Microprocessors and Microsystems 000 (2016) 1–8

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A clock synchronization method for EtherCAT master

Xin Chen, Di Li, Jiafu Wan*, Nan Zhou

School of Mechanical & Automotive Engineering, South China University of Technology, Guangzhou, China

ARTICLE INFO

Article history:
Received 28 October 2015
Revised 16 January 2016
Accepted 3 March 2016
Available online xxx

Keywords:
EtherCAT
Clock synchronization
Packet loss

ABSTRACT

EtherCAT has been widely applied in the motion control domain due to its advantages of the fast response speed, low CPU usage and good synchronization performance. Although the built-in distributed clock (DC) synchronization mechanism exhibits strong performance between slaves, the method of clock synchronization between the master and the reference clock is left open for selection by users and there are very few studies in this area. This paper introduces three synchronization modes for EtherCAT and analyzes the reasons for packet loss in free run mode and DC mode. This paper presents a novel method to realize the master clock synchronization with the reference clock. The proposed method can eliminate the settling time which is unusual in other synchronization mechanisms. The method is implemented on Windows with real-time extension and experimental results prove its feasibility.

1. Introduction

Real-Time Ethernet (RTE) represented by the EtherCAT protocol enables an advanced control architecture [1]. Previously, motion control systems have used field bus technology to build their motion control network, such as DeviceNet, Profibus, CC-Link, CANopen, Modbus and so on. Due to the speed and bandwidth limitations of the protocol, the feedback control loops (such as the position loop and the speed loop) in the motion control system must be contained locally within the execution [18]. The appearance of EtherCAT and other RTE protocols break this structure. However, regardless of the type of communication protocol used by the motion control system, its pursuit is the same: high speed, high precision, reliable stability and good multi-axis coordination ability. Motion control systems can be divided into three research areas: control systems, communication networks and informatics, and real-time operation systems [2,15]. Rather than the Performance Indicators (PIs) defined in IEC61784 [16,17], motion control systems are more focused on the overall performance of the communication protocols, such as the Minimum Cycle Time (MCT) which is related to speed and precision, the packet loss rate which is related to stability, the clock synchronization accuracy which is related to the multi-axis coordination ability and so on.

In a distributed control system, clock synchronization is required to guarantee coordination between the controller and the executors. There are several clock synchronization methods, such as Network Time Protocol (NTP) and Simple NTP (SNTP) for the Internet, Time-Trigged Protocol (TTP) for aerospace, and Serial Real Time Communication Specification (SERCOS) and Precision Time Synchronization Protocol (PTP, IEEE1588) for industry. Due to the high accuracy of the IEEE1588 protocol, this protocol and its derivative methods are widely adopted in RTE. The Precision Transparent Clock Protocol (PTCP) used in PROFINET, CIP-sync used in Ethernet/IP and the modified PTP used in EPA are variants of IEEE1588. At present, PTP studies are mainly focused on its fault tolerance [3], interoperation [4], implementation, and modifications to its algorithm. There are a number of different methods to decrease uncertain delay and noise for getting clock skew, such as Kalman Filter, optimization based Linear Programming (LP), an averaging method, and a Linear Regression (LR). EtherCAT has employed a simpler clock synchronization mechanism, which is known as the Distributed Clock (DC). Its clock accuracy are well below 1 μs [5] and the clock jitter between the slave and reference clock can be as low as 12 ns [6]. However, this mechanism is not ideal for clock synchronization between the master and the slave [7]. If clock synchronization between the master and the slave is not taken into account, the system will lose packets periodically, which affects the overall stability of the motion control system. Additionally, a lower clock synchronization precision will limit the number of cycles of periodic data and the impact speed and precision of the motion control system. Therefore, researchers have proposed several methods for clock synchronization between the EtherCAT master and reference clock, such as the adoption of a PI controller for the master clock that is adjusted according to the reference clock. However, these methods have several drawbacks for an EtherCAT master that is based on a PC. Firstly, a frequent thread switch may reduce the clock accuracy of the master.

* Corresponding author: Tel.: +8618814124373.
E-mail addresses: chentonyxin@163.com (X. Chen), itdili@scut.edu.cn (D. Li), mejwan@scut.edu.cn, jiafuwan_76@163.com (J. Wan), nanchow@qq.com (N. Zhou).

http://dx.doi.org/10.1016/j.micpro.2016.03.002
0141-9331/

Secondly, some operating systems do not allow clock speed modification. Ganz has abandoned DC and modified EtherCAT by integrating the PTCP protocol. However, this method relies on the application layer protocol of EtherCAT, which is Ethernet over EtherCAT (EoE). Therefore, it belongs to the application level of the EtherCAT clock synchronization [8]. This method can increase the data load and affect the MCT of the system. Moreover, it has even lower accuracy than DC [9].

When an EtherCAT system is switched to its operation state, cyclic data can be transmitted. At this moment, to avoid packet loss, the master clock should be running at a stable speed which is equal to the reference clock. Neither a PI controller nor the DC mechanism can achieve a predictable settling time, as even the DC mechanism synchronizes the slave clock to the reference clock for a number of periods before the system switches to its operation state. In order to avoid packet loss in the operation state and stabilize the master clock to be in line with the reference clock, this paper introduces a fast clock synchronization method between the master clock and the reference clock by tracing the speed of the reference clock.

