(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116455501 A
(43)申请公布日 2023.07.18

(21)申请号 202310398383.1

(22)申请日 2023.04.14

(71)申请人 天津大学
地址 300350 天津市津南区雅观路135号

(72)发明人 王太勇 曹立波 董靖川 赵志丹 田冲 韩超

(74)专利代理机构 天津盈佳知识产权代理事务所(特殊普通合伙) 12224
专利代理师 孙宝芸

(51)Int.Cl.
H04J 3/06(2006.01)

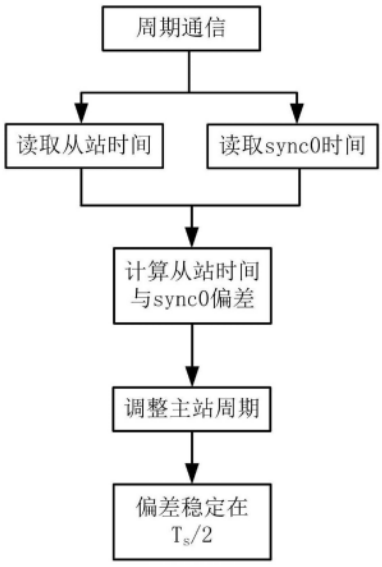
权利要求书3页 说明书14页 附图5页

(54)发明名称

一种数据帧同步通信方法、系统、数控机床

(57)摘要

本发明属于机械制造工程数控数据处理技术领域，公开了一种数据帧同步通信方法、系统、数控机床。所述数据帧同步通信方法包括：通过调整主站中断时间，实现实时以太网协议EtherCAT主从同步；利用从站时间测量获取主从站时钟偏差；调整主站周期，使偏差稳定在半个插补周期$T_s/2$内。本发明提出了一种基于参考从站稳定sync0的时钟同步机制。Sync0非常稳定，几乎没有抖动，抖动只有十几个ns。主站将在插补周期内调整其触发时间，以自适应和动态的解决因主从不同步而引起的丢包问题。此外，所提方法几乎没有增加计算和通信负载，只需要在插补周期内增加4个字节，即可修改EtherCAT帧。



CN 116455501 A

1.一种数据帧同步通信方法,其特征在于,用于分布式架构的数控系统和参考伺服,该方法包括以下步骤:

S1,通过调整主站中断时间,实现实时以太网协议EtherCAT主从同步;

S2,利用从站时间测量获取主从站时钟偏差;

S3,调整主站周期,使偏差稳定在半个插补周期$T_s/2$内。

2.根据权利要求1所述的数据帧同步通信方法,其特征在于,在步骤S1中,通过调整主站中断时间,实现实时以太网协议EtherCAT主从同步,包括:

所述主站为数控系统,从伺服驱动器获取周期性触发中断的时刻$t_{interrupt}$,IO数据处理时间T_{IO},数控系统的计算时间T_{CNC_calc}的方法,消除周期性触发中断的时刻$t_{interrupt}$,IO数据处理时间T_{IO},数控系统的计算时间T_{CNC_calc}和伺服从站周期性触发sync0的时刻t_{sync0}之间的偏差,t_{sm}是CNC发送命令位置后伺服驱动器接收命令位置的时刻,由FPRD命令从伺服驱动器寄存器中获取;伺服从站收到数据帧的时刻t_{sm}公式为:

$$t_{sm}=t_{interrupt}+T_{IO}+T_{CNC_calc}+T_{Frame}+T_{Wire_delay}$$

$T_{safety_interval}$更新为:

$$T_{safety_interval}=t_{sync0}-t_{sm}+T_{Wire_delay}$$

式中,IO数据处理时间T_{IO}、数控系统的计算时间T_{CNC_calc}和数控系统主站装载数据帧的时间开销T_{Frame}消失,而传输延迟T_{Wire_delay}与数控系统无关;$T_{safety_interval}$为安全区间,T_{wire_delay}为传输延迟。

3.根据权利要求1所述的数据帧同步通信方法,其特征在于,在步骤S2中,利用从站时间测量获取主从站时钟偏差包括:

所述从站包括多个伺服从站,第一伺服从站为参考伺服;

数控系统根据参考伺服的sync0修改通信周期,数控系统以伺服周期任务开始时间t_{sync0}作为参照,并在距离伺服周期任务开始时间t_{sync0}的数控主站发送数据包$t_{transmit}$时刻发送命令位置;$T_{interupt}$是ARM中产生周期中断的时刻;中断后,数控系统将IO数据加载到通信缓冲区,然后数控系统处理IO数据,并在$t_{transmit}$时刻向交叉以太网将IO数据帧传输到实时以太网协议EtherCAT;IO数据处理时间T_{IO}和数控系统的计算时间T_{CNC_calc}通过数控系统中的定时器获得;传输延迟T_{wire_delay}延时后,参考伺服将接收到命令位置IO数据,t表示一个时刻,T表示时间间隔;如果命令位置在t_{sync0}之前发送,即:

$$T_{safety_interval}=t_{sync0}-t_{sm}+T_{wire_delay}>0$$

式中,$T_{safety_interval}$是安全区间,t_{sync0}是伺服从站生成sync0信号的时刻,$t_{transmit}$是数控主站发送数据包的时刻,t_{sm}是伺服从站收到数控系统发送的数据帧时的时刻,T_{wire_delay}是传输延迟,只要在安全区间$T_{safety_interval}$内发送数据,则数据不丢失。

4.根据权利要求3所述的数据帧同步通信方法,其特征在于,参考伺服在t_{sm}处接收到命令位置,经过一段时间即$t_{sync0}-t_{sm}$,然后在t_{sync0}处进入其周期任务;如果当驱动系统在t_{sync0}处获取命令位置时,数控系统尚未发送,则发生丢失数据包;数控系统发送命令位置的时刻早或者晚,则接收命令位置的时刻响应变小或变大,如果$t_{sync0}-t_{sm}$的结果小于零,则发生数据包丢失;如果$t_{sync0}-t_{sm}$的结果为常数,则数据包不丢失,所述常数参考安全区间表达式为:

$$t_{sync0}-t_{sm}=T_{refs_interval}=T_s/2$$

式中，$T_{refs_interval}$为参考安全区间，T_s为插补周期，$T_s/2$为半个插补周期。

5.根据权利要求1所述的数据帧同步通信方法，其特征在于，在步骤S3中，调整主站周期，使偏差稳定在半个插补周期$T_s/2$内包括：进行数控系统与参考伺服之间的补偿，安全区间$T_{safety_interval}$和参考安全区间$T_{refs_interval}$较大者决定是提前还是延迟触发数控系统的插补周期；安全区间为：

$$T_{safety_interval}=t_{sm}-t_{sync0}+T_{wire_delay}$$

式中，$T_{safety_interval}$是安全区间，t_{sm}是伺服从站收到数控系统发送的数据帧时的时刻；t_{sync0}是伺服从站生成sync0信号的时刻；t_{sm}和t_{sync0}通过FPRD命令从参考伺服的寄存器中获得；T_{wire_delay}是传输延迟。

6.根据权利要求5所述的数据帧同步通信方法，其特征在于，数控系统与参考伺服之间的补偿具体包括：

如果安全区间$T_{safety_interval}$大于参考安全区间T_{refs_zone}，数控系统的时钟频率大于参考伺服的时钟频率，则t_{sm}变小，数控系统延迟触发插补周期，否则，提前触发插补周期。

7.根据权利要求5所述的数据帧同步通信方法，其特征在于，数控系统中的平均预补偿$\Delta\overline{T}_{CNC}$公式为：

$$\Delta\overline{T}_{CNC}=\frac{s_{CNC}}{s_{ref_s}}\cdot\Delta\overline{T}\cdot\mathrm{sgn}(\Delta\overline{T})$$

式中，$\Delta\overline{T}_{CNC}$是平均预补偿的长度，s_{CNC}和s_{ref_s}分别是CNC主站和参考伺服中n个插补周期的总和，$\Delta\overline{T}$是伺服从站的预补偿量，$\mathrm{sgn}(\Delta\overline{T})$是伺服从站的预补偿量的符号。

8.根据权利要求6所述的数据帧同步通信方法，其特征在于，数控系统的补偿长度$T_{compensation}$是平均预补偿的长度$\Delta\overline{T}_{CNC}$和插补周期内剩余的可补偿长度$T_{remain_interval}$之间的最小值，表达式为：

$$T_{compensation}=\min\left(\frac{\sum_{i=1}^{N}T_{CNC}}{\sum_{i=1}^{N}T_{ref}}\cdot\frac{\sum_{i=1}^{N}T_{safety_interval}-N\cdot T_{refs_interval}}{N},T_{CNC}-\left(T_{IO}+T_{CNC_calc}+T_{frame}\right)\right)$$

式中，$T_{compensation}$是补偿长度，T_{CNC}是数控系统的通信周期，T_{ref}是参考伺服从站的通信周期，$T_{safety_interval}$是安全区间，$T_{refs_interval}$是参考安全区间，T_{frame}是5μs的开销加上每字节数据80ns的时间开销，T_{IO}是IO数据处理时间，T_{CNC_calc}是数控主站的计算时间，N是周期数量，$T_{remain_interval}$是插补周期内剩余的可补偿长度。

9.一种数据帧同步通信系统，通过如权利要求1-8任意一项所述数据帧同步通信方法实现，其特征在于，该系统包括：

主从同步模块，用于通过调整主站中断时间，实现实时以太网协议EtherCAT主从同步；

主从站时钟偏差获取模块，用于利用从站时间测量获取主从站时钟偏差；

偏差稳定模块，用于调整主站周期，使偏差稳定在半个插补周期$T_s/2$内。

10.一种实施权利要求1-8任意一项所述数据帧同步通信方法的数控机床，其特征在于，所述数控机床包括主站和五个伺服组成的从站，五个伺服机以链状结构与主站通信，主

站运行在数控系统中,第一个从站为参考伺服。

一种数据帧同步通信方法、系统、数控机床

技术领域

[0001] 本发明属于机械制造工程数控数据处理技术领域，尤其涉及一种数据帧同步通信方法、系统、数控机床。

背景技术

[0002] EtherCAT是专为工业设计的实时以太网协议，具有通信效率高的特点。由于出色的实时通信性能，许多关于EtherCAT的科学文章出现在机械领域。此前，陈旭提出了一种在Windows上实现主站时钟与参考时钟同步的新方法。其他学者，如S.Park，在Linux上引入了一种新的同步算法。有些文章介绍了一种基于DSP+FPGA的数控平台的高精度时间同步方法。有些文章将同步帧嵌入到EtherCAT帧中，提高了EtherCAT主从站同步精度。另外，一些学者注重对分布式时钟的分析。一篇关于EtherCAT分布式时钟机制准确性的论文，对分布式时钟机制的从几个方面进行了分析。有些文章关注EtherCAT中同步性能的改进，尽管EtherCAT从站之间的有分布式同步机制，但它们之间仍然存在同步错误。为此，这篇文章提出了一种减少误差的方法。有些文章提出了一种评估方法来评估EtherCAT的分布式同步性能。有些文章考虑了许多可能存在的影响因素。有些文章在真实网络中测量评估了分布式时钟机制的精度。Sung对EPU的过程延迟感兴趣。

