文章编号:1671-4598(2020)06-0243-04 DOI:10.16526/j.cnki.11—4762/tp.2020.06.049 中图分类号:TP391 文献标识码:A

基于 Windows 平台的实时信息处理方法

段红亮[1,2], 刘天博[1], 邵春收[1], 王 鹏[1], 朱元元[1]
(1. 北京航天长征飞行器研究所, 北京 100076;
2. 西北工业大学 机电学院, 西安 710072)

摘要: 微软的 Windows 操作系统由于具备良好通用性、图形用户界面以及众多的技术支持基础而成为测试设备的首选解决方案, 但基于抢占式多任务调度策略就决定了 Windows 系统的非实时性属性, 对于实时性要求较高、时序控制要求严格的武器系统测试, Windows 系统不适于作为地面测试使用; 针对基于 Windows 平台武器地面测试设备存在的非实时性问题, 提出了一种排他性线程独占技术和高精度软时钟技术, 可以实时处理以太网、串口等 IO 信息, 低成本且低复杂度地解决武器地面测试设备的实时性问题, 进一步保障武器系统测试的准确性和可靠性。

关键词: Windows 平台; 测试设备; 实时性; 排他性线程; 软时钟

Real—time Information Processing Method Based on Windows Platform

Duan Hongliang[1,2], Liu Tianbo[1], Shao Chunshou[1], Wang Peng[1], Zhu Yuanyuan[1]
(1. Beijing Institute of Long March Vehicle, Beijing 100076, China;
2. School of Mechanical Engineering, Northwestern Polytechnical University, Xi' an 710072, China)

Abstract: As good versatility, graphical user interface and numerous technical supporting foundations, Microsoft Windows Operating System has become the preferred solution for ground test equipment of weapon. However, the preemptive multi—task scheduling strategy determines the non—real—time property of Windows system, which is not suitable for ground test of weapon system with strong real—time performance. Aiming at the non—real—time problem of weapon ground test equipment based on Windows platform, this paper presents an exclusive thread exclusive technology and high precision soft clock technology, which can processes I/O information in real time such as Ethernet and serial port and et al. This method can realize the real—time problem of ground test equipment with low cost and complexity.

Keywords: Windows platform; testing equipment; real—time; exclusive threads; soft clock

0 引言

大部分武器系统设备都是强实时性的, 设备之间信息交互都有严格的时间限制, 要求毫秒级或微秒级, 甚至是纳秒级, 所以主流的弹上设备均基于 DSP、FPGA、单片机等架构或者是组合架构实现, 也有少部分采用实时操作系统平台。地面测试设备用于武器设备的开发测试、出厂测试、上弹总装前后或者年度定检等等一系列测试工作, 是所有武器必配设备, 地面测试设备是武器设备的“诊断医生”, 只有经过地面测试设备认证为“健康”的弹上设备才允许使用, 地面测试设备的研制开发甚至要早于弹上设备。

地面测试设备与弹上设备进行通信和指令交互, 完成功能测试等。微软的 Windows 操作系统由于具备良好通用性、图形用户界面以及众多的技术支持基础而成为武器地面测试设备的首选解决方案。目前绝大部分型号的地面测试设备为了兼顾成本和开发难度等因素, 采用 Windows+x86 架构实现。但基于抢占式多任务调度策略就决定了 Windows 系统的非实时性属性, 作为通用高性能操作系统而设计的操作系统, 不能保证中断响应时间的确定性, 也没有提供让线程获得确定执行时间的机制。在测试时地面测试软件上偶尔会出现某条指令超时的现象, 这时就很难判断是弹上设备指令确实超时, 还是由于地面测试设备非实时性而造成的时间误判。

针对 Windows 平台的非实时性问题, 工业控制上一般采用 RTX 硬实时解决方案[1-2], RTX (Real—Time eXtension) 是由美国 IntervalZero 公司开发的基于 Windows 平台扩展的实时控制解决方案, 但需要单独购买 RTX 软件扩展包, 且 RTX 信息处理软件独立于 Windows 平台需求单独开发, 而武器地面测试设备由于生产量少、需求变化快等特点, 成本与周期上不适于采用 RTX[3]。文献 [4] 中提出了一种循环缓存处理方法来提高 Windows 系统的实时性, 这种用空间换时间的方法在操作系统负载较小时一定程度上提高了实时性, 但由于对缓存读写线程仍为操作系统的一个普通线程, 在操作系统负载较大时, 循环缓存处理方法的实时性不能得到保证。