The EtherCAT protocol defines three synchronization modes of the slave: free running, synchronous with an SM event, or synchronous with a DC SYNC event [10]. These modes exhibit different packet loss rates and synchronization performance. The remainder of this paper is structured as follows: in Section 2, the EtherCAT protocol and three synchronous modes in the slave are briefly discussed; Section 3 analyses the reasons for packet loss and issues experienced by the DC mechanism and PI controller in meeting requirements; Section 4 describes a clock synchronization method for the EtherCAT master; Section 5 introduces the implementation of this method on Windows with RTX, a real-time extension; Section 6 describes the performance of this method.

2. EtherCAT protocol basics

2.1. EtherCAT protocol

EtherCAT was developed by Beckhoff in Germany and is supported by the EtherCAT Technology Group (ETG), which has over 1300 members. It is listed by the Semiconductor Equipment and Materials International (SEMI) [11], the International Electrotechnical Commission (IEC) and the International Organization for Standardization (ISO) [12]. There are four states in the EtherCAT system: initialization (INIT), pre-operation (Pre-OP), safe operation (SAFE-OP), and operation (OP). Information is exchanged with slaves in the OP state. The communication mode of EtherCAT is master/slave, and there is only one master within the network. EtherCAT supports many types of topologies, such as line, star, tree and so on. However, regardless of the network topology, EtherCAT networks are based on a physical ring topology. As shown in Fig. 1 all frames are generated by the master, and return to the master after passing through all the slaves. Each slave processes and propagates each frame onward in a very short time. Without processing data, the timestamp of the reference clock is sent to the master and slaves by a special EtherCAT command known as read multiple write (either ARMW or FRMW).

Similar to the communication model of industrial Ethernet, the model of EtherCAT consists of three parts: a physical layer, a data link layer and an application layer. A special device acts as an interface between the EtherCAT fieldbus and the slave application, called an EtherCAT Slave Controller (ESC). ESC has a DPRAM to store the data exchanged between the master and slaves and uses SyncManager (SM) to guarantee consistency and security of data exchange between the master and slaves. There are two communication modes: buffered mode and mailbox mode. The mailbox mode implements a handshake mechanism for the data exchange,

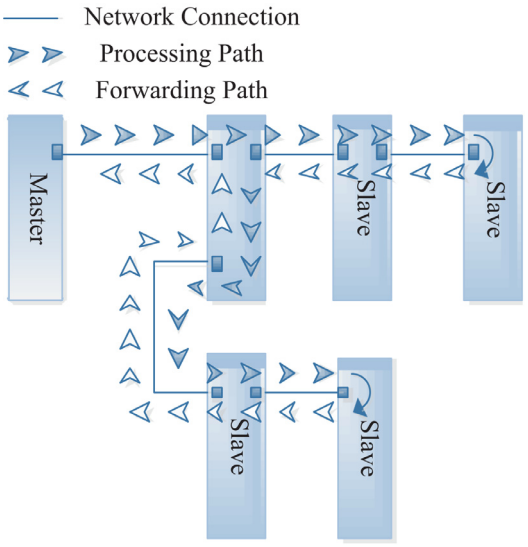


Fig. 1. Data transmission path of EtherCAT.

to ensure that no data is dropped. The buffered mode allows both sides, the EtherCAT master and local application, to access the communication buffer at any time. Generally, the buffered mode is employed to process data exchange.

2.2. Three synchronization modes of EtherCAT

The EtherCAT protocol defines three synchronization modes: free running (FR), synchronous with a SM event (SM) and synchronous with a SYNC event of local clock (DC). The difference between the three synchronization modes is the trigger mode of the local application event [10], as shown in Fig. 2. The free running mode is shown in Fig. 2A, where the trigger event is generated by the local application layer. The SM mode is shown in Fig. 2B where the trigger event is an IRQ generated by the ESC. When the data sent by the master reaches the DPRAM of each slave, the ESC triggers an IRQ signal to the local application. The DC mode is shown in Fig. 2C. The local application events are triggered by three signals: IRQ, SYNC0 and SYNC1. IRQ triggers the event where the local application layer copies the data from the ESC, SYNC0 triggers the event where the local application layer enables the output, and SYNC1 triggers the event where the local application layer enables the input and copies the data to the ESC. When SYNC1 is omitted, the input and copy events will be executed after a determined time which can be set by user. A comparison of the three modes is listed in Table 1.

3. Packet loss analysis and clock synchronization methods

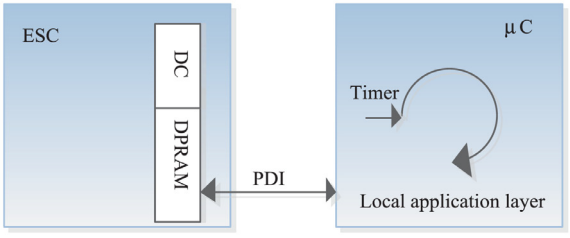
3.1. Reasons for packet loss

In SM mode, the output and input of the slave are controlled by the master. When the slave application layer receives the data, it is implemented immediately. Therefore, there is no packet loss in SM mode. However, due to propagation and execution delays, the local application of different slaves can have different execution times that will lead to lower coordination between different slaves. For free running and DC mode, the time taken for the data to reach the slave ESC is determined by the master and the time for data transfer from the ESC to the local application is determined by the slave timer, either in the application layer or in the ESC. Regardless of which synchronization mode is employed, the slave timer is

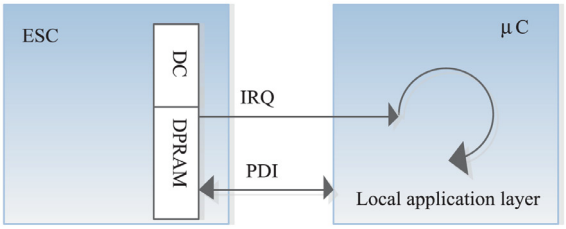
Please cite this article as: X. Chen et al., A clock synchronization method for EtherCAT master, Microprocessors and Microsystems (2016), http://dx.doi.org/10.1016/j.micpro.2016.03.002

Table 1
Three synchronization modes.