[0003] 分布式时钟机构在由伺服构成的驱动系统中具有优越的性能，但很少有学者研究数控系统和伺服间的EtherCAT同步机制，而且数控系统和伺服呈现链式结构，而应用于嵌入式数控系统而不带操作系统(OS)的同步机制就更少了。

[0004] 再者，EtherCAT因其100Mbps或1000Mbps的高速传输、处理器占用率低、同步性能好等诸多优势，在运动控制领域得到了广泛的应用。分布式时钟同步机制虽然在从站间表现出较强的性能，但主从之间的时钟同步方法仍然空白，在该领域研究较少。

[0005] 通过上述分析，现有技术存在的问题及缺陷为：在机床操控中，控制同步精度低，不能有效地解决嵌入式架构的数控系统与伺服系统之间的时间(节拍)同步，使得进一步的工作不能更精确的进行同步切割；现有技术中主站在插补周期内调整其触发时间中，不能自适应和动态的解决因主站与从站不同步而易引起数据丢包问题，使得机床加工中同步性能差，加工精度低。

发明内容

[0006] 为克服相关技术中存在的问题，本发明公开实施例提供了一种数据帧同步通信方法、系统、数控机床，具体涉及一种面向嵌入式数控系统中基于EtherCAT总线的数据帧同步通信方法。

[0007] 所述技术方案如下：数据帧同步通信方法用于分布式架构的数控系统和参考伺服，该方法包括以下步骤：

[0008] S1，通过调整主站中断时间，实现实时以太网协议EtherCAT主从同步；

[0009] S2，利用从站时间测量获取主从站时钟偏差；

[0010] S3,调整主站周期,使偏差稳定在半个插补周期$T_s/2$内。

[0011] 在步骤S1中,通过调整主站中断时间,实现实时以太网协议EtherCAT主从同步,包括:

[0012] 所述主站为数控系统,从伺服驱动器获取周期性触发中断的时刻$t_{interrupt}$,IO数据处理时间T_{IO},数控系统的计算时间T_{CNC_calc}的方法,消除周期性触发中断的时刻$t_{interrupt}$,IO数据处理时间T_{IO},数控系统的计算时间T_{CNC_calc}和伺服从站周期性触发sync0的时刻t_{sync0}之间的偏差,t_{sm}是CNC发送命令位置后伺服驱动器接收命令位置的时刻,由FPRD命令从伺服驱动器寄存器中获取;伺服从站收到数据帧的时刻t_{sm}公式为:

[0013] $$t_{sm}=t_{interrupt}+T_{IO}+T_{CNC_calc}+T_{Frame}+T_{Wire_delay}$$

[0014] $T_{safety_interval}$更新为:

[0015] $$T_{safety_interval}=t_{sync0}-t_{sm}+T_{Wire_delay}$$

[0016] 式中,IO数据处理时间T_{IO}、数控系统的计算时间T_{CNC_calc}和数控系统主站装载数据帧的时间开销T_{Frame}消失,而传输延迟T_{Wire_delay}与数控系统无关;$T_{safety_interval}$为安全区间,T_{wire_delay}为传输延迟。

[0017] 在步骤S2中,利用从站时间测量获取主从站时钟偏差包括:

[0018] 所述从站包括多个伺服从站,第一伺服从站为参考伺服;

[0019] 数控系统根据参考伺服的sync0修改通信周期,数控系统以伺服周期任务开始时间t_{sync0}作为参照,并在距离伺服周期任务开始时间t_{sync0}的数控主站发送数据包$t_{transmit}$时刻发送命令位置;$T_{interupt}$是ARM中产生周期中断的时刻;中断后,数控系统将IO数据加载到通信缓冲区,然后数控系统处理IO数据,并在$t_{transmit}$时刻向交叉以太网将IO数据帧传输到实时以太网协议EtherCAT;IO数据处理时间T_{IO}和数控系统的计算时间T_{CNC_calc}通过数控系统中的定时器获得;传输延迟T_{wire_delay}延时后,参考伺服将接收到命令位置IO数据,t表示一个时刻,T表示时间间隔;如果命令位置在t_{sync0}之前发送,即:

[0020] $$T_{safety_interval}=t_{sync0}-t_{sm}+T_{wire_delay}>0$$

[0021] 式中,$T_{safety_interval}$是安全区间,t_{sync0}是伺服从站生成sync0信号的时刻,$t_{transmit}$是数控主站发送数据包的时刻,t_{sm}是伺服从站收到数控系统发送的数据帧时的时刻,T_{wire_delay}是传输延迟,只要在安全区间$T_{safety_interval}$内发送数据,则数据不丢失。

[0022] 在一个实施例中,参考伺服在t_{sm}处接收到命令位置,经过一段时间即$t_{sync0}-t_{sm}$,然后在t_{sync0}处进入其周期任务;如果当驱动系统在t_{sync0}处获取命令位置时,数控系统尚未发送,则发生丢失数据包;数控系统发送命令位置的时刻早或者晚,则接收命令位置的时刻响应变小或变大,如果$t_{sync0}-t_{sm}$的结果小于零,则发生数据包丢失;如果$t_{sync0}-t_{sm}$的结果为常数,则数据包不丢失,所述常数参考安全区间表达式为:

[0023] $$t_{sync0}-t_{sm}=T_{refs_interval}=T_s/2$$

[0024] 式中,$T_{refs_interval}$为参考安全区间,T_s为插补周期,$T_s/2$为半个插补周期。

[0025] 在步骤S3中,调整主站周期,使偏差稳定在半个插补周期$T_s/2$内包括:进行数控系统与参考伺服之间的补偿,安全区间$T_{safety_interval}$和参考安全区间$T_{refs_interval}$较大者决定是提前还是延迟触发数控系统的插补周期;安全区间为:

[0026] $$T_{safety_interval}=t_{sm}-t_{sync0}+T_{wire_delay}$$

[0027] 式中,$T_{safety_interval}$是安全区间,t_{sm}是伺服从站收到数控系统发送的数据帧时的时

刻；t_{sync0}是伺服从站生成sync0信号的时刻；t_{sm}和t_{sync0}通过FPRD命令从参考伺服的寄存器中获得；T_{wire_delay}是传输延迟。

[0028] 在一个实施例中，数控系统与参考伺服之间的补偿具体包括：

[0029] 如果安全区间$T_{safety_interval}$大于参考安全区间T_{refs_zone}，数控系统的时钟频率大于参考伺服的时钟频率，则t_{sm}变小，数控系统延迟触发插补周期，否则，提前触发插补周期。

[0030] 在一个实施例中，数控系统中的平均预补偿$\Delta\overline{T}_{CNC}$公式为：

[0031] $$\Delta\overline{T}_{CNC}=\frac{s_{CNC}}{s_{ref_s}}\cdot\Delta\overline{T}\cdot\mathrm{sgn}(\Delta\overline{T})$$

[0032] 式中，$\Delta\overline{T}_{CNC}$是平均预补偿的长度，s_{CNC}和s_{ref_s}分别是CNC主站和参考伺服中n个插补周期的总和，$\Delta\overline{T}$是伺服从站的预补偿量，$\mathrm{sgn}(\Delta\overline{T})$是伺服从站的预补偿量的符号。

[0033] 在一个实施例中，数控系统的补偿长度$T_{compensation}$是平均预补偿的长度$\Delta\overline{T}_{CNC}$和插补周期内剩余的可补偿长度$T_{remain_interval}$之间的最小值，表达式为：

[0034] $$T_{compensation}=\min\left(\frac{\sum_{i=1}^{N}T_{CNC}}{\sum_{i=1}^{N}T_{ref}}\cdot\frac{\sum_{i=1}^{N}T_{safety_interval}-N\cdot T_{refs_interval}}{N},T_{CNC}-\left(T_{IO}+T_{CNC_calc}+T_{frame}\right)\right)$$

[0035] 式中，$T_{compensation}$是补偿长度，T_{CNC}是数控系统的通信周期，T_{ref}是参考伺服从站的通信周期，$T_{safety_interval}$是安全区间，$T_{refs_interval}$是参考安全区间，T_{frame}是5μs的开销加上每字节数据80ns的时间开销，T_{IO}是IO数据处理时间，T_{CNC_calc}是数控主站的计算时间，N是周期数量，$T_{remain_interval}$是插补周期内剩余的可补偿长度。

[0036] 本发明的另一目的在于提供一种数据帧同步通信系统，用于实现所述数据帧同步通信方法的数据帧同步通信系统，该系统包括：

[0037] 主从同步模块，用于通过调整主站中断时间，实现实时以太网协议EtherCAT主从同步；

[0038] 主从站时钟偏差获取模块，用于利用从站时间测量获取主从站时钟偏差；

[0039] 偏差稳定模块，用于调整主站周期，使偏差稳定在半个插补周期$T_s/2$内。

[0040] 本发明的另一目的在于提供一种实施所述数据帧同步通信方法的数控机床，所述数控机床包括主站和五个伺服组成的从站，五个伺服机以链状结构与主站通信，主站运行在数控系统中，第一个从站为参考伺服。

[0041] 结合上述的所有技术方案，本发明所具备的优点及积极效果为：针对上述现有技术存在的技术问题以及解决该问题的难度，紧密结合本发明的所要保护的技术方案以及研发过程中结果和数据等，详细、深刻地分析本发明技术方案如何解决的技术问题，解决问题之后带来的一些具备创造性的技术效果，具体描述如下：

