高 级 篇

第 6 章 “步调, 一致”

主机心跳时间与设备时间同步

6.1 “时间”

目前人类所能认识的生存空间是一个四维空间，除了三维的位置空间外还有一个单向的时域空间，宇宙中万事万物生息演变无不烙上时间的印记，人类发展史中不同地域不同民族根据日月星辰的规则运动衍生出多种计时方法，到目前应用最广泛的是格林威治时间，通用的计时单位有年、月、日、时、分、秒、毫秒、微秒.....等，各个国家根据首都或重要城市的位置加上一个固定时域差将其转化为本地时，如：北京时间＝格林威治时间＋8 小时。

TTCANopen 系统时间使用格林威治时间作为系统的计时基准，但准许各国应用者在无跨时域时将其用本地时取代，也准许用户系统自行定义时间起点。

根据 CAN 总线通讯的特点和大多数应用对时间分辨精度的需求，TTCANopen 系统时间的分辨精度被规定为 0.1ms，也就是 TTCANopen 系统时间的基本计时单位为 0.1ms，系统使用一个 32 位的无符号整数标记一天之中的 0.1ms 的个数（系统寄存器空间），取值范围 0～863999999；系统使用一个 32 位的无符号整数标记天数（设备配置参数空间），并规定 2000 年为 TTCANopen 系统时间的“元年”，即 2000 年的 1 月 1 日为 TTCANopen 系统时间记时的第一天。

由此，我们可以推算出 2001 年 11 月 12 日 10 时，是 TTCANopen 系统时间的 982 天第 360000000 个计时单位（0.1ms）；2015 年 1 月 1 日 0 时，是 TTCANopen 系统时间的 5480 天第 0 个计时单位（0.1ms）；

6.2 “同步”

传统的工业控制过程一般都是由事件和指令驱动的，而对于有较为复杂时序要求的系统则显得有些力不从心。虽然我们可以巧妙的使用一些广播指令完成多个设备同时动作，但如

果我们需要在完成一个动作后间隔多少毫秒再完成下一个动作，又或在某个准确时刻设备执行某一动作等，就有了一定的难度，因为指令的发送和接收是需要时间的，总线上会有竞争等等，都不能保证我们按预定时序或时刻完成规定的动作。在 CANopen 中使用了同步计数指令，也就是在总线上用高优先级发送同步计数指令，为系统提供同步基准，设备根据要求，以同步计数为基准执行动作，这样做所付出的代价是总线效率，高频度的发送时标指令对总线带宽的消耗是不可小视的。

为了解决工作时序问题 TTCANopen 采用了另外的方法。

这是一个痛苦的抉择，我们自认为解决的比较充分，但付出的代价仍然是昂贵的，我们需要网络中的所有设备具备“时钟”功能，这些“时钟”被总线上的“主时钟”同步，这样系统便有了一个“统一的时钟”即“TTCANopen 系统时间”，这就是时序工作的时间基准。

6.2.1 主时钟

通常是由专用时钟发生器（心跳器）为系统提供主时钟，这里的主时钟实际就是所谓“主机心跳”，根据系统的需求，我们会每隔一段时间，发送一次主机心跳，这个间隔（默认）一般是 1 秒钟，网络上所有的设备都会在收到“主机心跳”后，完成设备的对时工作，使“设备时钟”与“主时钟”同步。

一切看似顺理成章，天衣无缝。可仔细想想却不尽然，通常网络上的设备是收到“主机心跳”后在 CAN 的接收中断服务程序中完成“对时”的，为了传递较为准确的时间信息给网络上的设备，我们指定“主机心跳”指令的传输结束点为“计时时刻”（这里我们忽略线路时延和设备接收中断处理时延），这样“主机心跳”的发送起始时刻就应该比整秒时刻提前一个指令传输的时间，这个提前量与指令的传输速率和长度直接相关，好在这两个数值我们可以直接获取和计算得到。这里需要提醒的是由于“插位”的存在，每次的“主机心跳”实际发送的序列“位”个数并不尽相同，这给发送“主机心跳”时刻带来了一定的困难，我们总不能每次心跳都要计算帧长，然后规划起跳时刻，这是不能接受的。我们希望“主机心跳”指令是定长的，甚至是简洁的，于是邓先生提出了一个分两步走的方案，即先发送一条“预报帧”，预告下面即将发送的“心跳帧”的完成时刻，然后再发送这条固定的“心跳帧”，这样总线上的设备收到“心跳帧”后，立即将设备时间与预报帧中的时间内容进行同步，完成设备时与主时钟的同步过程。