文献 [5] 提出了一种 Windows 操作系统中的排他性线程独占技术, 使用该线程可以独占一个 CPU 逻辑核, 而不再受 Windows 任务管理器的调度, 物理上将该逻辑核等效 DSP 使用, 保证了信息处理的实时性。基于文献 [5-7]

收稿日期:2020-03-29; **修回日期**:2020-04-15。

作者简介:段红亮(1983-),男,河北邯郸人,博士生,工程师,主要从事雷达电子对抗、机电一体化设计方向的研究。

的研究成果，面向武器地面测试设备，本文提出了一种实用的实时信息处理方法，可以实时处理以太网、串口等有线或无线 I/O 信息，该方法开发难度低、周期短且不增加设备成本。

论文结构如下：第 1 部分简要分析了 Windows 非实时性机理，第 2 部分详细阐述高精度软时钟技术及实现方法，第 3 部分地面测试软件的实时测试方法，第 4 部分介绍对本方法的有效性试验分析，最后一部分对全文进行了总结。

1 Windows 非实时性机理

1.1 x86 的指令集

x86 的 GPP 架构本身不能实现纳秒级强实时性要求，这是通用 CPU 的 CISC 指令集决定的，有两个层次原因，首先是 CISC 指令集长短不一，所以占用 CPU 时钟周期不一；二是 Intel 在指令集中有很多优化算法，同一条汇编指令在同一个 CPU 的不同负载下可能会被编译成不同的汇编指令，目前 Intel 的 CPU 的时钟周期都是亚纳秒级，所以 x86 CPU 难以实现纳秒级的高精度定时，用于十纳秒级定时精度是普遍可以接受的。从目前的弹上设备所要求的毫秒级或微秒级实时性要求来看，x86 CPU 架构本身的定时精度是可以忽略不计。

1.2 调度原理

Windows 操作系统调度及响应时延的不确定性。以串口通信为例（并口、以太网口类似），地面测试设备与弹上设备的信息交互流程如图 1 所示，忽略弹上设备处理时延（或者假定为一个确定值）。