[0042] 本发明与现有技术在调整插补周期控制量原理上存在不同，现有技术中是基于PD控制实现的，本发明是基于偏差速度设定的调整量。

[0043] 现有技术主要对象为主要是面向数控机床，本发明可以应用到多种有实时控制需求的领域，比如控制器等；本发明可以为运动控制器、PLC控制器等工业产品。

[0044] 现有技术中面向的对象是基于嵌入式系统架构，本发明不局限于嵌入式架构系统，适用于调整通信周期的任何架构的系统。

[0045] 本发明提出的ASA方法，以伺服的稳定sync0为参考，自适应调整数控系统的插补周期，有效地解决了分布式架构中数控系统与参考从站之间的时间同步问题。

[0046] 本发明提出了安全区间的概念，安全区间确保数控系统发送命令位置的时刻始终与sync0相差约$T_s/2$。无论数控时钟比伺服时钟快还是数控时钟比伺服时钟慢，都可以有效地保证在伺服进入其周期性任务时刻t_{sync0}，数控系统可以在距离sync0的$T_s/2$时提前完成的命令位置发送。数据包丢失问题有效的被ASA解决。

[0047] 本发明的实验1进行了ASA方法和没有任何同步方法的对比实验，从图11可以看出，在不同步的情况下，同步误差即实际安全区间$T_{safety_interval}$与参考安全区域$T_{refs_interval}$的偏差一直在波动，且趋势逐渐上升，最终会导致数控系统与参考伺服之间出现多个插补周期的偏差，造成数据包丢失，带来轮廓误差。实验时间，仅为300个周期，如果插补周期为2ms，即600ms，同步误差已经达到约30μs。随着同步误差增大的趋势，在20000个插补周期时出现一个插补周期大小的偏差。即，当时间为40s时，出现一次丢包。数控系统根本不能稳定加工。然而，采用ASA方法，同步误差始终在0左右波动，保证了稳定的加工。从图11可以看出，使用ASA可以显著提高CNC和参考从站之间的同步性能，验证了ASA方法在数控系统中补偿的有效性。同时，该算法丰富了适用于无操作系统的嵌入式平台上的同步方法。

[0048] 通过实验1的对比实验，实验对比了无同步、ACF和ASA方法的同步性能。如果没有同步机制，数控系统和驱动系统之间的时间漂移在每个插补周期中都会发生，并且由于累积甚至相差几个周期。ACF也是一样的。但是使用ASA，数控系统和驱动系统可以实现良好的同步性能。

[0049] 在实验2中，在500μs、1ms和2ms的不同插补周期中，本发明提出的ASA方法均表现出良好的性能，平均同步精度可达32-71ns。此外可以控制同步精度。

[0050] 本发明提出的方法几乎不添加计算负载，公式(2)用于获得补偿间隔的长度$T_{compenate}$，而FPRD cmd仅增加四个字节的通信负载。可以得出结论，随着处理器频率的提高，同步精度也能够进一步提高。因此，ASA方法可以有效地解决嵌入式架构的数控系统与伺服系统之间的时间(节拍)同步问题，进一步的工作可以更精确的同步机制并与真正的切割实验比较。

[0051] 把技术方案看作一个整体或者从产品的角度，本发明所要保护的技术方案具备的技术效果和优点，具体描述如下：EtherCAT是一种实时以太网协议，具有高达100Mbps或1000Mbps的传输速率。凭借其出色的分布式时钟(DC，Distribute Clock)同步机制，在工业控制领域被广泛应用。分布式时钟同步机制固化入从站芯片ET1100，被主流伺服驱动器从站厂商大量使用。但CNC主站无法直接利用ET1100从站芯片，发挥分布式时钟出色的同步性能。对于不具备实时操作系统(RTOS)的数控系统主站，更是无法保障通信的可靠性。为了实现将高实时性高带宽的EtherCAT应用到数控领域的目的，本发明提出了数控系统和伺服驱动器间的加工位置自适应同步控制方法，首先从分布式时钟同步机制入手，展现其伺服驱动器从站间突出的同步特性。然后分析了由于数控系统主站和伺服驱动器从站间的不同步，出现了数据包丢失现象，导致数控系统无法稳定加工。剖析了丢包现象的原因，提出了安全区间概念和基于sync0的主从自适应同步方法。最终，消除了数控系统和伺服驱动器间

的数据传输延时，同时使通信数据包丢失现象消失，实现了两者之间位置信息同步控制。并将EtherCAT主站的应用提升到一个全新高度，彻底脱离实时操作系统。通过实验验证了算法整体的适用性。

[0052] 本发明提出的ASA方法以参考伺服驱动器的稳定的sync0为参考，通过自适应调整插补周期T_s并进行预补偿，有效地解决了数控系统与参考伺服驱动器时间不同步引起的丢包问题，并消除了数据传输延时。提出了安全区间的概念。安全区间保证CNC发出命令位置的时刻始终离sync0约$T_s/2$，无论CNC时钟比参考伺服时钟快还是CNC时钟比参考伺服时钟慢，都能有效保证在t_{sync0}伺服进入周期任务时，CNC可以提前在距离sync0 $T_s/2$时刻完成的命令位置发送，解决了报文丢失问题。通过实验1的对比实验，说明了ASA方法能够有效的解决丢包问题。在不同步的情况下，数控系统与伺服驱动器在每个插补周期内都会产生时间漂移，并且随着时间的推移，累积漂移甚至可能是几个周期。然而，有了ASA，CNC和伺服驱动器具有良好的同步性能。在MSTS的对比实验中，如果不受中断模块的限制，ASA同步误差比MSTS降低了4.182倍。另外，该算法完全不受实时系统的限制，不仅解决了丢包现象，而且精准加工。而且，采用多次(例如500次)测量的平均值所消耗的时间，ASA几乎比MSTS减少了一半。实验三的结果表明，该算法适用于不同品牌或厂家的伺服驱动器。实验四中，在不同的插补周期500μs、1ms和2ms下，文中提出的ASA方法均表现出良好的同步性能，平均同步精度可达32-71ns。重要的是，ASA将EtherCAT提升到了一个全新的水平，完全实现了EtherCAT主站不再依赖RTOS。它超越了以往学者的研究。而且，该方法基于一个简单的等差数列求和公式，以几乎不增加计算量的方式，便获得精确的预补偿$\Delta T_{pre\text{-}compensation}$。只需在FPRD命令中添加两个或四个字节，几乎不会增加通信负载。另外，CNC主站发送命令位置的时刻和参考伺服接收命令位置的时刻具有共同的参考t_{sync0}，消除了CNC与参考伺服相对独立的绝对时间产生的累积误差。随着处理器频率的提高，同步精度能够提高为1ns。因此，ASA方法可以有效地解决无操作系统的嵌入式数控系统和伺服驱动器的时间(节拍)同步问题。该方法也可用于分布式多主系统的时间同步通信。

[0053] 相比于现有技术，本发明的优点还包括：本发明提出了一种基于参考从站稳定sync0的时钟同步机制。sync0非常稳定，几乎没有抖动，抖动只有十几个ns。主站将在插补周期内调整其触发时间，以自适应和动态的解决因主从不同步而引起的丢包问题。此外，所提方法几乎没有增加计算和通信负载，只需要在插补周期内增加4个字节。即可修改EtherCAT帧。在机床上进行了实验，证明了所提方法在提高同步性能方面具有优势，平均帧抖动仅有32-71ns。

[0054] 本发明使EtherCAT主站不再局限于实时操作系统，拓宽了应用范围。本发明提出了一种自适应同步算法(ASA)，用于分布式架构的嵌入式数控系统和参考伺服。本发明提出的同步机制可以自适应、动态调整插补周期调整代替调整时钟，以提高加工精度，同步填补EtherCAT协议的空白。所提出的方法几乎没有增加计算和通信负载，只需要在插补周期内添加四个字节，用来修改EtherCAT帧。本发明解决了EtherCAT受限于实时操作系统的问题。

附图说明

[0055] 此处的附图被并入说明书中并构成本说明书的一部分，示出了符合本公开的实施例，并与说明书一起用于解释本公开的原理；

[0056] 图1是本发明实施例提供的数据帧同步通信方法流程图；

[0057] 图2是本发明实施例提供的数据帧同步通信方法原理图；

[0058] 图3是本发明实施例提供的EtherCAT数据帧结构图；

[0059] 图4是本发明实施例提供的数据包丢失的原因分析图；

[0060] 图5是本发明实施例提供的数控系统和驱动系统的时钟模型示意图；

[0061] 图6是本发明实施例提供的时钟源的温度漂移示意图；

[0062] 图7是本发明实施例提供的从数控系统传输到驱动系统的数据帧的通信时序示意图；

[0063] 图8是本发明实施例提供的参考安全区间示意图；

[0064] 图9是本发明实施例提供的ASA算法流程图；

[0065] 图10是本发明实施例提供的轴实验平台示意图；

[0066] 图11是本发明实施例提供的基于安全区间的ASA原理图；

[0067] 图12是本发明实施例提供的安全区间转移到伺服驱动器原理图；

[0068] 图13是本发明实施例提供的半个插补周期$T_s/2$参考安全区域长度原理图。

具体实施方式

[0069] 为使本发明的上述目的、特征和优点能够更加明显易懂，下面结合附图对本发明的具体实施方式做详细的说明。在下面的描述中阐述了很多具体细节以便于充分理解本发明。但是本发明能够以很多不同于在此描述的其他方式来实施，本领域技术人员可以在不违背本发明内涵的情况下做类似改进，因此本发明不受下面公开的具体实施的限制。

[0070] 实施例1，如图1所示，本发明实施例提供的数据帧同步通信方法，用于分布式架构的嵌入式数控系统和参考伺服，包括以下步骤：

[0071] S1，通过调整主站中断时间，实现实时以太网协议EtherCAT主从同步；使EtherCAT主站，完全脱离实时系统的限制。

[0072] S2，利用从站时间测量获取主从站时钟偏差；传统主从站时间偏差利用主站时间减去从站偏差，存在固有偏差。而本方法，完全消除两者的固有偏差，实现偏差精确测量。

