps.

Various error sources were discussed to estimate their influences to the resulting timing signals.

The advantage of this method is the small amount of electrical interfaces to distribute the timing information on a serial transmission line. This makes it easy to read, specially for embedded microcontrollers. It further closes the realtime-loop onboard and finally the experimenters gain flexibility in designing their electronics and software packages.

REFERENCES

[1] DLR. Newsletter countdown. 15, 2011.

[2] Stephan Seidel. *Eine Quelle für die Interferometrie mit Bose-Einstein-Kondensaten für Höhenforschungsraketen*. PhD thesis, Gottfried Wilhelm Leibnitz Universität Hannover, 2014.

[3] Markus Wittkamp. *Telemetrie und Steuerung von Experimenten auf den Nutzlasten von Höhenforschungsraketen*. Master's thesis, FernUniversität in Hagen, 09 2013.