DOI: 10.15439/2019F148
ISSN 2300-5963 ACSIS, Vol. 18

GNSS-based Sound Card Synchronization

Alexander Carôt
Anhalt, University of Applied Sciences
Lohmannstr. 23
06366 Köthen, Germany
Email: alexander.carot@hs-anhalt.de

Hasan Mahmood
Symonics GmbH
Geierweg 25
72144 Dußlingen, Germany
Email: hasan.mahmood@symonics.com

Christian Hoene
Symonics GmbH
Geierweg 25
72144 Dußlingen, Germany
Email: christian.hoene@symonics.com

***Abstract*—Audio communication on the public Internet suffers from not synchronized word clocks of the involved audio devices. The resulting clock drift leads to audio dropouts, which is typically compensated by a sample rate conversion (SRC) in standard telecommunication systems. This, however, does not fulfill the requirements of a high-quality audio system, in which all devices share one and the same word clock. Professional IP based network audio systems such as DANTE or AVB with their respective clock synchronization techniques have so been limited to LAN usage, where network jitter and loss have negligible importance regarding the required accuracy in the dimension of several nanoseconds. In a WAN, however, jitter in the millisecond dimension would lead to unacceptable measurement errors for the intended clock synchronization. As a consequence, we decided to investigate alternative clock synchronization techniques for WAN-distributed devices and developed a GNSS-based approach, which leads to precise clock synchronization.**

I. Introduction and problem

The term "distributed music" or "network music performance" describes a scenario, in which at least two dislocated musicians perform together as if being in the same room. This domain has been investigated for more than two decades [3]. However, with the increasing stability of nowadays available broadband networks another quality reducing factor has become relevant: Despite commonly applied standard audio sample rates of 44,1 kHz, 48 kHz or 96 kHz [10] the word clocks of two different devices do not run in precise synchrony for physical reasons. With respect to audio networking, clock drift means that one audio process is running faster than the remote one. As a consequence, the faster process will not receive a sufficient amount of audio samples, which in turn leads to a buffer underrun and a corresponding audio dropout in specific intervals ranging between 10 and 30 seconds depending on the actual amount of drift. On the other side, the slower process receives too many samples, which eventually leads to a buffer overrun in the same interval, which also corresponds to disturbances in the audio signal. In context with low-latency audio networking the network buffers should be adjusted as low as possible, however, due to the described clock drift problem this proportionally increases the probability for audio dropouts.

In LAN-based sound systems (Local Area Network) such as Dante [2] or AVB [7], each device of the audio network is therefore synchronized to either a dedicated master clock or a specific device on the network, which was previously identified as the clock master. With respect to our Internet-based application, this synchronization process typically cannot be applied due to the existing network jitter in the common dimension of at least one millisecond and rather more. In [4] we presented an approach, which is able to provide WAN-based synchronization (Wide Area Network) by averaging the resulting measurement error, however, the reliability of this approach is limited depending on the actual amount of network jitter.

fast-music is part of the fast-project cluster (fast actuators sensors & transceivers), which is funded by the BMBF (Bundesministerium fur Bildung und Forschung).

II. Concept

In this section, we present a novel concept, which eventually provides reliable sound card synchronization for WAN-distributed devices. Our previous and not perfectly reliable approach uses the WAN itself as the source of synchronization. In contrast, our new concept takes advantage of global navigation satellite systems (GNSS) as the synchronization link between the involved sound devices. We consider GNSS such as the global positioning system (GPS) excellent sources for a grandmaster clock in order to synchronize the word clocks of the involved devices. GNSS transceivers provide a 1-PPS (one pulse per second) output signal, which gives a high pulse at every start of a second [5]. This pulse is our reference point to the absolute start of a second. The UTC (Coordinated Universal Time) time received from GNSS can be used to synchronize the word clocks, but there is a delay between the time received by the GNSS module and time set in the processor, hence it is not accurate up to microsecond level. In order to compensate this offset we take advantage of the time stamping capabilities of IEEE 1588 clocks, which capture the moment of the actual pulse occurrence so that the local time can be set later according to the precise reference. Despite the synchronization with the 1-PPS pulse the local oscillator and the GNSS clock still suffer from the inherent clock drift so that the clocks go out of sync after a couple of seconds. We overcome this problem via a feedback loop and a proportional, derivative and integral (PID) controller that keeps track of the clock drift and adjusts the clock rates respectively. The 1588 clock follows the GNSS clock to a precision and accuracy of greater than one microsecond. Once the 1588 clock is synchronized, we apply the appropriate fine-tuning to an audio clock PLL (phase locked loop) accordingly. Figure 1 illustrates