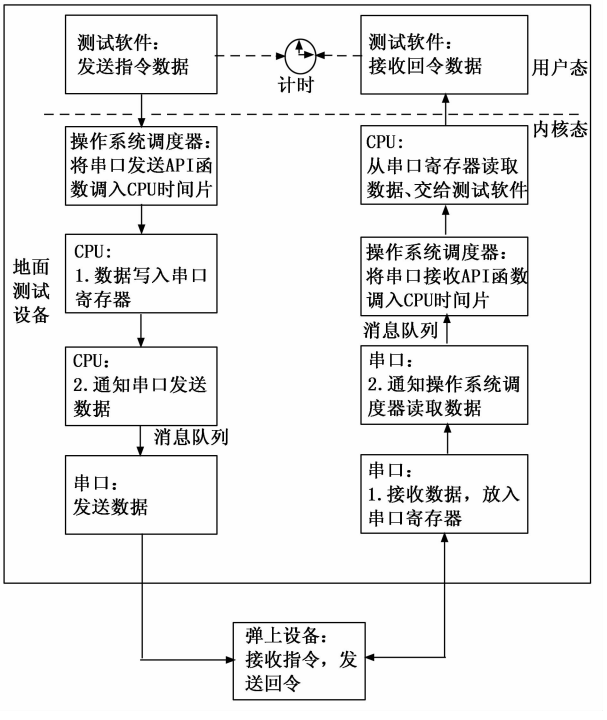

图 1 信息流示意图

例如，有条指令要求弹上设备在 100 ms 内回复指令，那么处理过程如下：

1）测试软件打开某个串口，或在软件初始化时即打开这个串口，并配置串口参数；

2）当有个指令发送时，测试软件调用 WinAPI 串口函数，发送该指令，同时启动计时器，设置 100 ms 的超时值；

3）数据由用户态进入内核态，操作系统调度器需要将串口发送 API 指令调入到 CPU 任务队列中，当占用 CPU 当前时间片时，CPU 执行该指令；

4）CPU 将发送数据写入到串口 Tx 寄存器中，同时通知串口有数据到达，请发送；

5）串口通过物理层链路发送给 DS 设备，DS 设备响应该指令后发送回令给串口；

6）串口接收回令数据，放入到 Rx 寄存器中，同时通知操作系统有数据到达，通知消息本身是放入到操作系统的消息队列中；

7）操作系统调度器在消息队列中读到串口的通知后，需要将串口接收 API 指令调入到 CPU 任务队列中，当占用 CPU 当前时间片时，CPU 执行该指令；

8）CPU 从串口 Rx 寄存器读取数据，交给测试软件，由内核态进入用户态；

9）测试软件根据数据协议解析该数据，确认为当前指令的有效回令后，停止计时器，同时判断计时器是否超 100 ms。至此一个完整的信息交互结束。

Windows 是基于消息响应机制的多线程的非实时性操作系统，尽管微软没有公开 Windows 操作系统内核，当通过一些公开资料和微软的技术文档可知，一个线程一旦调入到 CPU 中，那么最少占有一个时间片的时间长度才会有可能被调出，而一个时间片长为 1/16 s。

在 Windows 的任务管理器的进程栏可以看到，假如用户未打开任何一个应用程序，操作系统都会有约 100 个进程在运行，地面测试软件运行时，也只是其中之一个进程。这些进程都有不同的优先级，所以当一个进程从进入 CPU 调度队列到占用 CPU 时间片，这个时间是不可预知的，这是 Windows 非实时性的最主要根源所在。具体到我们的串口收发数据流程中，第 3）步和第 7）步是时延不确定的最主要环节。

1.3 试验验证

本文进行了一项测试，当地面测试设备只运行测试软件，其他所有应用软件都停止（包括杀毒软件），程序循环发送指令，等待 100 ms，若超时未收到回令则停止循环并报错，若正常则继续循环。测试 6 个小时左右，发生 2 次超时。

作为对比测试，在运行测试软件循环测试的同时，打开杀毒软件，进行全盘查杀病毒，测试 1 个小时左右，发生了 12 次超时。主要原因是杀毒时频繁的磁盘 IO 操作占

用 CPU 资源，杀毒软件进程也需要大量占用 CPU 时间片，从而提高了与地面测试软件的碰撞冲突概率，造成了大量的超时现象。

2　高精度软时钟

高精度时钟源是实现信息实时处理的前提和基准。例如通信设备上多采用外部时钟源或卫星授时方法，而在地面测试设备上外挂时钟源或增加卫星授时模块的方法会显著增加复杂度和成本，不易实现。

为了使各种实时处理任务能在 Windows 平台上正确地执行，利用操作系统 API 函数实现软时钟是可行的。然而 Windows 目前提供的延时函数最小时间粒度都是 ms 级，且理想情况下精度仅能达到百 ms 级，所以直接调用 API 函数也不能保证时间精度。

对于单核 CPU，Intel 汇编指令 RDTSC 可以获得纳秒级的精度[8]。然而，这项技术不再适用于多核环境[9]。为了构造高精度的软件时钟，本文采用了文献［5］所提出的排他性线程独占技术，它为输出实时行为的特定任务分配一个独占的处理器内核。一个线程被创建并专用于在一个独占的 CPU 逻辑核上执行。该线程所做的一切就是调用 RDTSC 来计算当前系统时间。由于线程是不可中断的，这种方法为内核模式的实时过程获得了一个高精度的时钟实用程序。图 2 为构造这种时钟的伪代码。

```
1  //Real-timer with nanosecond level
2  void Exclusive_Thread real_timer(core Num) {
3      start_tick = __rdstc();
4      freq = GetCPUFreq();
5      do {
6          end_tick = __rdstc();
7          time = (end_tick - start_tick)/freq;
8      } while(time < THRESHOLD);
9      //for example, THRESHOLD = 1 ms
10     NotifyOtherThreads(TIME_OUT);
11     return;
12 }
```

图 2　高精度时钟源伪代码

通过设置软件时钟，可以在每一个阈值时间段中执行一次 TIME _ OUT 事件。例如，将阈值定义为 1 ms 或 1 μs，在主频为 2.8 GHz 频率的 CPU 上，观察到实际的超时时间周期比阈值大几纳秒，这是合理的，因为在 CPU 中执行单个汇编指令所需的时间为纳秒级。所以本方法所生成的软时钟可以达到十纳秒级精度。

3　测试软件实时信息处理方法

3.1　测试软件实时性设计

为保证地面测试软件实时性，信息交互线程也采用排他性线程独占技术，独占一个 CPU 逻辑核，由于 Intel 的 i5 系列之后都是至少包含了 4 个物理核（单核双线程技术，等效为 8 个逻辑核），所以地面测试设备中被独占 2～3 个逻辑核后并不会显著操作系统的其他功能，如图 3 所示。