[0073] S3，调整主站周期，使偏差稳定在半个插补周期$T_s/2$内。

[0074] 在本发明实施例中，在步骤S1中，所述主站为数控系统；在步骤S2中，所述从站包括多个伺服从站，第一伺服从站为参考伺服。

[0075] 在本发明实施例中，图2是本发明实施例提供的数据帧同步通信方法原理。

[0076] 在本发明实施例步骤S1中，通过调整主站中断时间，实现实时以太网协议EtherCAT主从同步，包括：

[0077] 为了解决数控系统与伺服驱动器之间的数据包丢失问题，消除两者之间的数据传输延时，ASA动态修改数控系统的插补周期，实现与参考伺服的时间同步的目的。如上所述，数控系统根据参考伺服的sync0修改自身的插补周期。在图11中，$t_{interrupt}$是ARM中的插补周期的中断时刻，$t_{transmit}$是数控主站发送数据包的时刻。$t_{interrupt}$完成后，CNC将IO数据加载到通信缓冲区，处理IO数据并在$T_{transmit}$时刻通过交叉以太网线将IO数据帧传输到以太网。CNC系统以伺服周期任务的开始时间t_{sync0}为基准，在远离t_{sync0}的$t_{transmit}$时刻向网络发送命令位置，伺服驱动器接收到命令位置的IO数据，延时等待一段时间，再进入其任务周期。t表示

一个时刻,T是时间间隔。如果命令位置在t_{sync0}之前发送,即:

[0078] $T_{safety_interval}=t_{sync0}-(t_{interrupt}+T_{IO}+T_{CNC_calc}+T_{Frame})>0$

[0079] 不会发生数据包丢失。$t_{transmit}$和t_{sync0}之间的间隔称为$T_{safety_interval}$。t_{sync0}可通过FPRD cmd从驱动器寄存器中获取。

[0080] 但是,$t_{interrupt}$以及T_{IO}和T_{CNC_calc}都来自CNC系统,而t_{sync0}则来自伺服驱动器。由于时钟源的不同,$t_{interrupt}$和t_{sync0}之间仍然存在偏差。因此,本发明提出了一种不从CNC系统获取$t_{interrupt}$、T_{IO}和T_{CNC_calc},而从伺服驱动器获取周期性触发中断的时刻$t_{interrupt}$,IO数据处理时间T_{IO},数控系统的计算时间T_{CNC_calc}的方法,消除周期性触发中断的时刻$t_{interrupt}$,IO数据处理时间T_{IO},数控系统的计算时间T_{CNC_calc}和伺服从站周期性触发sync0的时刻t_{sync0}之间的偏差,t_{sm}是CNC发送命令位置后伺服驱动器接收命令位置的时刻,直接由FPRD命令从伺服驱动器寄存器中获取;伺服从站收到数据帧的时刻t_{sm}公式为:

[0081] $t_{sm}=t_{interrupt}+T_{IO}+T_{CNC_calc}+T_{Frame}+T_{Wire_delay}$

[0082] $T_{safety_interval}$更新为:

[0083] $T_{safety_interval}=t_{sync0}-t_{sm}+T_{Wire_delay}$

[0084] 式中,IO数据处理时间T_{IO},数控系统的计算时间T_{CNC_calc}和数控系统主站装载数据帧的时间开销T_{Frame}消失,而T_{Wire_delay}与数控系统无关;$T_{safety_interval}$为安全区间,T_{wire_delay}为传输延迟,另外,这种消失将减少处理器的计算量,提高实时性。

[0085] 在本发明实施例步骤S3中,调整主站周期,使偏差稳定在半个插补周期$T_s/2$内。

[0086] 图12中,伺服驱动器在t_{sm}接收到命令位置,经过一段时间,即$t_{sync0}-t_{sm}$,然后在t_{sync0}进入其周期性任务,如果伺服驱动器在t_{sync0}接收到命令位置时CNC系统没有发送,则会出现丢包现象。如果CNC系统发出命令位置的时刻早或晚,则接收命令位置的时刻t_{sm}小或大。$t_{sync0}-t_{sm}$的结果是可变的,如果$t_{sync0}-t_{sm}$的结果小于零,则会发生丢包。如果$t_{sync0}-t_{sm}$的结果为正常数,则不可能发生丢包,如果$t_{sync0}-t_{sm}$是常数而不是变量,处理器的计算负担将减少。因此,引入了参考安全区域概念,并将其命名为T_{refs_zone}。

[0087] 对于ASA参考安全区域合适的长度,在数控机床中,CNC的时钟更快。因此,公式中的t_{sm}将更小,因此,安全区间$T_{safety_interval}=t_{sync0}-t_{sm}+T_{Wire_delay}$将更长。然而,对于另一台数控机床中CNC时钟更慢,导致另一台机床的安全区间$T_{safety_interval}$的长度将更小。因此,为了提高ASA对两种机床的通用性,需要设置合理的参考区值T_{refs_zone}。

[0088] 如图13所示,假设$T_{refs_zone}=T_s/2$。如果一台机床中,数控系统时钟比参考伺服时钟快,那么t_{sm}更小。但是,帧中的命令位置必须在$t_{1,sync0}$之后由CNC发送,$t_{1,sync0}$是在上一个插补周期T_s中参考伺服进入周期任务的时刻。否则,在上一插补周期T_s中未处理的命令位置将被覆盖,会出现数据包丢失现象。因此,在上一个的插补周期中,安全区的最大长度为$T_s/2$。即使如果在另一台机床中,数控系统时钟比参考伺服时钟慢,那么t_{sm}更大。帧中的命令位置必须在$t_{2,sync0}$之前由CNC发送。$t_{2,sync0}$是当前插补周期T_s中参考伺服进入周期任务的时刻。否则,在当前插补周期T_s中无法接收命令位置,也会发生数据包丢失现象。因此,在当前插补周期内,安全区的最大长度仍然是$T_s/2$。$T_s/2$长度参考安全区域在两台机床中都是有意义的,所以参考安全区域T_{refs_zone}的长度是$T_s/2$。

[0089] 实施例2,本发明实施例提供的面向数控系统的EtherCAT总线主从同步方法包括:

[0090] 步骤1,数控系统和参考伺服之间的丢包分析:

[0091] 步骤1.1,丢包分析;EtherCAT协议是一种特殊的以太网协议,其中以太网类型0x88A4,而广为人知的TCP/IP或UDP的以太网类型0x0800。EtherCAT帧,也称为数据包,嵌入在以太网帧,与TCP/IP和UDP相同。如图3,EtherCAT数据帧结构所示,一个完整的EtherCAT帧由以太网帧头、EtherCAT帧头和EtherCAT PDU组成。

[0092] 数据包丢失的原因分析如图4所示。T_{CNC}是数控系统的插补周期,T_{drive}是驱动系统的插补周期。t_{wire_delay}是网络线路延迟。如果数控系统和驱动系统的时钟源相同,则数控系统应在t_1处向驱动系统发送命令位置,然而数控系统却在t_2处发送命令位置。原因是数控系统的晶体振荡器频率低于驱动系统的晶体振荡器频率,导致数控系统的实际时间段T_{CNC}在同一个插补周期T_s内却大于T_{drive}。当驱动系统接收命令位置时,然而数控系统尚未发送的命令位置。这种非同步现象将引起数据包丢失。

[0093] 步骤1.2,由工作频率差异和启动时间引起的异步:数控系统和驱动系统的工作时钟源不同,这是丢包的主要原因。数控系统和驱动系统的节拍几乎不可能具有相同的振动频率,如图5数控系统和驱动系统的时钟模型所示,即使数控系统和驱动系统的晶体来自同一型号和系列,偏差仍然存在。以从8:00到8:15的15分钟为例,即UTC+8,也称为北京时间。数控系统的实际时间为14.9分钟,而在相同的15分钟内驱动系统的为15.1分钟。数控系统和驱动系统中存在的偏差,导致数控系统发送命令位置的时间与驱动系统的接收命令位置的时间不一致,进而数控系统发送的命令位置不能被驱动系统接收。另外,数控系统和驱动系统的启动时间也不一致,这导致数控系统和驱动系统的存在初始差异。这种差异也导致数控系统的发送位置与驱动系统的接收位置的时刻不一致。而且,这种差异始终存在。

[0094] 步骤1.3,温度引起的异步:此外,还存在一种现象,即任何晶体的时钟频率都会随温度漂移。如图6时钟源的温度漂移所示,是数控系统的时钟在不同温度下的时钟频率,驱动系统也一致。温度漂移现象也会导致数控系统发送位置和驱动系统接收位置的时序不一致。

[0095] 为了解决这个问题,传统的方法是选择两个时钟源中的一个作为参考时钟源,另一个使用参考时钟作为参考来稍微修改自己的节拍,但是在处理器的许多架构中校正时钟的节拍是非常困难的,例如ARM。本发明中提出的ASA(Adaptive synchroniztion algorithm)方法,无需校正时钟源的频率,就可以消除数控系统和驱动系统之间的偏差。

[0096] 步骤2,数控嵌入式数控系统中的自适应同步算法:从本质上讲,丢失数据包是错误的陈述。事实上,驱动系统总是定时接收数控系统发送的命令位置。根本问题是,数控系统没有及时发送命令位置,如图4。

[0097] 步骤2.1,不丢失数据包的安全区间:很少有论文提出在分布式架构中调整时间或时钟以实现同步,但在嵌入式数控系统中不可能调整晶体振荡器的时间或频率。因此,本发明提出调整插补周期,实现数控系统与伺服之间的同步。伺服中存在3类分布式时钟同步机制,并分析了同步精度。之所以选择同步到sync0的机制,是因为sync0非常稳定,只有12ns的抖动。基于此,所提出的ASA方法充分利用了来自参考从站非常稳定的sync0,并将sync0设置为参考以调整数控系统插补周期。

[0098] 如图7所示,是从数控系统传输到驱动系统的数据帧的通信时序。时序由T_{io},T_{CNC_calc},T_{frame},T_{wire_delay},T_{adjust_time},T_{servos_delay}组成。T_{io}是将IO数据加载到通信缓冲区的时间。T_{CNC_calc}是数控的传输时间。T_{frame}是通过交叉以太网电缆将IO数据帧传输到以太网的时

间，T_{frame}大约是5μs开销加上每字节开销80ns的时间。T_{wire_delay}是从数控系统到驱动系统的帧延迟时间，称为导线延迟，100BASE-TX的1μs延时和大约5ns/m的线路延迟下。T_{adjust_time}是在数控系统内部通过调整周期处理不同步问题的时间。T_{servos_delay}是伺服系统的延迟时间。T_{servos_delay}可以由每个伺服独立设置，通常为0。