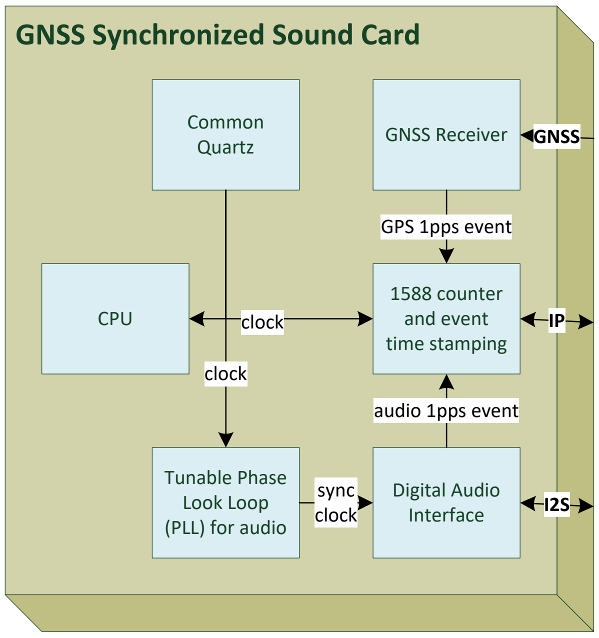

Fig. 1. Block diagram of GNSS synchronized sound card

the described key factors of our theoretical concept. Regarding the upcoming implementation, our concept considers an i.MX7 board by NXP Semiconductors. The i.MX7 series offers a highly integrated processor designed to enable secure and portable applications within the Internet of Things and it suits our demands because it supports the required features with respect to IEEE 1588 and audio word clocking. In that context we will now describe the particular hardware architecture that supports hardware timestamping and time keeping within the Ethernet driver. Afterwards, we will explain how the clocking is realized and how the audio word clock can be controlled and fine-tuned.

A. Hardware architecture

To allow for IEEE 1588, the MAC hardware in i.MX7 by NXP Semiconductors is combined with a time-stamping module to support precise time-stamping of incoming and outgoing frames and granule control of the IEEE 1588 time [11]. Figure 1 shows the block diagram of the i.MX7 board architecture.

At the centre of the time stamping module is a 32-bit counter register, which keeps track of IEEE 1588 time on the hardware level. It is incremented after every rising edge received from the oscillator by an amount specified by $ENET\_ATINC[INC]$ register. In our use case, this value is set to 10 nanoseconds because it corresponds to the Ethernet clock with a frequency of 100 MHz. The $ENET\_ATPER$ register contains the number of nanoseconds after which the counter will wrap around. It is programmed with a value of 10^9 so that the counter resets itself in intervals of one second [11].

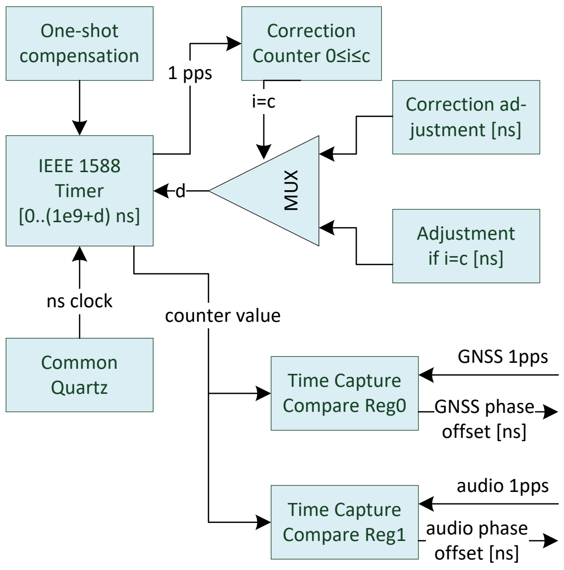

Fig. 2. Block diagram of the adjustable IEEE 1588 counter and the event time stamping

The block diagram of the adjustable timer is illustrated in figure 2.