主机心跳序列：

预报帧　0x0 0x0010[1] 0x01 0x19 0x08 0x00 0x2A 0x75 0x15 0x00 0x00 0xE8 0x03

心跳帧　0x0 0x0011[2] 0x01 0x19 0x00

预报帧指令中使用了“系统寄存器空间”0x0010[1]和 0x0011[2]（关于“系统寄存器空间”的使用我们将在“生产者－消费者”章节中进行阐述）；地址 0x01 是发送主时钟的“心跳器”地址；预报帧前 4 个字节数据为一个 32 位无符号整数，表述 TTCANopen 系统时 0.1ms 的计数（计数范围：0～863999999），接下来 2 个字节为 2 个 8 位无符号整数，分别表述“心跳器”CAN 端口接收和发送错误计数，最后 2 个字节数据为一个 16 位无符号整数，表述主机心跳间隔，单位 ms（默认为 1000ms），该值应当能够被 86400000 整除。

心跳帧是一个固定的不携带数据的数据帧。

在 TTCANopen 之 TC0066（主机心跳时间与设备时间同步）子协议中规定了 TTCANopen 系统时间的格式、主机心跳指令格式，但没有给出实现的工程方法，这需要用户去实现。“心跳器”要将标准的格林威治时间转换为 TTCANopen 系统时间，提前 4 个最长帧的时间发送“预报帧”，然后提前一个“心跳帧”的时间发送心跳帧，默认在整秒时刻将心跳帧发送完成；网络上的设备收到心跳帧立即在其接收中断中使用预收的系统时间完成设备时间同步。为了防止发送心跳帧时总线被占用，总线上的所有设备在收到“预报帧”后，都应该规避对总线的发送占用，直到收到心跳帧或超时后，重新恢复对总线的发送占用；另外“心跳器”发送“心跳帧”是一次性的，禁止重发，一旦遇到总线被占用的时候，取消本次发送，直到下个发送时刻。注意：以上对心跳帧的限制并不针对预报帧，预报帧所携带的数据只在一个心跳周期内有效。

6.2.2 设备时钟

这也是我们感到最痛苦的地方，因为每一个入网的设备都需要有一个能够与主时钟同步的“设备时钟”，这无形中提升了设备和整个系统的成本，乃至于基于 TTCANopen 协议的使用成本，这是我们非常不愿意看到的，然而确是“不得不”。北京中铁航王剑宇总经理说：“手机从单纯的通话设备发展到现在成为多功能的个人终端，设备资源提升是必然的，随着微电子技术的发展其使用成本确是在下降的。”的确，随着 32 位单片机的普及，对于“设备钟”以及后面将要提到“类”的应用所带来的成本压力已经显得并不那么沉重，我们最初的担心似乎有些多余。

和“主时钟”一样，“设备时钟”本质上是一个 0.1 毫秒分辨率的计数器，设备在接收到“心跳帧”的中断服务程序里，完成与主时钟的同步调整。在我们研制的亚当模块中，这种同步调整误差稳定在±0.01 毫秒之内。

有了统一的时钟，设备的时序动作就有了时间“基准”，更可以合理的安排那些周期上报信息的发送时序，均衡总线负荷。

6.3 “子帧相位”和“副帧相位”

网络设备有了统一的时间系统，我们就可以整体规划设备心跳和周期上报指令在时间轴上的分布了。我们制定一个整数常量 M（32 位无符号整数），称之为“子帧循环计数最大值”，或称为子帧周期，该值可以被 864000000 整除。系统时间与其求余加 1 后的结果，就是一个不断重复循环的 1～M 的计数，称之为“子帧循环计数”。我们再定义整数常量 N（32 位无符号整数），称之为“子帧相位”，取值范围 1～M。这样我们就可以根据上报指令的多少，在 1～M 间划分 K 个子帧相位，并为每一个设备心跳或周期上报指令分配一个不同的“子帧相位”值，当“子帧循环计数”等于某“子帧相位”值时，对应的设备心跳或周期上报指令便会被触发上传。（提示：“子帧相位”值的选择，应当尽量均匀分布在 1～M 值之间，并避让“主机心跳”时段。相邻的“子帧相位”之间应当可以容纳传输一条以上的 CAN 指令时间。）