[0099] 为了解决数控系统和驱动系统之间的数据包丢失问题，ASA可以动态修改数控系统的周期，并同步到参考伺服的计时。如上所述，数控系统根据参考伺服的sync0修改其通信周期。数控系统以伺服周期任务开始时间t_{sync0}作为参照，并在距离伺服周期任务开始时间t_{sync0}的数控主站发送数据包$t_{transmit}$时刻发送命令位置；$T_{interupt}$是ARM中产生周期中断的时刻；中断后，数控系统将IO数据加载到通信缓冲区，然后数控系统处理IO数据，并在$t_{transmit}$时刻向交叉以太网将IO数据帧传输到实时以太网协议EtherCAT；T_{IO}和t_{CNC_calc}可以通过嵌入式数控系统中的定时器获得。T_{frame}大约是5μs的开销加上每字节数据80ns的时间开销。T_{wire_delay}后，驱动系统将接收到IO数据，这些IO数据是命令位置。T_{wire_delay}在100M以太网上的1μs基础延时，加上5ns/m的网线延迟。t表示一个时刻，T表示时间间隔。如果命令位置是在t_{sync0}之前发送，即：

[0100] $$T_{safety_interval}=t_{sync0}-t_{sm}+T_{wire_delay}>0 \quad (1)$$

[0101] 式中，t_{sync0}是伺服从站生成sync0信号的时刻，t_{sm}伺服从站收到数控系统发送的数据帧时的时刻。只要在区间$T_{safety_interval}$内发送数据，则数据不丢失。

[0102] 步骤2.2，基于sync0的参考安全区间：如图8参考安全区间所示，驱动系统在t_{sm}处接收到命令位置，经过一段时间即$t_{sync0}-t_{sm}$，然后在t_{sync0}处进入其周期任务。如果当驱动系统在t_{sync0}处获取命令位置时，数控系统尚未发送，则会发生丢失数据包的现象。数控系统发送命令位置的时刻早或者晚，则接收命令位置的时刻响应变小或变大。导致$t_{sync0}-t_{sm}$的结果是变量，如果$t_{sync0}-t_{sm}$的结果小于零，则会发生数据包丢失。如果$t_{sync0}-t_{sm}$的结果是一个正常数，则数据包丢失是不可能的。另外，如果处理器计算常数而不是变量的，则计算负担将减少。因此，$t_{sync0}-t_{sm}=T_{refs_interval}=T_s/2$被设置为常数，这个常数可作为参考安全区间，并命名为$T_{refs_interval}$，T_s为插补周期，$T_s/2$为半个插补周期。

[0103] 步骤2.3，嵌入式数控系统中的补偿范围：包括数控系统与参考伺服之间的补偿，如图9所示，$T_{safety_interval}$和$T_{refs_interval}$较大者决定是提前还是延迟触发数控系统的插补周期。参考安全区间为$T_{refs_interval}$，值为$T_s/2$。安全区间为：

[0104] $$T_{safety_interval}=t_{sm}-t_{sync0}+T_{wire_delay}$$

[0105] t_{sm}是驱动系统接收命令位置的时刻，t_{sync0}是获取命令位置并在驱动系统中进入其周期性任务的时刻，t_{sm}和t_{sync0}可以通过FPRD命令从参考伺服的寄存器中获得，传输延迟为T_{wire_delay}。

[0106] 具体原理如图9所示，ASA算法包括：如果$T_{safety_interval}$大于T_{refs_zone}，那么CNC系统的时钟频率大于参考伺服的时钟频率。则t_{sm}变小，CNC系统应该延迟触发插补周期。否则，提前触发插补周期。所示，$\Delta\overline{T}$不能直接用于数控系统CNC的自适应调整周期，因为它是根据伺服驱动器的时钟来计算的。数控系统CNC中的平均预补偿$\Delta\overline{T}_{CNC}$公式为：

[0107] $$\Delta\overline{T}_{CNC}=\frac{s_{CNC}}{s_{ref_s}}\cdot\Delta\overline{T}\cdot\mathrm{sgn}(\Delta\overline{T})$$

[0108] 式中，$\Delta\overline{T}_{CNC}$是平均预补偿的长度，s_{CNC}和s_{ref_s}分别是CNC主站和参考伺服中n个插补周期的总和，$\Delta\overline{T}$是伺服从站的预补偿量，$sgn(\Delta\overline{T})$是伺服从站的预补偿量的符号。

[0109] 在一个实施例中，数控系统的补偿长度$T_{compensation}$是平均预补偿的长度$\Delta\overline{T}_{CNC}$和插补周期内剩余的可补偿长度$T_{remain_interval}$之间的最小值，表达式为：

[0110] $$T_{compensation} = \min\left(\frac{\sum_{i=1}^{N} T_{CNC}}{\sum_{i=1}^{N} T_{ref}} \cdot \frac{\sum_{i=1}^{N} T_{safety_interval} - N \cdot T_{refs_interval}}{N}, T_{CNC} - \left(T_{IO} + T_{CNC_calc} + T_{frame}\right)\right)$$

[0111] 式中，$T_{compensation}$是补偿长度，T_{CNC}是数控系统的通信周期，T_{ref}是参考伺服从站的通信周期，$T_{safety_interval}$是安全区间，$T_{refs_interval}$是参考安全区间，T_{frame}是5μs的开销加上每字节数据80ns的时间开销，T_{IO}是IO数据处理时间，T_{CNC_calc}是数控主站的计算时间，N是周期数量，$T_{remain_interval}$是插补周期内剩余的可补偿长度。

[0112] 实施例3，本发明实施例提供一种数据帧同步通信系统，包括：

[0113] 主从同步模块，用于通过调整主站中断时间，实现实时以太网协议EtherCAT主从同步；

[0114] 主从站时钟偏差获取模块，用于利用从站时间测量获取主从站时钟偏差；

[0115] 偏差稳定模块，用于调整主站周期，使偏差稳定在半个插补周期$T_s/2$内。

[0116] 上述装置/单元之间的信息交互、执行过程等内容，由于与本发明方法实施例基于同一构思，其具体功能及带来的技术效果，具体可参见方法实施例部分，此处不再赘述。

[0117] 所属领域的技术人员可以清楚地了解到，为了描述的方便和简洁，仅以上述各功能单元、模块的划分进行举例说明，实际应用中，可以根据需要而将上述功能分配由不同的功能单元、模块完成，即将所述装置的内部结构划分成不同的功能单元或模块，以完成以上描述的全部或者部分功能。实施例中的各功能单元、模块可以集成在一个处理单元中，也可以是各个单元单独物理存在，也可以两个或两个以上单元集成在一个单元中，上述集成的单元既可以采用硬件的形式实现，也可以采用软件功能单元的形式实现。另外，各功能单元、模块的具体名称也只是为了便于相互区分，并不用于限制本发明的保护范围。上述系统中单元、模块的具体工作过程，可以参考前述方法实施例中的对应过程。

[0118] 本发明实施例提供的数据帧同步通信方法已经应用于天大精益S6系列数控系统，实现该系列数控系统的高性能输出。

[0119] 实施例4，本发明实施例还提供了一种计算机设备，该计算机设备包括：至少一个处理器、存储器以及存储在所述存储器中并可在所述至少一个处理器上运行的计算机程序，所述处理器执行所述计算机程序时实现上述任意各个方法实施例中的步骤。

[0120] 本发明实施例还提供了一种计算机可读存储介质，所述计算机可读存储介质存储有计算机程序，所述计算机程序被处理器执行时可实现上述各个方法实施例中的步骤。

[0121] 本发明实施例还提供了一种信息数据处理终端，所述信息数据处理终端用于实现于电子装置上执行时，提供用户输入接口以实施如上述各方法实施例中的步骤，所述信息数据处理终端不限于手机、电脑、交换机。

[0122] 本发明实施例还提供了一种服务器，所述服务器用于实现于电子装置上执行时，

提供用户输入接口以实施如上述各方法实施例中的步骤。

[0123] 本发明实施例提供了一种计算机程序产品，当计算机程序产品在电子设备上运行时，使得电子设备执行时可实现上述各个方法实施例中的步骤。

[0124] 所述集成的单元如果以软件功能单元的形式实现并作为独立的产品销售或使用时，可以存储在一个计算机可读取存储介质中。基于这样的理解，本申请实现上述实施例方法中的全部或部分流程，可以通过计算机程序来指令相关的硬件来完成，所述的计算机程序可存储于一计算机可读存储介质中，该计算机程序在被处理器执行时，可实现上述各个方法实施例的步骤。其中，所述计算机程序包括计算机程序代码，所述计算机程序代码可以为源代码形式、对象代码形式、可执行文件或某些中间形式等。所述计算机可读介质至少可以包括：能够将计算机程序代码携带到拍照装置/终端设备的任何实体或装置、记录介质、计算机存储器、只读存储器(Read-Only Memory，ROM)、随机存取存储器(Random Access Memory，RAM)、电载波信号、电信信号以及软件分发介质。例如U盘、移动硬盘、磁碟或者光盘等。

[0125] 在上述实施例中，对各个实施例的描述都各有侧重，某个实施例中没有详述或记载的部分，可以参见其它实施例的相关描述。

[0126] 为验证本发明实施例提供的数据帧同步通信方法的积极效果，本发明进行如下实验。

[0127] 本发明通过实验验证所提方法的同步性能。如图10轴实验平台，实验平台是由一个数控系统和五个伺服组成的数控机床，五个伺服机以链状结构的形式出现，数控系统是EtherCAT主站，主站是Simle Open EtherCATMaster(SOME)，并运行在STM32H765TI嵌入式数控系统，嵌入式数控系统的频率为400MHz。五个伺服是从站，即驱动系统。第一个从站是参考伺服。

[0128] 实验1，进行了ASA方法和没有任何同步方法的对比实验。本发明实施例提供的ASA和没有同步算法的同步误差实验中可以看出，在不同步的情况下，同步误差即实际安全区间$T_{safety_interval}$与参考安全区域T_{refs_zone}的偏差一直在波动，且趋势逐渐上升，最终会导致数控系统与参考伺服之间出现多个插补周期的偏差，造成数据包丢失，带来轮廓误差。实验时间，仅为300个周期，如果插补周期为2ms，即600ms，同步误差已经达到约30μs。随着同步误差增大的趋势，在20000个插补周期时出现一个插补周期大小的偏差。即，当时间为40s时，出现一次丢包。数控系统根本不能稳定加工。然而，采用ASA方法，同步误差始终在0左右波动，保证了稳定的加工。从本发明实施例提供的ASA和没有同步算法的同步误差实验中可以看出，使用ASA可以显著提高CNC和参考从站之间的同步性能，验证了ASA方法在数控系统中补偿的有效性。同时，该算法丰富了适用于无操作系统的嵌入式平台上的同步方法。