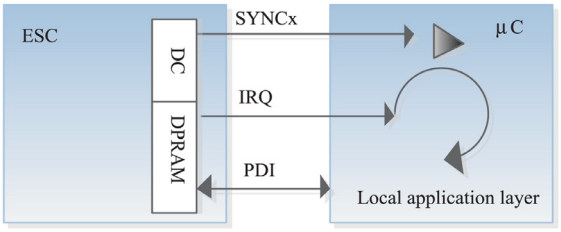
Mode	Trigger mode	Enable output and input	Determinacy of cycle time
FR	Application timer	Application timer	Yes
SM	External	master	No
DC	External	DC	Yes



A. Free Running mode

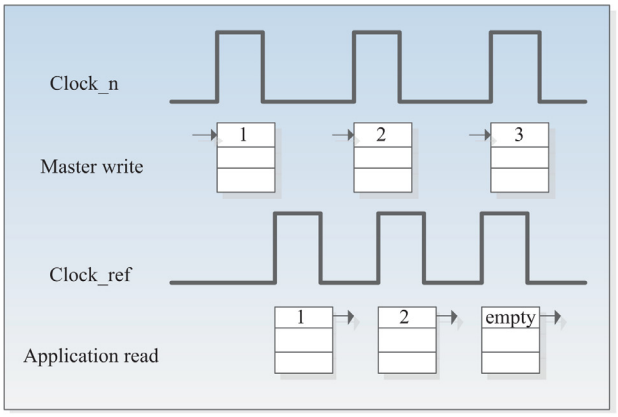


B. SM mode

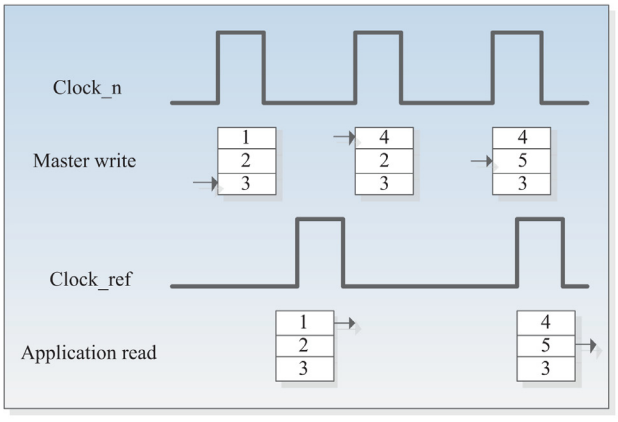


C. DC mode

Fig. 2. Block diagrams of three synchronization modes A. Free running mode B. SM mode C. DC mode.



A.Master clock running lower than the slave clock



B. Master clock running faster than the slave clock

Fig. 3. Packet loss block diagrams A. Master clock running lower than the slave clock B. Master clock running faster than the slave clock.

different from the master timer. The clock synchronization accuracy is widely known to influence the coordination between slaves. Asynchronization between the master and slave can lead to packet loss, as shown in Fig. 3. When the speed of the master clock is slower than the slave clock, the local application receives empty packets for a period of time. Although no packets are lost, the local application outputs nothing during this cycle. However, when the speed of the master clock is faster than the slave clock, data which has not yet been read by the local application is overwritten by other data sent by the master, resulting in data loss.

When the speed of the master clock is different from the slave clock, the problem of packet loss cannot be solved by changing the buffer size. The fundamental reason for the loss is clock asynchronization between the master clock and the slave clock. However, due to the propagation delay and clock jitter, the time when the frame reaches the slave is not deterministic. To avoid packet loss due to delays and jitter, the time interval between the ESC receiving the frame and triggering SYNC signal is required, as shown in Fig. 4.

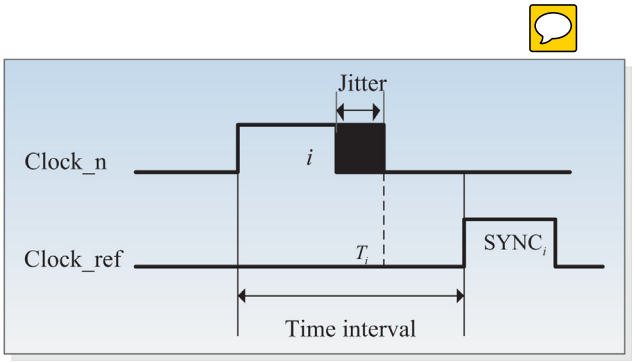


Fig. 4. Time interval block diagram.