The $ENET\_ATCOR$ register is designed for the fine grain tuning of the counter. It defines after how many clock cycles the correction counter should be applied. The amount of correction is specified in $ENET\_ATINC[INC\_CORR]$. If the value of $ENET\_ATINC[INC\_CORR]$ is greater than $ENET\_ATINC[INC]$ the counter speeds up, if it is less the counter slows down. Furthermore, the system supports a one-shot offset event generation. In that context, the module contains the $ENET\_ATOFF$ register. It holds the final nanosecond value after which the counter is reset for one single time.

B. Audio clocking architecture

The audio word clock in the i.MX7 is derived from the board's main 24 MHz phase-locked loop (PLL) oscillator. In that context, our requirements determine that the audio word clock's frequency must reside in a significantly lower dimension of 44,1 kHz, 48 kHz or 96 kHz and therefore represents a divisor of the board's main frequency of 24 MHz. Furthermore, the final frequency must be adjustable with respect to our previously described synchronization approach. Regarding the fine-tuning functionality the board contains three registers: In the following the $CCM\_ANALOG\_PLL\_AUDIO[DIV\_SELECT]$ register will be abbreviated with Div, the $CCM\_ANALOG\_PLL\_AUDIO\_NUM$ register will be abbreviated with Num and the $CCM\_ANALOG\_PLL\_AUDIO\_DENOM$ register will be abbreviated with $Denom$. Based on these registers the output of the audio PLL depends on the following calculation:

$$F_{output} = F_{osc} * (Div + Num/Denom)$$

Div multiplies the 24 MHz base clock frequency with the integer specified. The actual granule fine tuning is provided by adding a 32-bit fraction denoted by Num and $Denom$. The resulting audio PLL goes through a pre-divider and a post-divider to get the respective clock frequencies. Pre and post dividers have 64 steps so the frequency change is always an integer multiple of the audio PLL frequency:

$$F_{peripheral} = F_{src} * (Div_{post}/Div_{pre})$$

Afterwards, a so-called Sound Asynchronous Interface root clock (SAI) is derived from the audio PLL, which represents the master for the final audio word clock. The SAI root clock feeds a bit clock generator, which eventually generates the final square-waves-based audio word clock and eventually determines the sample capture and playback of the sound card.

III. Implementation

This chapter describes the actual implementation of our concept. First, we describe how the synchronization of the local time to UTC time is being realized with an accuracy of one second. Secondly, granule control of the time with nanosecond accuracy is explained and how to keep the clock synchronized with the GNSS time using a PID controller. Eventually, the final audio clock is synchronized with the IEEE 1588 clock, which in turn is in sync with the GNSS clock as intended.

A. IEEE 1588 synchronization to UTC time

In order to let the IEEE 1588 clock follow UTC time, first, the local clock must be set to UTC time. This time can be received from the GNSS module periodically once every second [1]. A daemon called *gpsd* is used as an interface between the i.MX7 and the GNSS module. *Gpsd* provides a socket connection between the module and the host [6]. The IEEE 1588 clock ID can be retrieved with the function *phc open()* [9]. The definition of this procedure is found in the library *phc2sys* [9]. After the retrieval of the clock ID the POSIX function *clock settime (clockid t clockid, const struct timespec *tp)* [8] is called and the current time is read from the gpsd buffer. Hence, the IEEE 1588 time is set to UTC time at the nearest second.