[0129] 实验2，使用2m长的以太网电缆，数控系统和参考伺服之间的测量值为1.01μs。实验的同步精度为1μs。T_{io}和T_{CNC_calc}可以通过嵌入式数控系统中的定时器获得。T_{frame}大约是5μs的开销加上每字节数据80ns开销。T_{CNC}是插补周期。最后，补偿区间$T_{compensate}$的长度可以通过公式(2)得到，达到自适应数控系统插补周期的目的。

[0130] 为了进一步评估ASA的同步性能，建立了对比实验2。Park提出了一种EtherCAT主从站之间的同步方法，将主站和参考从站之间的偏差转化至全部的伺服从站中的偏差(master and reference servo to servos，MSTS)。MSTS通过调整所有伺服驱动器的时间

而不是主站的系统时间来实现主从同步。实验在基于STM32H7x的嵌入式系统中进行，频率为400MHz。因此，主站的最大频率为400MHz，主站中定时器模块的最大频率为200MHz。也就是说，主站定时器的最大精度为5ns，因此主站与参考伺服之间的系统时间误差通过MSTS传递给伺服驱动器的精度也为5ns。而STM32H7x由于中断模块的限制，中断模块的精度仅为50ns，而CNC的插补周期由中断模块产生，因此ASA调整的插补周期仅为50ns的倍数。

[0131] 表1不同方法的同步误

[0132]

方法	同步误差			
	Max（ns）	Mean（ns）	Min（ns）	调整单位（ns）
ASA	651	42.296	-749	50
MSTS	104	17.668	-50	5

[0133] 表1是ASA和MSTS实验的数据分析，本发明提供的ASA和MSTS方法的同步误差实验结果中可以发现，MSTS的同步误差比ASA稍小。这是由于主站中断模块的限制造成的。如果ASA精度也可以调整5ns，则同步误差将小于MSTS，具体分析如下。ASA只能以50ns为单位调整插补周期T_s，这远远大于MSTS 5ns调整单位。此外，伺服驱动器的时间精度为1ns单位，该1ns单位的调整量来自ETG的时间控制回路。因此，MSTS调整单元精度为1-5ns。公式(3)是预补偿序列与抖动之间的关系，抖动可由公式$e=T_{safety_interval}-T_{refs_zone}$中获知。在实验部分抖动称为同步误差。公式(4)中，由于n是500，因为最后一项e_{n+1}减去第一项e_1的差远远小于500个同步误差e_i的总和，所以将公式(4)近似为公式(5)，忽略了公式(4)前两项，同时比例系数k为1。e_{mean}是同步误差e_i之和的平均值。因此，可以得出结论，预补偿的平均值与平均同步误差成正比。因此，如果插补周期T_s的调整单元，即平均预补偿，减少10倍，从50ns变为5ns，则平均同步误差也减少10倍。实验中采用多次测量的平均值。根据平均同步误差，可以精确推断，如果ASA精度如果在不受中断模块限制的情况下调整单位变为5ns，那么ASA的同步误差42.296ns/10＝4.23ns将大大小于MSTS中的17.668ns，同步误差减小4.182倍。

[0134]
$$\begin{pmatrix} \Delta T_{pc_2} \\ \Delta T_{pc_3} \\ \vdots \\ \Delta T_{pc_n} \\ \Delta T_{pc_{n+1}} \end{pmatrix} = 2\begin{pmatrix} e_2 \\ e_3 \\ \vdots \\ e_n \\ e_{n+1} \end{pmatrix} - \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{n-1} \\ e_n \end{pmatrix} \quad (3)$$

[0135]
$$\Delta\overline{T} = \frac{1}{n}\left(e_{n+1} - e_1 + \sum_{i=1}^{n} e_i\right) \quad (4)$$

[0136]
$$\Delta\overline{T} \propto k \cdot e_{mean} \quad (5)$$

[0137] 可以看到，MSTS使用主站时钟作为基准来调整所有伺服驱动器的时间。因此，伺服驱动器的时间精度和加工精度完全取决于主站的实时性。如果主站不是实时系统或者主站的实时性不是很好，比如不带RTOS的实验平台ARM STM32H7x，虽然不会出现丢包，但是插补精度和加工精度会大大降低。然而，基于极其稳定sync0的ASA完全不受RTOS的限制，不仅解决了丢包现象，而且能够稳定地精确地加工。另外，在ASA中采用多次(例如500次)测量的平均时间消耗为1228ns，而MSTS为2440ns。计算开销几乎减少了一半。

[0138] 实验3，为了验证所提出的ASA的通用性，在不同的伺服驱动器上进行了实验3。实验仍由1台数控系统主站和5台伺服从站组成。5台伺服驱动器由CTB的2台伺服驱动器和久同的3台伺服驱动器组成。一个实验中，来自CTB的伺服被用作参考伺服，即第一个从站。另一个实验中，参考伺服来自久同的3台伺服驱动器。表2是两个实验的结果。从表2可以看出，无论是最大同步误差、最小同步误差还是平均同步误差，两个实验的结果都非常相似。这是因为所提出的ASA方法与伺服品牌或制造商无关，它只与伺服驱动器中的EtherCAT从站芯片或FPGA核有关，EtherCAT从站芯片或FPGA核都来自同一个ETG技术团体。因此，本发明提出的ASA算法适用于不同的伺服品牌或厂家。

[0139] 表2不同伺服城从站厂家的同步误差

[0140]

制造商	同步误差		
	Max（ns）	Mean（ns）	Min（ns）
CTB	471	-39.778	-569
久同	491	-30.919	-629

[0141] 实验4是EtherCAT主站分别以500μs、1ms、2ms插补周期发送命令位置的实验。本发明提供的数控系统500μs插补周期的实验中，样本数量是1200个。实验结果表明，最大同步误差为929ns，平均同步误差仅为61ns。本发明给出了数控系统(CNC)1ms插补周期的实验，共3400个样本。实验结果表明，最大同步误差为769ns。本发明给出了数控系统2ms插补周期的实验，也一共是3400个样本。实验结果表明，最大同步误差仅为749ns。

[0142] 为了进一步证明所提方法的性能，每个实验用ASA方法重复五轮以上，并利用最大同步误差、最小同步误差和平均同步误差进行分析，如表3所示。最大同步误差为正值，表示CNC系统发送命令位置的时刻落后于伺服驱动器的sync0的距离大于$T_s/2$，$T_s/2$是参考安全区域的长度。最小同步误差为负值，表示发送命令位置的时刻落后于伺服驱动器的sync0的距离小于$T_s/2$。平均同步误差表示CNC系统发送命令位置的时刻与t_{sync0}-$T_s/2$之间的偏差的平均值。最大同步误差仅为471-611ns。这意味着CNC发送指令的时刻位置距sync0的距离为$T_s/2$+471-611ns。最小同步误差为-909～-609ns，平均同步误差为32-71ns。即，CNC发送命令位置的时刻距sync0的平均距离仅为$T_s/2$+32～71ns。也就是说，在伺服驱动器接收到命令位置之前$T_s/2$的时刻，CNC几乎可以稳定地完成发送命令位置。因此，不可能发生丢包现象。由此可见，ASA的稳定效果非常好。同步误差越小，与sync0的距离越稳定在$T_s/2$。CNC可以从距离sync0的$T_s/2$时刻稳定的发出命令位置，sync0是伺服驱动器进入周期性任务的时刻，ASA有效避免了丢包现象。因此，本发明提出的PDJC能够有效地解决丢包问题。

[0143] 表3不同插补周期的同步误差

[0144]

	数控系统插补周期		
同步误差	500μs	1ms	2ms
Max（ns）	471	511	611
Min（ns）	-909	-729	-609
Mean（ns）	71.36	32.457	44.569

[0145] 本发明中提出的ASA方法以参考伺服驱动器的稳定的sync0为参考，通过自适应调整插补周期T_s并进行预补偿，有效地解决了数控系统与参考伺服驱动器时间不同步引起的丢包问题，并消除了数据传输延时。提出了安全区间的概念。安全区间保证CNC发出命令位置的时刻始终离sync0约$T_s/2$，无论CNC时钟比参考伺服时钟快还是CNC时钟比参考伺服时钟慢，都能有效保证在t_{sync0}伺服进入周期任务时，CNC可以提前在距离sync0$T_s/2$时刻完成的命令位置发送，解决了报文丢失问题。通过实验1的对比实验，说明了ASA方法能够有效的解决丢包问题。在不同步的情况下，数控系统与伺服驱动器在每个插补周期内都会产生时间漂移，并且随着时间的推移，累积漂移甚至可能是几个周期。然而，有了ASA，CNC和伺服驱动器具有良好的同步性能。在MSTS的对比实验中，如果不受中断模块的限制，ASA同步误差比MSTS降低了4.182倍。另外，该算法完全不受实时系统的限制，不仅解决了丢包现象，而且精准加工。而且，采用多次(例如500次)测量的平均值所消耗的时间，ASA几乎比MSTS减少了一半。实验3的结果表明，该算法适用于不同品牌或厂家的伺服驱动器。实验4中，在不同的插补周期500μs、1ms和2ms下，本发明中提出的ASA方法均表现出良好的同步性能，平均同步精度可达32-71ns。重要的是，ASA将EtherCAT提升到了一个全新的水平，完全实现了EtherCAT主站不再依赖RTOS。它超越了以往学者的研究。而且，该方法基于一个简单的等差数列求和公式，以几乎不增加计算量的方式，便获得精确的预补偿$\Delta T_{pre\text{-}compensation}$。只需在FPRD命令中添加两个或四个字节，几乎不会增加通信负载。另外，CNC主站发送命令位置的时刻和参考伺服接收命令位置的时刻具有共同的参考t_{sync0}，消除了CNC与参考伺服相对独立的绝对时间产生的累积误差。随着处理器频率的提高，同步精度能够提高为1ns。因此，ASA方法可以有效地解决无操作系统的嵌入式数控系统和伺服驱动器的时间(节拍)同步问题。该方法也可用于分布式多主系统的时间同步通信。