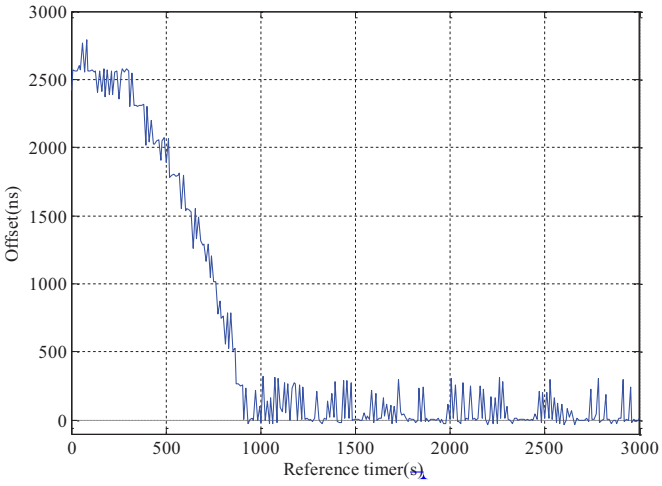


Fig. 5. Process of local clock synchronization with reference clock.

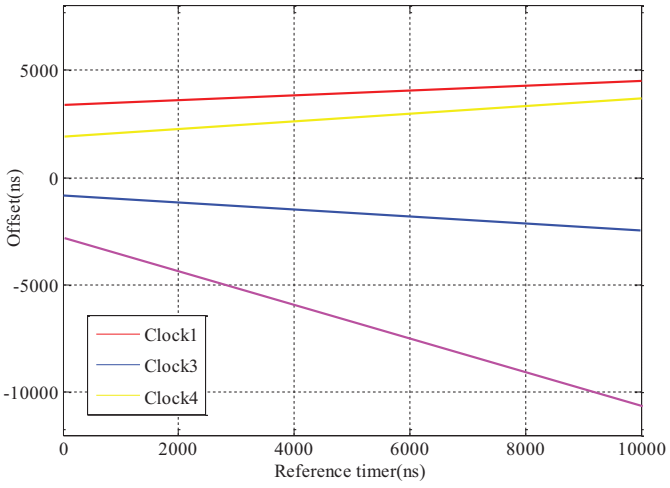


Fig. 6. Offset of local clocks with reference clock.

3.2. Clock synchronization methods

The DC mechanism consists of three phases: propagation delay measurement, offset compensation and drift compensation. Propagation delay measurement and offset compensation can be done during the Pre-OP state. Drift compensation occurs in every cycle during the data processing and uses special EtherCAT commands known as read multiple write (either ARMW or FRMW). These commands are sent periodically by the master to distribute the system time of the reference clock to all the other slaves. Each slave compares the received value (t_{ref_time}) to its system time (t_{local_time}), and determines a value Δt which is used to tune the speed of its local clock.

$$\Delta t = (t_{local\_time} + t_{offset} - t_{prop\_delay}) - t_{ref\_time} \quad (1)$$

Each clock's value is increased by 10 ns when the average value of Δt is close to 0, whereas 11 ns or 9 ns are added when it is detected that the local time is slower or faster than the reference clock, respectively. As shown in Fig. 5, the DC mechanism can effectively synchronize the slave clock to the reference clock due to strong compensation for the propagation delay and offset and high stability and accuracy of the local clock. However, when the offset is 2.5 ms, this method needs more than 1 s to reach a steady state. Meanwhile, this mechanism need change the speed of the clock directly. Therefore, the DC mechanism does not apply in the EtherCAT master implementing on PC.

The PI controller has the same problem as DC [13]. The PI controller obtains the time deviation between the two clocks through a proportional amplifier and an error accumulator and uses this to adjust the time of the clock. However, it also has several requirements. Firstly, it is difficult to calculate the controller parameters. Secondly, it is difficult to determine the settling time and overshoot with certainty. Thirdly, there is a continuous requirement for the clock time to be synchronized.

4. A clock synchronization method

4.1. Clock model

A timer consists of a crystal oscillator and a counter. Since it is not possible to obtain crystal oscillators with exactly the same properties each time, different timers have different levels of drift [11]. In other words, different timers will have different speeds. There are two reasons for deviation between different clocks: different start times and different speeds [5]. Fig. 6 shows four clocks with different start times and clock skews which change linearly with the reference clock.

For an absolute clock, the current time $T(t)$ is as shown in formula (2)

$$T(t) = T_{offset} + F(t)t \quad (2)$$

T_{offset} is the offset between the start time of the clock and the absolute clock. $F(t)$ is the speed of the clock, which is directly proportionate to the frequency of its crystal. When the speeds of the reference clock and the local clock are the same, only the initial clock error exists. When both the clocks have different speeds, the deviation between them will increase with time. Generally, the speed of the clock can be seen as a constant. However, in a real world application, the clock speed will be modified by the environmental temperature and the age of the crystal oscillator.

4.2. Clock synchronization method

Different clock speeds are the primary reason for an increase in time deviation between two clocks. When two clocks have equal speeds, their time deviation is constant. In ideal conditions, the clock speed can be considered a constant. Therefore, the speed of the master clock is proportional to the speed of the reference clock. The proposed method uses linear curve fitting by collecting the reference clock data periodically to calculate their specific value. Based on the specific value, the master clock adjusts its clock speed by changing the cycle time. The process is completed in one clock synchronization cycle (a clock synchronization cycle is different from the system cycle time, but can be set to be equal). As shown in Fig. 7, $Clock_{ref}$ is the time of the reference clock and $Clock_n$ is the time of the master clock. Just before time t_1, the master calculates the specific difference between the master clock and the reference clock. After time t_1, the master adjusts its clock speed to be equal to the reference clock speed. Meanwhile, the offset ($T_{offset\_r\_n}$) between the master clock and the reference clock will be compensated at time t_1.