由图 3 可以看到，逻辑核 3、5、7 分别被 3 个线程独占，在运行中被独占的逻辑核的 CPU 使用率始终为 100%，不会被操作系统任务管理器中断，从而保证了实时性。

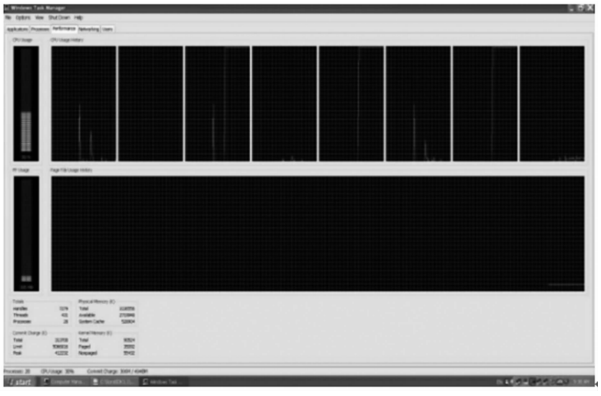

图 3　独占性线程的 CPU 使用率

信息交互线程在与弹上设备发生信息交互时，使用本文第 2 章所设计的高精度时钟源来作为时钟基准，可以精确计算出弹上设备回令是否超时。

3.2　测试软件高速计算设计

地面测试软件与弹上设备接口分为有线和无线接口两类，其中有线接口多为串口、并口、1553B 接口和以太网口等，无线接口多为各类专用无线通信接口，如遥测系统无线接口等。

一般来说，有线接口的计算量不大，若地面测试软件与弹上设备接口为无线接口，无线通信协议的计算量是非常巨大的，尤其是物理层，包含了信道编码与解调、FFT 等。一般来说，GPP 的运算能力是不能胜任基带数字信号处理计算的，这也是为什么到目前为止商用无线通信都是用专用基带芯片的原因。为了突破这个计算瓶颈，可充分借鉴 FPGA 的原理，地面测试软件可将所有输入与输出结果映射为一个查找表（LUT），然后将 LUT 装载到 CPU 的二级缓存中。一方面节省了计算的时间开销，另一方面减少了查找的时间[10-14]。在通用 CPU 中，不同存储器的典型容量和访问时间如图 4 所示。

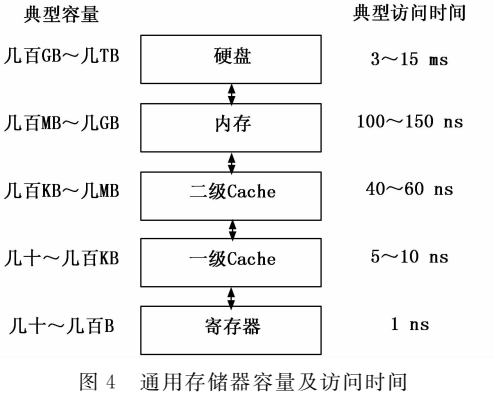

图 4　通用存储器容量及访问时间

3.3　测试软件低时延传输设计

由图 1 可以看到串口在 Windows 平台下的信息流，是

包含了多个环节。

在信息流中，每一个环节意味着数据的转移和传递，为了避免程度中内存拷贝所引入的处理时延，要在地面测试软件设计开发时尽可能多地采用数据“零”拷贝，即利用 MDL（memory descriptor list）来进行内存操作，这样可显著减小信息传输时延，避免造成地面测试设备向弹上设备回令时超时。

4 实验结果与分析

用两台地面测试设备搭建试验场景。为不失一般性，测试设备之间采用串口（RS422）通信，两台测试设备均是 Windows 平台，安装了采用了本文所述实时信息处理方法的软件。试验方法如下：