然而，并不是每一个设备心跳和上报指令都需要在一个子帧循环计数中发送完成，这对节省总线带宽不利，有些慢变量可以有更长的发送周期。这样我们定义一个整数 W（32 位无符号整数），称之为“副帧计数”，每当“子帧循环计数”循环一周该值加 1。我们再定义一个整数常量 P（16 位无符号整数，可被 W 的上限＋1 整除），称之为“副帧循环计数最大值”，或称为副帧周期，“副帧计数”与其求余加 1 后的结果，就是一个不断重复循环的 1～P 的计数称之为“副帧循环计数”。我们定义整数常量 Q（16 位无符号整数），称之为“副帧相位”，取值范围 1～P。这样我们就可以在为慢变量分配“子帧相位”的同时，再分配一个“副帧相位”，只有当“子帧循环计数”等于其“子帧相位”，且“副帧循环计数”等于其“副帧相位”时，该慢变量上传指令才会被触发。

有了“子帧相位”和“副帧相位”，我们就可以合理的分配各种周期上传指令的发送位置了。“子帧循环计数”一个循环称为“子帧周期”；“副帧循环计数”一个循环称为“副帧周期”；“子帧”和“副帧”复合完成一次循环，称为一个“全帧”见图 6－1。图中我们

称每一个纵列为一个“波道”。我们可以选择其中一列或几列，作为传输慢变量的波道。

读者很容易就会由此联想到我们在纯“主从模型”中提到的主机“轮询矩阵”，这里我们也给“子帧相位”和“副帧相位”的分配方案起一个名称，叫做“TTCANopen 网络周期触发时间相位矩阵”，简称“时间相位矩阵”。

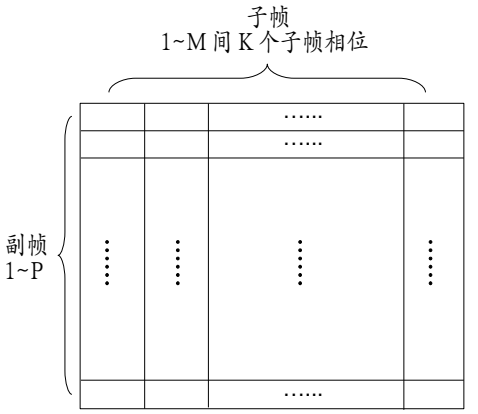

图 6－1 时间相位矩阵

“轮询矩阵”和“时间相位矩阵”的功用有类同，都是控制网络上各设备执行可预见周期性指令。但“时间相位矩阵”比“轮询矩阵”节省一半以上的总线带宽，由于是时间驱动，其指令执行的时刻和周期是精准的。而“轮询矩阵”受主机指令调度稳定性和发送指令在总线上竞争传输的影响是无法获得精准的指令执行时刻和周期的。

注：根据系统需求，TTCANopen 可以支持多组“副帧周期”，为系统同时提供多种副帧服务。

6.4 “时间相位矩阵”设计举例

在一个 TTCANopen 网络中有 20 个设备，每个设备都有 10 个周期上报指令，上报周期为每秒一次。整个网络每秒只发送一个“设备心跳”，网络通讯速率为 100kbps。

根据上述条件，我们可以知道上报指令周期为 1 秒钟，转化为 TTCANopen 时间是 10000（0.1ms 计数），也就是一个“子帧周期”。在一个“子帧周期”中上报的指令条数为 200＋1 个，也就是要划分出 201 个“子帧相位”。在 1～10000 循环计数的高端由于有“主机心跳”需要避让，在这里我们避让 5 个最长帧的传输时间；在低端由于“主机心跳”对其他指令的禁发抑制，可能会有指令在“主机心跳”后竞争发送，因此我们在低端同样避让 5 个