In the mechanism of the EtherCAT, the clock offset of the slaves with the reference clock is calculated in Pre-OP. The specific value also can be obtained in the Pre-OP state. As shown in Fig. 8, the clock should collect the timestamps and adjust the clock speed according to the current specific value in OP state due to the measurement error and the drift of the reference clock.

Please cite this article as: X. Chen et al., A clock synchronization method for EtherCAT master, Microprocessors and Microsystems (2016), http://dx.doi.org/10.1016/j.micpro.2016.03.002

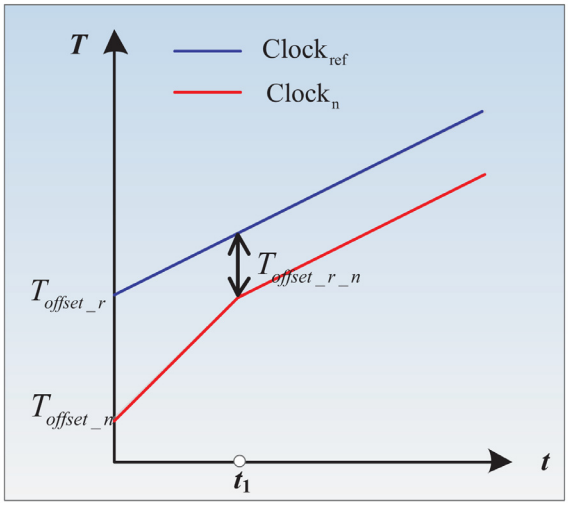


Fig. 7. Clock synchronization method block diagram.

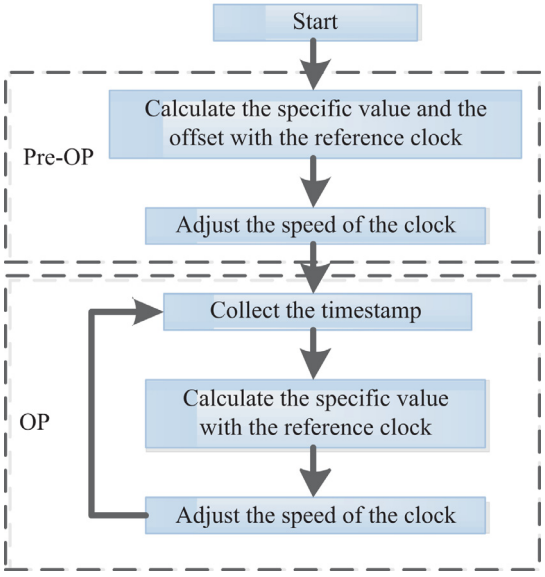


Fig. 8. The flowchart of the clock synchronization method based on EtherCAT.

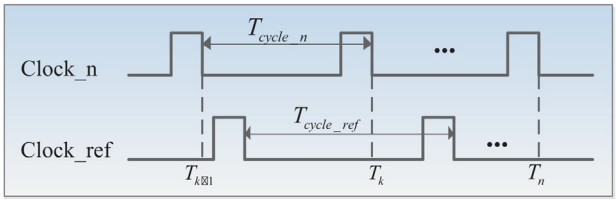


Fig. 9. Relationship between the timestamp and cycle time of the master and reference clock block diagram.

4.3. Speed relationship between the master and reference clock

The reference clock time can be obtained every period during the frame for processing data, as shown in Fig. 9. Clock_n is the time of the master timer, while Clock_ref is the time of the reference clock.

The master reads the reference clock time ($T_{k-1}, T_k, \ldots, T_n$) each cycle that is generated by the timer of the master. The difference ΔT_k between T_k and T_{k+1} is measured for the interval T_{cycle_n} according to the master clock, which is not equal to the cycle time (T_{cycle_ref}) of the reference clock. Nevertheless, it is related to the speed of the master and reference clock as shown in formula (3).

$$\frac{F_r}{F_n} = \frac{T_k - T_{k-1}}{T_{cycle\_n}} \tag{3}$$

The ideal value of the master cycle time is not T_{cycle_n} but T_{cycle_ref}. In other words, the value of ΔT_k obtained by the master strives to be equal to T_{cycle_ref}. The master clock should adjust the cycle time to be equal to T_{cycle_ref}, as shown in formula (4). $T_{cycle\_n\_ral}$ is the value that is set after adjustment:

$$\frac{T_k - T_{k-1}}{T_{cycle\_n}} = \frac{T_{cycle\_ref}}{T_{cycle\_n\_ral}} \tag{4}$$

Since, ΔT_k is influenced by the indeterminate nature of the propagation delay and the master clock jitter, its value has some disturbance. Therefore, a more accurate value can be obtained by calculating an average value of ΔT_k. At this point, the period of the clock synchronization is different from that of the cycle time.