B. Granule control of the 1588 clock

Once the 1588 clock time is accurately set at the nearest second, an offset between the start of a second in the UTC time and the local clock time can be observed as described in the concept. The required offset compensation is realized via hardware time stamping. As soon as the 1-PPS pulse occurs on the interrupt line, the current value of the 1588 counter is latched on to the Timer Capture Compare Register $ENET\_TCCR$ by the hardware to be inspected later on by the software [11]. This enables a precise calculation of the phase offset. Since the counter is reset in intervals of one second, this value represents the true second offset between the local clock and the absolute start of the second. A one-shot event through $ENET\_ATOFF$ is then applied in the interrupt handler, whose value is set to the value latched in $ENET\_TCCR$, which enables the timer. When the 1588 counter reaches this value, it is set to zero and starts again resulting in the desired offset removal.

After the offset compensation has been applied it is possible to take care of the actual clock drift. The drift can be controlled by an adjustment value that is applied to the counter every second. Ideally, every second would consist of 10^9 ns. However, as the clocks are drifting, the GNSS second is not exactly equal to 10^9 ns of the i.MX7 quartz. Instead, we see a difference to be equalized. To compensate for this clock drift, the IEEE 1588 timer counts to $10^9 + d_{normal}$, where d_{normal} is the drift offset. This principle compensates the clock drift with a certain degree of precision, however, we further optimize it by using d_c as a correction value instead of the d_{norma} value once every c seconds. This approach is called Proportional-Integral-Derivative (PID).

More precisely, the $ENET\_ATINC[INC\_CORR]$ register and the $ENET\_ATCORR$ register, which allow granule control over the 1588 counter, can be sped up or slowed down using the Timer Increment. The $ENET\_ATINC[INC\_CORR]$ register has the new increment and the $ENET\_ATCORR$ register defines its frequency. When the number of clock cycles of the 1588 counter equals the $ENET\_ATCORR$ value, the 1588 counter is incremented by the $ENET\_ATINC[INC\_CORR]$ nanosecond value instead of the usual $ENET\_ATINC[INC]$ value.

The resulting adjustment assumes a difference in time between our 1588 clock start of second and the GNSS start of second. The difference is obtained from the current value latched on to the $ENET\_TCCR$ register. Speeding up the 1588 counter is realized by increasing the $ENET\_ATCORR$ register value. Decreasing it slows down the timer.

C. Synchronizing the audio clock with the IEEE 1588 clock

The final step of synchronizing the audio PLL with the IEEE 1588 clock is realized via a task, which is scheduled in intervals of one second. It receives the clock drift correction from the previous step. According to this value, the audio PLL frequency is changed respectively in order to reflect the change of the audio word clock speed. However, the clock drift compensation of the audio clock is different compared with IEEE 1588: The audio PLL allows to tune the audio bit clock with a very fine grain resolution of less than 0.1 Hz.

Even if the clock drift of the audio signal is compensated, we will need to determine the precise time phase offset of the audio signal with respect to the GNSS clock. Otherwise, the incoming and outgoing audio signals would not be in sync. In order to achieve this, we introduce a special synchronization mode. This synchronization mode is activated only at the startup period because we assume that the time offset does not change if the clocks are running synchronously. Instead of a digital audio output signal, an artificial 1-PPS signal is generated.

In the synchronization mode, the serial audio bit output of the SAI is connected with the serial audio input to a direct

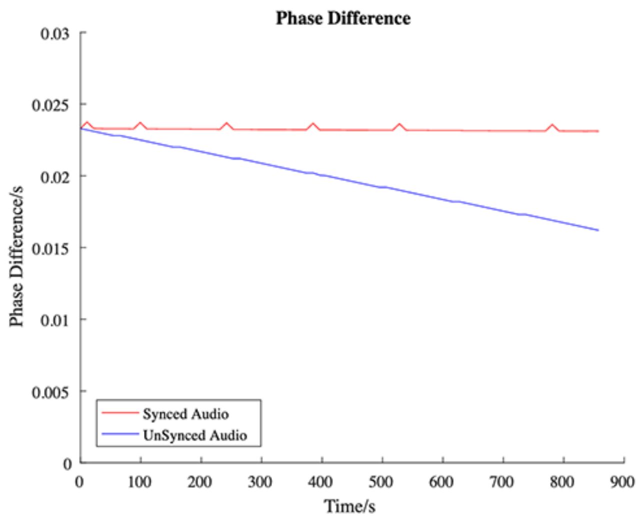

Fig. 3. Audio word clock drift with and without our synchronization

feedback loop. In addition, the SAI data output – filtered by a D flip-flop that is driven by the audio bit clock – is connected to an IEEE 1588 time capture register (similar to the 1-PPS signal of the GNSS receiver). Then, we can measure the delay between sound and GNSS if playing out an artificial digital audio pattern.