[0146] 以上所述，仅为本发明较优的具体的实施方式，但本发明的保护范围并不局限于此，任何熟悉本技术领域的技术人员在本发明揭露的技术范围内，凡在本发明的精神和原则之内所作的任何修改、等同替换和改进等，都应涵盖在本发明的保护范围之内。

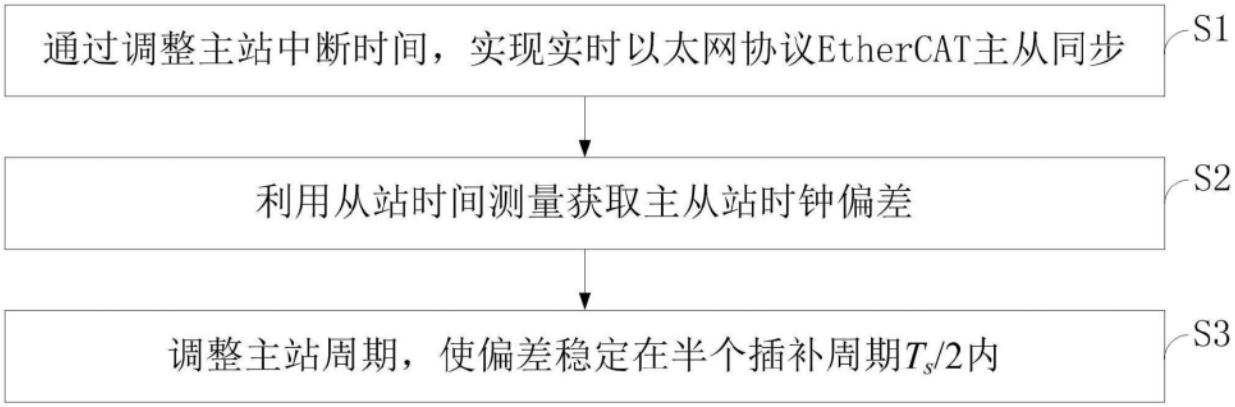


图1

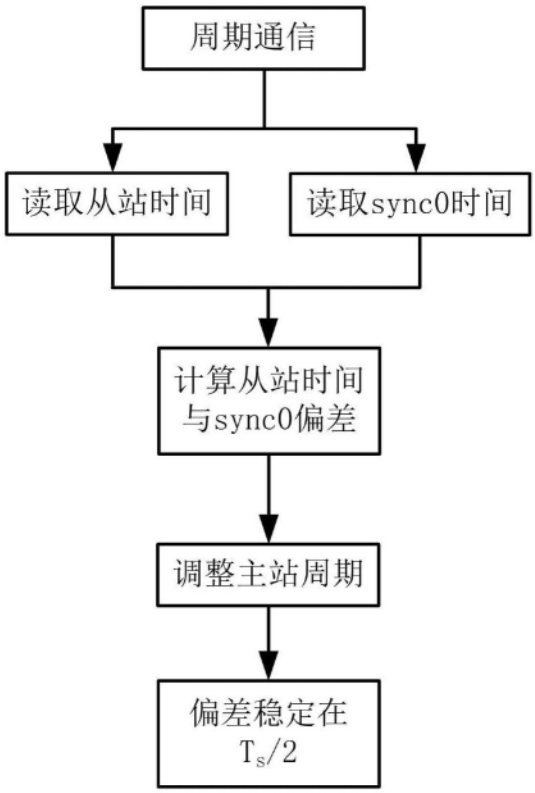


图2

以太网帧头			EtherCAT帧头	EtherCAT PDU								以太网帧尾
目的MAC地址 -6字节	源MAC地址 -6字节	以太网类型 -2字节	协议类型和PDU长度-2 -1字节	CMD -1字节	IDX -1字节	ADP -2字节	ADO -2字节	数据长度 -1字节	IRQ -2字节	Data -n字节	WKC -2字节	FCS -4字节