In a real environment, the speed of the clock is influenced by the environmental temperature and the age of the crystal. Although the speed of the clock is not a constant, it can be considered to be a constant within a certain time. To obtain effective clock synchronization and guarantee that no packets will be dropped, the master needs to adjust the master clock speed by following any changes in the reference clock speed, as in formula (5). T_{ref_test} is the average value of ΔT_k during N periods of the master. The synchronizing cycle between the master and the reference clock approximate to $N^* T_{cycle\_ref}$:

$$T_{cycle\_m}(k+1) = \frac{T_{cycle\_ref}}{T_{ref\_test}} T_{cycle\_m}(k) \tag{5}$$

5. Implementation in RTX system

RTX is offered by the company Intervalzero to solve real-time problems on the Windows platform, and is a hardware real-time extension system [14]. The hardware abstraction layer of RTX provides three high precision clocks: CLOCK1 (also called CLOCK_SYSTEM), CLOCK2 (CLOCK_FASTEST) and CLOCK3 [15]. The clock accuracy of CLOCK2 can reach 1 μs, and it is typically used as the timing reference clock in actual applications. For a clock synchronization process, clock offset compensation is accomplished by adjusting the counter value to control the clock speed. However, in an RTX system, its clock speed cannot be amended. This makes it difficult to implement the synchronized method using an EtherCAT distributed clock to implement the EtherCAT master. Although the clock speed of an RTX system cannot be adjusted, the clock period can be changed. For a master clock synchronized with the reference clock to solve packet loss, the clock speed of the RTX system can be changed by adjusting the clock period using the method described in Section 3.

5.1. Task scheduling with a deterministic time

In a control system, the periodic data (process data) carries the information required by each node, and it often has a higher demand for a deterministic time than the communication system is capable of, especially when it contains the timestamp. Orderly operation of the whole system can be guaranteed by task scheduling using a determined time. In the EtherCAT protocol, the data is divided into cycle data and non-cycle data according to the slave requirements. Cycle data processes periodically with the highest priority, while non-cycle data processes when the periodic data processing is idle. The cycle data tasks can be divided into four subtasks: the task of sending cycle data (Task 1), unpacking the

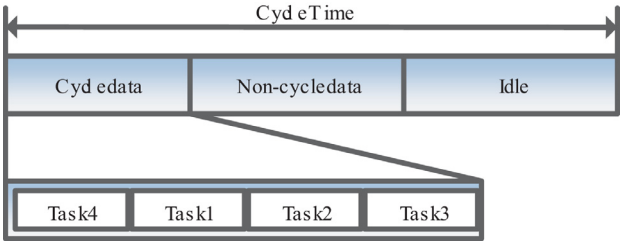


Fig. 10. EtherCAT scheduling block diagram.

Algorithm 1 Truncation error compensation

Input: The ideal cycle time $T_{cycle_m}(k+1)$of the $k+1$ cycle
Output: The actual cycle time of the $k+1$ cycle
1: Round $T_{cycle\_m}(k+1)$ to *Reslution*
2: Accumulate truncation *Error*
3: **if** *Error* > *Reslution*
4: Let $T_{cycle\_m\_round}(k+1)$ be equal to $T_{cycle\_m\_round}(k+1)$+ *Reslution*
5: Let *Error* be equal to *Error*- *Reslution*
6: Let $T_{cycle\_m}(k+1)$ be equal to $T_{cycle\_m\_round}(k+1)$
7: return $T_{cycle\_m}(k+1)$

received cycle data (Task 2), clock synchronization with the reference clock (Task 3) and setting the next cycle time (Task 4). Their scheduling is shown in Fig. 10. The four tasks have different requirements for a deterministic time. Since, Task 4 determines the cycle time, Task 4 has the highest requirement for a deterministic time. Task 1 also needs higher time determination since it reads the current time of the reference clock. Task 2 and Task 3 have lower requirements for a deterministic time since they finish the task within the current cycle.

5.2. Truncation error compensation

The clock precision of a RTX real-time kernel can reach 1 μs (CLOCK2), but its clock resolution is not as high as its precision. When the resolution is less than *Reslution*, system performance may be influenced. In order to guarantee system performance, this paper has chosen a minimum resolution of 10 μs. Therefore, truncation errors exist for the cycle time value due to the aliquant. The process of truncation error compensation is shown in Algorithm 1. $T_{cycle\_m\_round}(k+1)$ is the integer value of $T_{cycle\_m}(k+1)$.

6. Experimental setup and performance analysis

6.1. Experimental setup

The timestamp, which is the current time of the reference clock, can be obtained by the returning frame of every cycle. The value of the timestamp reflects the performance of clock synchronization between the master and the reference clock. As shown in Fig. 4, the cycle time of the system is T_p. After compensation for the propagation delay and initial offset, the current virtual time of the master $T_m(i) = T_s + T_p{}^*i$, T_s is the time that the EtherCAT system enters its operation state. Deviation between the master current time $T_m(i)$ and the reference clock T_i occurs due to jitter of the master clock. Therefore, the value of the timestamp can indicate whether or not the system has packet loss. The time interval of SYNC is set to be half the value of the cycle time.

This paper uses a packet-capturing method to obtain the periodic timestamp. The experimental platform is shown in Fig. 11. The frames which include the timestamp of the reference clock are captured by NANL-C500-RE of Hilsher. Since the clock synchronization between the slave and the reference clock is performed using the DC mechanism and has strong performance, the other slave clocks will maintain consistency with the reference clock when the EtherCAT enters the operation state.

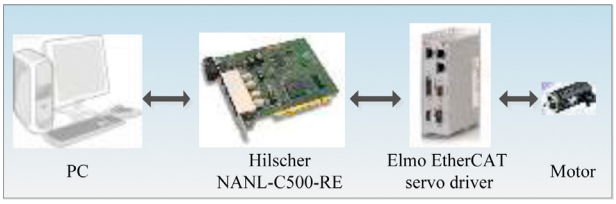


Fig. 11. Experimental platform block diagram.

Table 2
Jitter and packet loss rate for different cycle times.