IV. Evaluation

The clear and obvious purpose of this paper is the removal of clock drift in remotely distributed sound systems. Therefore we decided to perform our evaluation based on the comparison of two sound card's word clocks with regard to the phase of a low-frequency square wave, which is predestined in that context. In our setup, we generate a 20 Hz square signal with a frequency generator and feed it into two sound cards. The outputs are fed into a 2-channel oscilloscope, which displays both signals at the same time. If the sound card's clocks exhibit a drift the phase of the displayed square waves will drift as well over time. If our implementation is successful we expect the signal to stay in phase and the image on the oscilloscope should stay the same. The duration of the measurement was set to 15 minutes and we retrieved a phase measurement in intervals of 11 seconds. In figure 3 we present our results of this evaluation with and without our developed synchronization approach.

It is clear that without synchronization the phase drift increases proportionally and amounts to 7.5 ms after the final duration of 15 minutes. With our synchronization being applied the phase remains the same because the audio word clocks don't exhibit a drift anymore. In fact we can observe slight phase variations in fixed intervals, however, they are immediately compensated by our applied algorithm and cannot be considered problematic regarding the demands of our described use case.

V. Conclusions and Future Work

In this paper, we describe the successful implementation of a GNSS-based sound card synchronization technique for devices distributed in wide-are-networks (WAN) such as the public Internet. Measurement results clearly show the inherent audio word clock drift of approximately 7.5 ms over a duration of 15 minutes without our synchronization technique and precise synchrony of both sound cards and in turn no clock drift when applied. To our knowledge this is the first implementation able to provide the described functionality in the sound card domain. In the near future, we will develop a custom sound card, which benefits from this approach and integrates it into our remote music system. The drawback, however, is that GNSS is not necessarily available in any given environment – especially in basement rehearsal chambers – which is why we will also apply our older approach side-by-side with the new.

References

[1] L.J. Arceo-Miquel, Yuriy Shmaliy, and Oscar Ibarra-Manzano. "Optimal Synchronization of Local Clocks by GPS 1 PPS Signals Using Predictive FIR Filters". In: *IEEE Transactions* (2009), pp. 1833–1840. DOI: 10.1109/TIM.2009.2013654.

[2] Audinate Website. *Dante Overview*. [Online; accessed 12-May-2019]. 2019. URL: https://www.audinate.com/solutions/dante-overview.

[3] Alexander Carôt. "Musical Telepresence – A Comprehensive Analysis Towards New Cognitive and Technical Approaches". PhD thesis. Institute of Telematics – University of Lübeck, Germany, 2009.

[4] Alexander Carôt and Christian Werner. "External latency-optimized soundcard synchronization for applications in wide-area networks". In: *Proceedings of the 14th regional AES Convention*. Tokyo, Japan, July 2009.

[5] Bálint Ferencz. *Hardware Assisted IEEE 1588 Clock Synchronization Under Linux*. Master Thesis. Budapest University of Technology and Economics, 2013.

[6] *GPSd reference manual*. [Online; accessed 12-May-2019]. URL: http://catb.org/gpsd.

[7] Christoph Kuhr and Alexander Carôt. "A Jack Sound Server Backend to Synchronize to An IEEE 1722 AVTP Media Clock Stream". In: *Proceedings of the Linux Audio Conference 2019*. Stanford, USA, 2019.

[8] Donald A. Lewine. *POSIX programmers guide*. first. O'Reilly, 1994.

[9] *phc.h Source code*. [Online; accessed 12-May-2019]. URL: https://github.com/richardcochran/linuxptp/blob/master/phc.h.

[10] Ken C. Pohlmann. *Principles of Digital Audio*. fifth. The Mcgraw-Hill Companies, 2005.

[11] NXP Semiconductors. *iMx7d Dual Applications Processor Reference Manual*. 2019.