图3

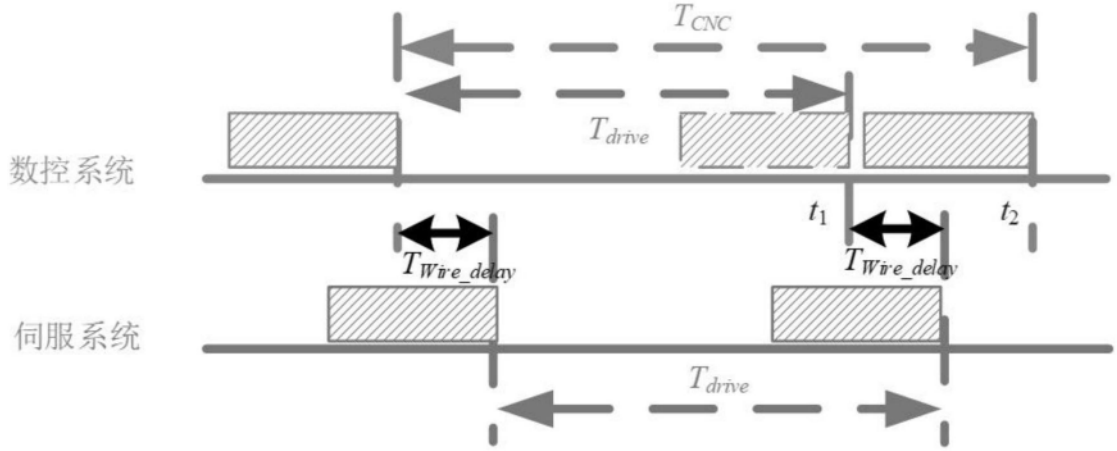

T_{CNC}
T_{drive}
数控系统
t_1
t_2
T_{Wire_delay}
T_{Wire_delay}
伺服系统
T_{drive}

图4

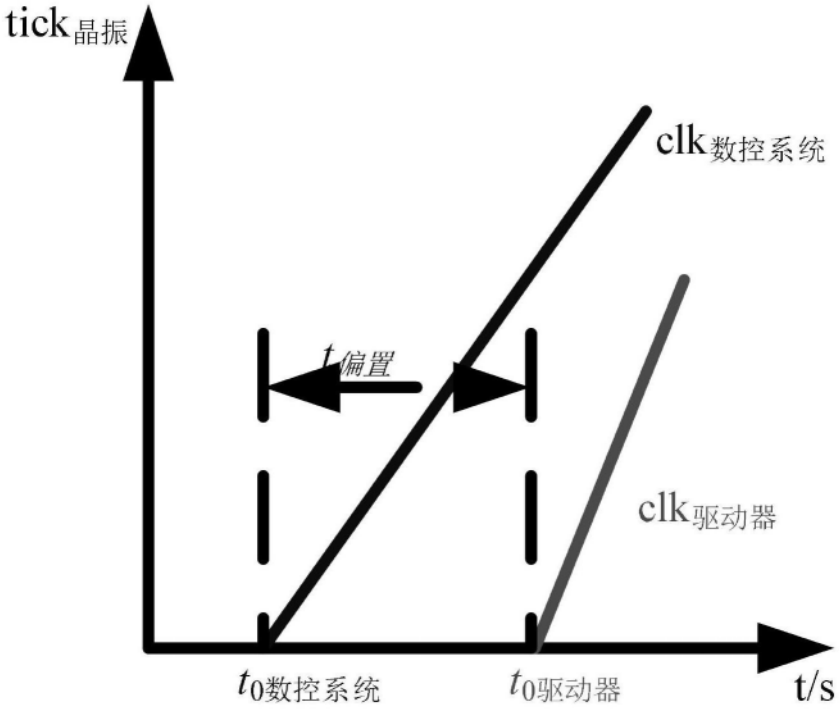

tick晶振
clk数控系统
$t_{偏置}$
clk驱动器
$t_{0数控系统}$
$t_{0驱动器}$
t/s

图5

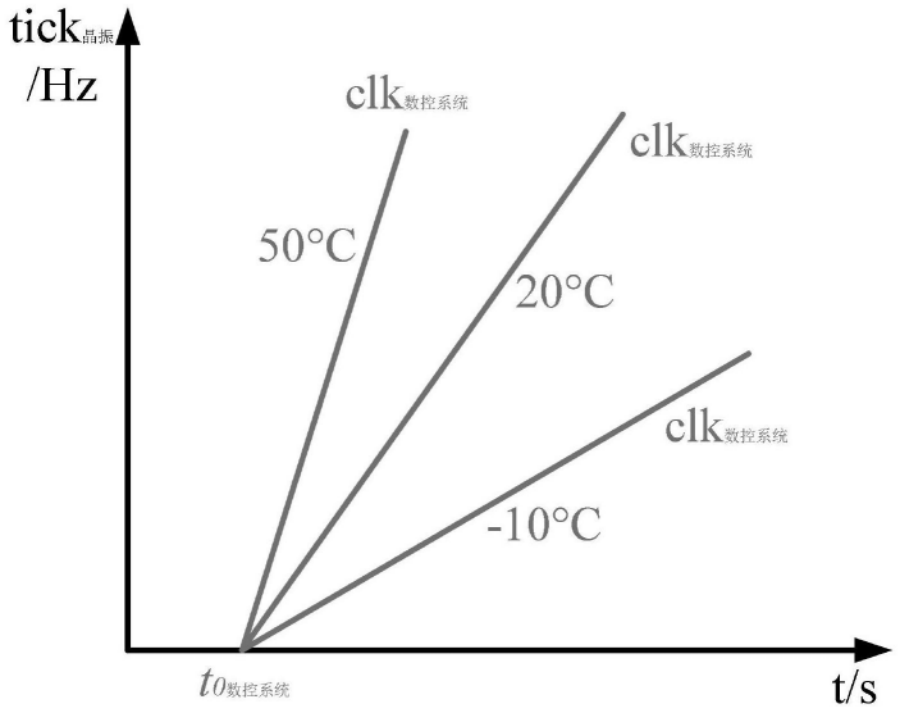

tick晶振
/Hz
clk数控系统
clk数控系统
clk数控系统
50°C
20°C
-10°C
t0数控系统
t/s

图6


IO状态时间
数控计算开销
帧开销
网线延时
调整时间
数控系统
伺服延时
伺服系统

图7

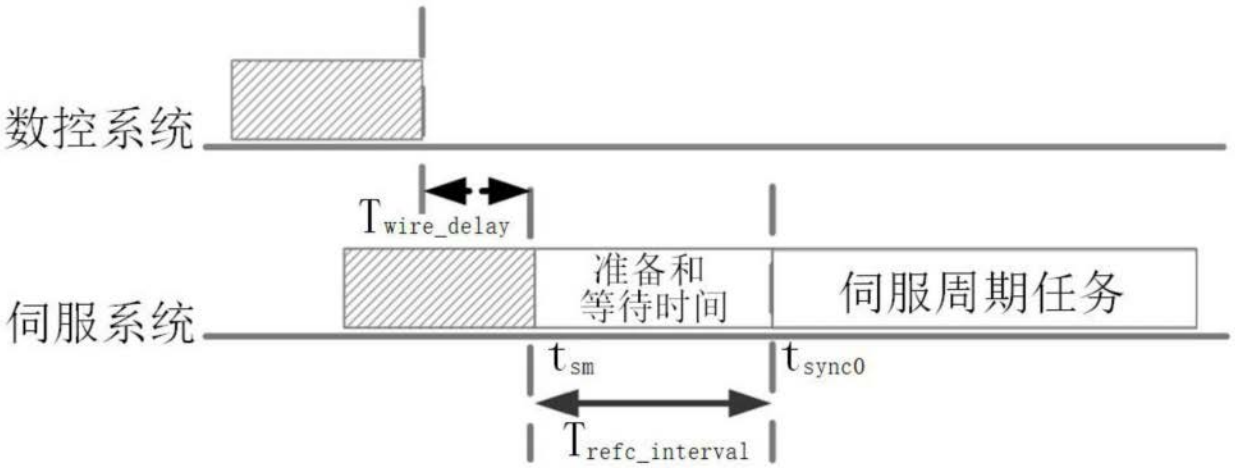

数控系统
Twire_delay
伺服系统
准备和
等待时间
伺服周期任务
tsm
tsync0
Trefc_interval

图8

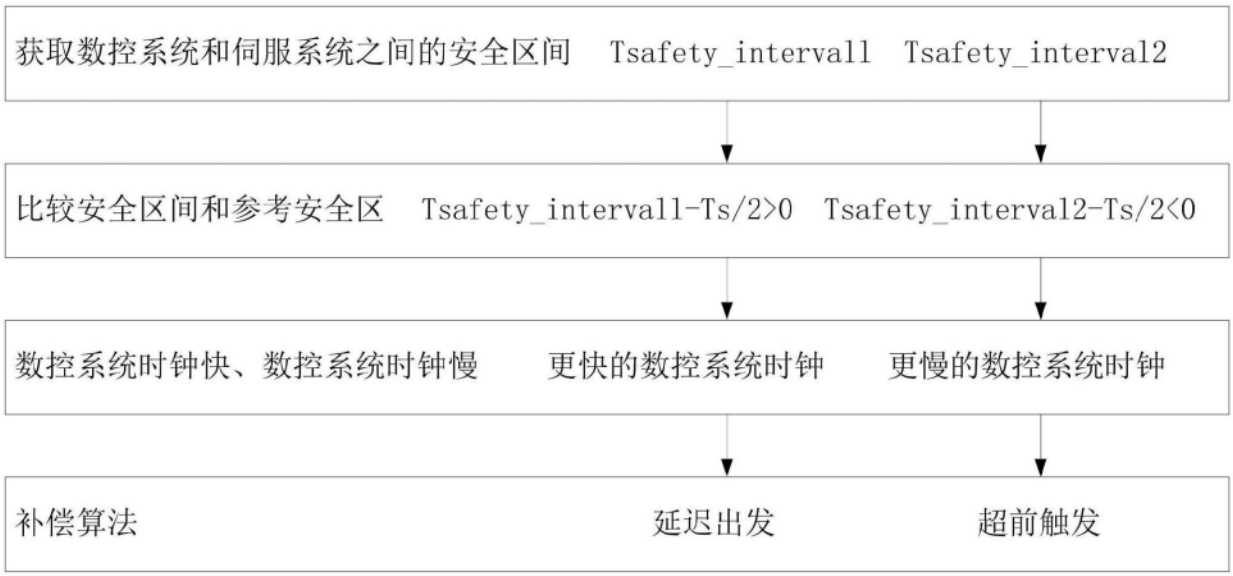

获取数控系统和伺服系统之间的安全区间 Tsafety_intervall Tsafety_interval2
比较安全区间和参考安全区 Tsafety_intervall-Ts/2>0 Tsafety_interval2-Ts/2<0
数控系统时钟快、数控系统时钟慢 更快的数控系统时钟 更慢的数控系统时钟
补偿算法 延迟出发 超前触发

图9

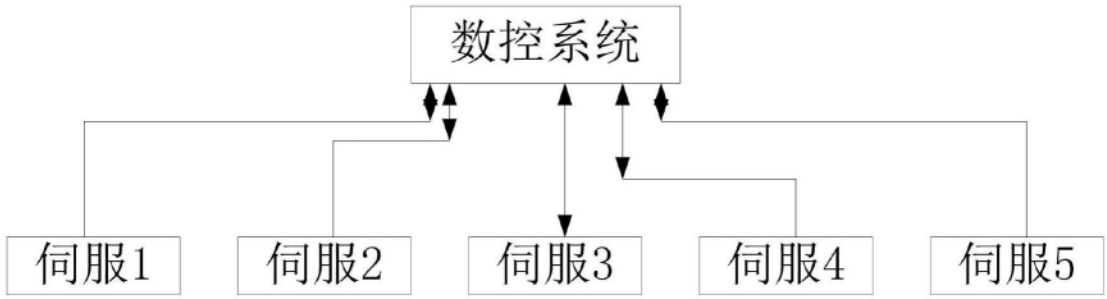

数控系统
伺服1
伺服2
伺服3
伺服4
伺服5

图10

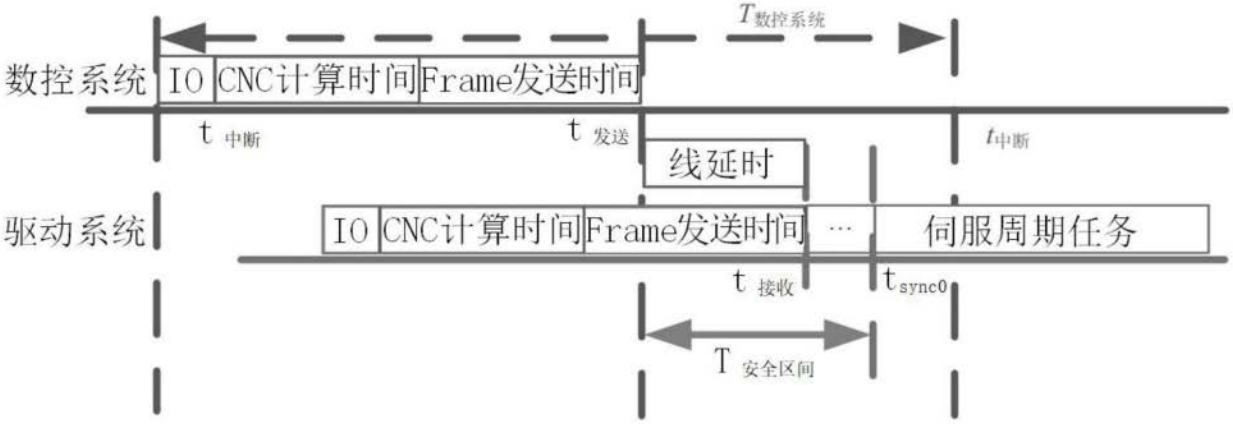

T数控系统
数控系统
IO
CNC计算时间
Frame发送时间
t 中断
t 发送
t中断
线延时
驱动系统
IO
CNC计算时间
Frame发送时间
…
伺服周期任务
t 接收
tsync0
T 安全区间

图11

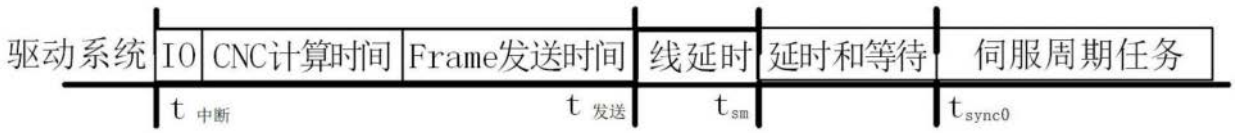

驱动系统
IO
CNC计算时间
Frame发送时间
线延时
延时和等待
伺服周期任务
t 中断
t 发送
t sm
t sync0

图12

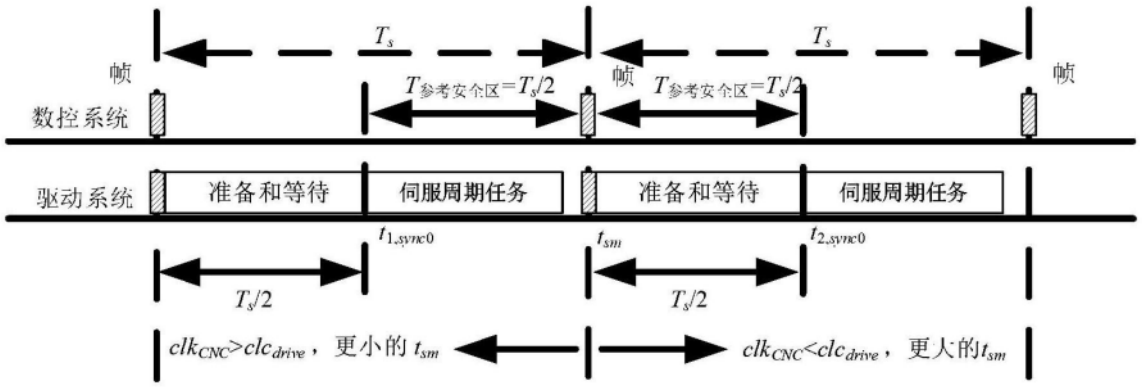

帧
帧
帧
数控系统
驱动系统
准备和等待
伺服周期任务
准备和等待
伺服周期任务
clkCNC>clcdrive，更小的 tsm
clkCNC<clcdrive，更大的tsm

图13