Cycle time	Jitter			Packet loss rate
	Min. (μs)	Max. (μs)	Width (%)	
500 μs	26.36	−56	16.5	0
250 μs	15.12	−66.96	32.8	0
100 μs	14.48	−49.96	64.4	0

6.2. Performance

In this paper, the system cycle time is set at three different values: 500 μs, 250 μs and 100 μs. The performance of the master clock synchronization jitter and packet loss is measured at each of these three values, as shown in Fig. 12. The blue timestamps are from the reference clock, every cycle which is determined by the master clock. The line of cycle time illustrates the speed of the master clock relative to the reference clock. The green line records the offset of the master and the reference clock. Although the master virtual clock has jitter, it does not require settling time compared with the DC mechanism and PI controllers. The values related to jitter and packet loss rate are listed in Table 2. There is no packet loss for any of the 500 μs, 250 μs and 100 μs cycle times. When the cycle time is 100 μs, the value of the width of the jitter is too high. Therefore, when the cycle time is 100 μs, there is poor clock synchronization performance using the proposed method.

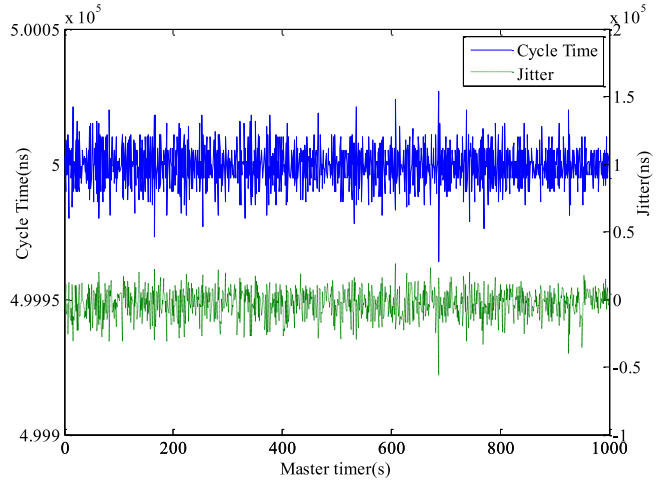
Compared with the accuracy of the DC mechanism, this method has a poor performance. The accuracy of the DC mechanism is less than 500 ns while this method is less than 50 μs. There are three reasons for master jitter. Firstly, the RTX clock resolution is only 10 μs and the cycle time is of the 100 μs order of magnitude. Secondly, the master clock has its own jitter. Thirdly, some incorrect received timestamps lead to strong jitter. Although the averaging method is used to decrease errors in the timestamp and the non-deterministic propagation delay, this does not have a sufficient effect.

7. Conclusions

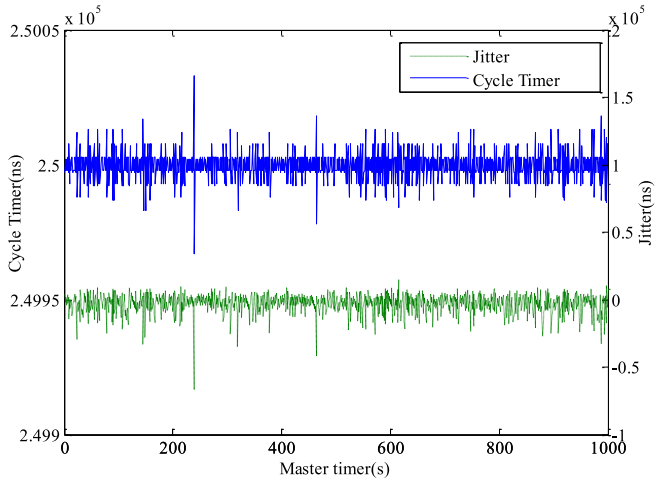
The paper compares three synchronization modes and analyzes their performance for multi-axis coordination and packet loss. The SM mode has lower multi-axis coordination, while the free running and DC modes exhibit packet loss. The packet loss is caused by lack of synchronization between the master clock and the reference clock. To ensure no packet loss throughout the whole process, this paper proposes a new clock synchronization method that can trace the reference clock speed quickly. Therefore, this method can realize clock synchronization without a settling time and guarantees no packet loss.

However, further improvement of this method is required. Although there are several reasons for strong jitter, the averaging method to decrease the effect of an incorrect timestamp is not

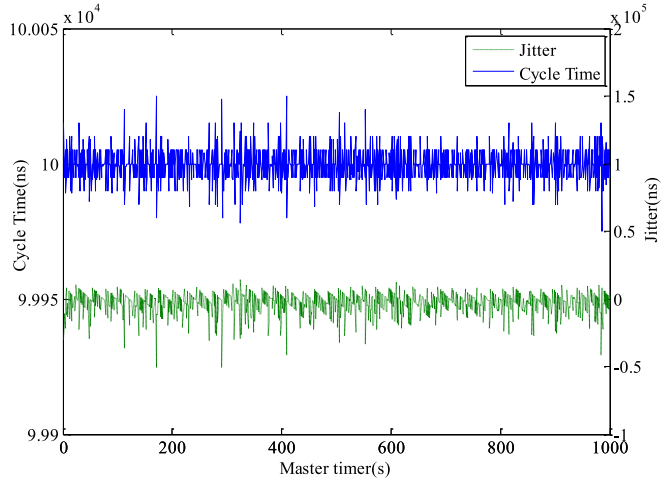
Please cite this article as: X. Chen et al., A clock synchronization method for EtherCAT master, Microprocessors and Microsystems (2016), http://dx.doi.org/10.1016/j.micpro.2016.03.002



A. Cycle time is 500us



B. Cycle time is 250us



C. Cycle time is 100us

Fig. 12. Performance of clock synchronization for different cycle times A. Cycle time is 500 μs B. Cycle time is 250 μs C. Cycle time is 100 μs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sufficient. In future research, a filter will be designed to avoid interference before the clock synchronization model.