最长帧的传输时间；通讯速率为 100kbps，一个最长帧按 160 位计算，其通讯使用时长为 16（0.1ms 计数），我们需要在 1～10000 子帧周期的高端和低端各避让 16＊5＝80（0.1ms 计数），因此我们可划分的“子帧相位”空间为 80～9920。相位间隔＝（9920－80）/201≈49。这样我们从 80 开始每间隔 49 设定一个子帧相位用于发送 201 个周期上报指令。

我们可以在上述 201 个子帧相位上，任选一个作为发送设备心跳的副帧波道，副帧周期为 20。在这个波道上每副帧相位（1～20）上安排一个设备心跳。

6.5 准 TTCAN[3] 的实现

如果 TTCANopen 网络中没有随机触发的指令，所有指令都是在预定时间发送的，那么我们就可以通过“时间相位矩阵”的排布，在 TTCANopen 中实现类 TTCAN 的功能。我们称之为“准 TTCAN”。在“准 TTCAN”中，所有设备都应限制其指令重发功能，防止指令重发扰乱整体发送时序。

6.6 讨论与提示

6.4.1 “预报帧”的提前量

“预报帧”一定要在“心跳帧”之前完成发送，理论上“预报帧”需提前两帧发送，但实际上在“预报帧”准备发送时总线可能已经被占用，“预报帧”只能等这帧发送完成后，再进行发送。另一方面，设备是在接收中断中处理“预报帧”的，设备需要停止总线发送占用，但很有可能此时设备已经在发送数据，只有等此帧发送完成后，才能禁止下一帧的发送。因此在最坏的情况下“预报帧”需要提前 4 帧的时间进行发送，才能使“心跳帧”按时发出，这里还不包括总线出错的情况，因此系统并不能保证“心跳帧”总是能够按时发出。还好时间信息是一种连续性信息，设备时钟具有一定的“守时”特性，因此偶然丢失的“心跳帧”对系统时间的影响并不大。

对于普通设备可以在收到“预报帧”后，执行停止对总线的占用；对于与主时钟同步的设备而言，是可以预知“心跳帧”到来的时刻，因此也能够不依赖“预报帧”控制停止对总线的占用行为。

6.4.2 设备振荡器

设备时钟的精度与驱动设备振荡器输出的频率稳定度和准确度直接相关，由于TTCANopen 系统时间分辨精度为 0.1ms，时间同步间隔默认为 1s，因此对设备振荡器的频率稳定度和准确度要求应当优于 10^{-5}，一般的 LC 振荡器已经不能满足要求，因此设备通常需要配置合格的晶体振荡器。

6.4.3 时钟发生器（心跳器）

我们强烈建议用户使用专用时钟发生器为系统提供主心跳，这种时钟发生器的时源可以来自卫星定位系统，如：北斗或 GPS，也可以来自标准的时码站。

专用时钟发生器一般都要求配备高稳定的晶体振荡器，在对绝对时要求不高的系统中，我们只需定期与广播电台对时，然后由专用时钟发生器守时即可为系统提供所需时间。

需要指出的是“主机心跳”只提供了一天 24 小时的 0.1ms 的时间计数，设备的天数则是由设备管理器通过广播指令发送至网络上每一个设备中的，设备天计数被定义在设备配置参数空间 0xF030～ 0xF033 上。

6.4.4 “主机心跳”的冗余设计

“主机心跳”其实更准确的讲应该叫做“系统时钟心跳”，由于习惯的原因，与“设备心跳”相对应，我们更喜欢管它叫“主机心跳”。

“主机心跳”对于 TTCANopen 系统是至关重要的，一旦“主机心跳”出现了问题或损坏，整个系统将失去同步时钟，也就意味着整个系统将是不可用的，因此在一些重要的应用环境中，我们必须要在总线上提供“主机心跳”的冗余设计。

最简单的方法就是在总线上接入多个“心跳器”，其中一个为工作机，其它为备机。为了区分是哪一个“心跳器”在工作，这些“心跳器”应该有自己独特的设备地址，为此我们刻意的分配 0x01、0x02 和 0x03 为“心跳器”专用地址。

当总线上的“心跳器”收到其他“心跳器”发送的“主机心跳”时，会自动将本机设置为备份机，当“心跳器”在设定的 n 个心跳周期内没有收到“主机心跳”，就会自动将本机设置为工作机向总线发送“主机心跳”。