1）测试设备 1 通过串口向测试设备 2 发送一个串口数据帧 1，同时记录时间戳 T_1；

2）测试设备 2 收到数据帧 1 后，立即向测试设备 1 返回一个数据帧 2；

3）测试设备 1 收到数据帧 2 后，记录时间戳 T_2，计算出时间差 $\Delta T = T_2 - T_1$；

4）重复上述步骤 1）～3），记录 1 000 次时间差数据；

5）打开测试设备 1 上杀毒软件，执行全盘扫描，重复上述步骤 1）～4），记录1 000次时间差数据；

6）关闭测试设备 1 上杀毒软件，打开测试设备 2 上杀毒软件，执行全盘扫描，重复上述步骤 1）～4），记录 1 000次时间差数据；

7）同时打开测试设备 1 和测试设备 2 上杀毒软件，执行全盘扫描，重复上述步骤 1）～4），记录 1 000 次时间差数据。

4 次测试结果分别如图 5 所示。

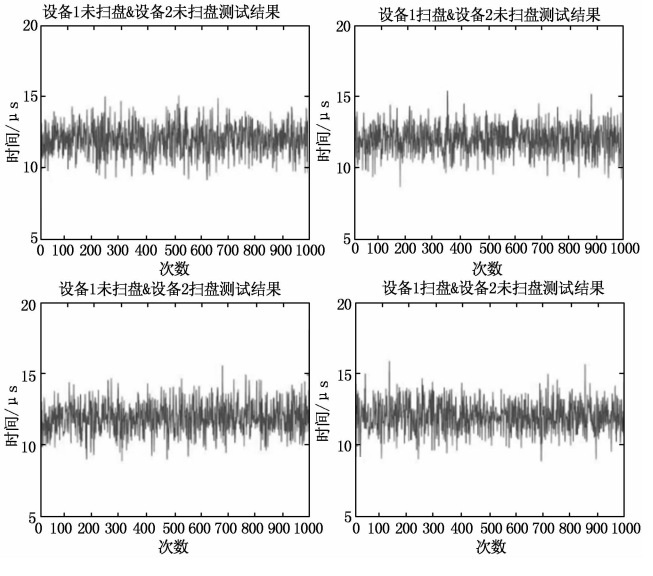

图 5 测试结果及对比

由测试结果分析，4 次测试均值依次为 12.2 μs，12.5 μs，12.1 μs，12.0 μs，相差不大。由图 5 可见，4 次结果均方差也分布在 2～4 之间，数据一致性较好，从而证明了本方法的有效性。

5 结束语

面向 Windows 平台下的地面测试设备，本文提出了一种基于排他性线程独占技术和高精度软时钟技术的实时信息处理方法，可以实时处理以太网、串口等有线或无线 I/O 信息，该方法开发难度低、周期短且不增加设备成本。首先简要分析了 Windows 非实时性机理，进而详细阐述高精度软时钟技术和地面测试软件的实时信息处理方法，最后对本方法的有效性试验验证分析，测试数据一致性较好。

参考文献：

[1] 刘 寰，秦现生，蒋明桔，等. 基于 Windows＋RTX 的 CNC 实时多任务调度设计 [J]. 测控技术，2012，31（3）：131－134.

[2] 李俊贤，李红宇. 基于 RTX 的实时通用测控软件设计与实现 [J]. 微电子学与计算机，2016，33（10）：120－124.

[3] 单 勇. 实时半实物仿真平台关键技术研究与实现 [D]. 长沙：国防科学技术大学，2010.

[4] 杜亚琦，周建英. 一种 Windows 系统中的实时信号处理方法 [J]. 电子设计工程，2018，26（13）：39－42.

[5] Tan K，Liu H，Zhang J，et al. “Sora：high performance software radio using general purpose multi－core processors [A]. Proc. ACM Commun.，2011，54（1）：99－107.

[6] 段红亮. 基于软件无线电的网络编码仿真平台研究与实现 [D]. 西安：西北工业大学，2013.

[7] Duan H，Huang D，Huang Y，et al. A time synchronization mechanism based on Software Defined Radio of general－purpose processor [A]. 7th International ICST Conference on Communications and Networking in China（CHINACOM）2012 [C]. IEEE，2012.

[8] Intel. IA－32 Intel architecture software developer’s manual，Volume 2A：Instruction Set Reference，A－M [Z]. Intel Inc.，2006.

[9] 陈 硕. 多核时代不宜再用 x86 的 RDTSC 指令测试指令周期和时间 [EB/OL]. http：//blog. csdn. net/solstice/article/details－－/5196544. 2010.

[10] 段红亮，黄登山，陈海华. 一种虚拟阵列扩展解相干的 DOA 算法 [J]. 计算机仿真，2013（01）：298－301.

[11] Beveridge J，Wiener R. Win32 多线程程序设计 [M]. 侯捷，译. 武汉：华中科技大学出版社，2001.

[12] 张 帆，史彩成. Windows 驱动开发技术详解 [M]. 北京：电子工业出版社，2008.

[13] 谭 文，杨 潇，邵坚磊. Windows 内核安全编程 [M]. 北京：电子工业出版社，2009.

[14] Richter J. Windows 核心编程 [M]. 第 4 版. 王建华，张焕生，侯丽坤，等译. 北京：机械工业出版社，2000.