Acknowledgements

This work was supported in part by the Fundamental Research Funds for the Central Universities (No. 2015ZZ079), the Natural Science Foundation of Guangdong Province, China (No. 2015A030308002), the Science and Technology Planning Project of Guangdong Province, China (No. 2013B011302016), and the National Natural Science Foundation of China (Nos. 61262013, 61572220, and 51575194).

References

[1] M. Rostan, J. Stubbs, D. Dzilno, EtherCAT enabled advanced control architecture, in: IEEE/SEMI Advanced Semiconductor Manufacturing Conference (July), 2010, pp. 39–44.
[2] J. Baillieul, P. Antsaklis, Control and communication challenges in networked real-time systems, Proc. IEEE 95 (1) (2007) 9–28.
[3] G. Gaderer, P. Loschmidt, T. Sauter, Improving fault tolerance in high-precision clock synchronization, IEEE Trans. Ind. Informat. 6 (2) (2010) 206–215.
[4] F. Paolo, F. Alessandra, R. Stefano, On the seamless interconnection of IEEE1588-based devices using a PROFINET IO infrastructure, IEEE Trans. Ind. Informat. 6 (3) (2010) 381–392.
[5] G. Cena, I. Bertolotti, S. Scanzio, Evaluation of EtherCAT distributed clock performance, IEEE Trans. Ind. Informat. 8 (1) (2012) 20–29.
[6] O. Dalimir, I. Reidar, P. Gunnar, EtherCAT-based platform for distributed control in high-performance industrial applications, in: 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation, 2013, pp. 1–8.
[7] G. Cena, I. Bertolotti, S. Scanzio, On the accuracy of the distributed clock mechanism in EtherCAT, in: 2010 8th IEEE International Workshop on Factory Communication Systems, 2010, pp. 43–52.
[8] D. Ganz, S. Leschke, H.D. Doran, Improving EtherCAT master-slave synchronization precisionusing PTCP embedded in EtherCAT frames, IEEE, in: 2015 Conference on Factory Communication Systems (May), 2015, pp. 1–7.
[9] P. Ferrari, A. Flammini, D. Marioli, A distributed instrument for performance analysis of real-time Ethernet networks, IEEE Trans. Ind. Informat. 4 (1) (2008) 16–25.
[10] EtherCAT Protocol Enhancements, Ethercat Technology Group, 1020 (V1.0.0) (2011).
[11] Standard for Sensor/Actuator Network Communications for EtherCAT, SEMI (E54.20).
[12] Industrial automation systems and integration – Open systems application integration framework, EtherCAT profiles, International Standard Organization 17545 (4).
[13] X.Y. Zhuang, H.H. Wang, IEEE 1588 clock synchronization algorithm based on Kalman filter, J. Electron. Meas. Instrum. 26 (9) (2012) 747–751.
[14] www.intervalzero.
[15] J. Wan, D. Li, H. Yan, P. Zhang, Fuzzy feedback scheduling algorithm based on central processing unit utilization for a software-based computer numerical control system, J. Eng. Manufact. 224 (7) (2010) 1133–1143.
[16] R. Gupta, M. Chow, Networked control system: overview and research trends, IEEE Trans. Ind. Electron. 57 (7) (2010) 2527–2535.
[17] R. Schlesinger, A. Springer, VABS-A new approach for real time Ethernet, in: IEEE 2013-39th Annual Conference on Industrial Electronics Society, 2013, pp. 4506–4511.
[18] J. Wan, D. Li, P. Zhang, Key technology of embedded system implementation for software-based CNC system, Chin. J. Mech. Eng-En. 23 (2) (2010) 241–248.

Xin Chen is a PhD Student at South China University of Technology (SCUT). She graduated from University of Electronic Science and Technology of China with a Master degree. Her research interests include the implementation and performance evaluation of real time network in actual application, clock synchronization in distributed system, cyber-physical systems.

Di Li is a Professor of School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China. Her research interests include motion control, industrial Ethernet, computer numerical control, embedded system and computer intelligence, computer vision, and cyber-physical systems.

Jiafu Wan is an Associate Professor at South China University of Technology (SCUT), China. He received the PhD degree in Mechatronic Engineering from SCUT in Jun 2008. From Oct 2008 to Jun 2012, he held a post-doctorate position in Computer Science and Engineering at SCUT. He is a project leader of several projects (e.g., NSFC). He is also managing editor for IJAACS (EI) and IJART (EI) and workshop chair of M2MC2012, M2MC2013, and MCC2013. Dr. Wan has authored/co-authored one book and 80+ scientific papers (with 30+ indexed by ISI SCIE, 40+ indexed by EI Compendex) and has been cited over 1200 times. His research interests include wireless sensor networks (WSNs), cyber-physical systems (CPS), the internet of things, mobile cloud computing, and embedded systems. He is a CCF senior member, and a member of IEEE and ACM.

Nan Zhou is a PhD student of school of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China. His research interests include distributed motion control and cyber-physical systems.

Please cite this article as: X. Chen et al., A clock synchronization method for EtherCAT master, Microprocessors and Microsystems (2016), http://dx.doi.org/10.1016/j.micpro.2016.03.002