6.4.5 “设备时钟同步”

在 TTCANopen 网络中的设备需要将自己本地时钟与系统时钟进行同步，这种“同步”

过程是通过接收“主机心跳”来完成的，通常设备只要接收到一个“主机心跳”就可以完成设备本地时钟与“系统时钟”的同步，但由于设备本地时钟存在钟差与稳定度的问题，这种与“主机心跳”的同步必然是周期性的持续过程。

理想的“主机心跳”是不会出错的，设备与“主机心跳”的同步也是非常简单的，然而我们的确不能保证“主机心跳”不出现“野值”。这种“野值”包含两层含义，一是心跳预报值出现错误产生“野值”；二是主机心跳时刻出现了偏差，提前或滞后。因此设备需要有能力剔除这些“野值”，使设备始终与真实的“系统时间”同步，不因“野值”的存在而产生偏差。幸运的是，“时钟”是一个稳定的可持续推演的变量，因此我们可以通过这种持续推演剔除超差的“野值”，使设备本地钟始终与在合理误差范围内的“主机心跳”进行同步。在剔除“野值”的过程中，缺失的“主机心跳”同步是靠设备自身的“守时”稳定性来保障的，一般来讲“野值”是偶发的，是容易被“剔除”的。

我们为广大读者提供的测试验证设备没有提供这种剔除“野值”的功能，是为了让系统能够暴露出更多问题。虽然如此，在我们长期的测试过程中，始终没有捕获到“野值”的出现，这可能得益于我们的测试环境还不够恶劣，以及 CAN 协议传输数据的可靠性。在“心跳器”中，我们是在时间中断中发送系统“主机心跳”的，且在发送前对总线状态进行了检测，一旦总线被占用便停止本次主机心跳，因此在系统测试时，我们可以测试到“主机心跳”的缺席，但从未检测到提前或滞后的现象（误差在 0.02ms 内）。尽管如此我们还是强烈建议用户在工程应用中，充分利用系统时间稳定可推演的持续性，对时间系统进行必要的“滤波”处理。

6.4.6 高精准的时间同步

在某些高端应用中，单靠特殊的 CAN 指令传递时间信息，远远不能满足时间精准度的要求，于是我们不得不增加一定的硬件设施解决这一难题，通常的做法是增加一对双绞线传递时间信息，可以是 B000 码，也可以是秒脉冲等。

这种靠增加硬件提供时间系统的方法，目前不在我们讨论的范围内。

6.4.7 “超采”与“漏采”

在本书的第三章关于轮询矩阵曾经提到过这两个词，但在这里所指的含义却完全不同，在第三章这两个词蕴含着负能量，而在本章则充满了正能量。

“超采”是指在一个“子帧周期内”可选择多个“子帧相位”发送同一过程变量，这样我们就可以在不提高子帧周期的情况下，对快变过程变量提供高于子帧周期的采集频度。

“漏采”是指当某时间相位触发时，发现其变量和上一周期触发时相同，则停止该指令的发送，可有效的节省总线带宽。

6.4.8 简易“心跳器”

为了降低用户的测试成本，我们利用自研的 USB 转 CAN 模块改制了一款简易“心跳器”，详见第七章。

6.4.9 TTCANopen 时间系统是不抗攻击的

虽然我们可以利用系统时间稳定可推演的持续性，对时间系统进行必要的“滤波”处理，以剔除“野值”，但它却不能抵抗来自网络上不良设备的“恶意攻击”。不良设备不需要了解系统设备的指令状况，只需不断的发送“伪心跳”，就可瘫痪整个系统。因此 TTCANopen 组织一开始就郑重声明：TTCANopen 不适用在矿山机械、银行、国防军工以及一切涉及到人身安全、重大财产安全和国家军事安全的领域。如若使用，用户必须对存在的系统风险做全面的评估。

注：［1］在 TTCANopen 引入双 ID 后，主机心跳预报帧寄存器地址改为 0x0A02。
［2］在 TTCANopen 引入双 ID 后，主机心跳心跳帧寄存器地址改为 0x0A03。
［3］TTCAN 由 Bosch 开发。