

基于 Visual C++6.0 的 Windows 应用程序定时器研究

王鹏飞, 王 鹏

(中国空空导弹研究院, 河南 洛阳 471009)

摘 要: Visual C++6.0 是 Window 平台下最为流行的编程工具之一。在测控系统中, PC 机与各种嵌入式系统之间的接口控制也通常采用 VC6.0 完成。然而嵌入式系统通常对时序有较高的要求, Windows 系统则并非实时操作系统。因此如何在 VC6.0 中实现各种不同精度的定时功能就成为许多面向嵌入式系统的 VC6.0 开发关键环节。文中研究和比较了 VC6.0 平台下各种定时器的用法和性能。根据试验结果, 并针对某实际工程需要, 采用时间戳计数器实现了一种微秒级定时器。通过测试, 该定时器可以实现微秒级高精度定时, 与某嵌入式系统平台配合良好, 并通过了设备验收, 证明了其设计有效性。

关键词: 定时器; 精确定时; Windows 平台; 嵌入式系统

中图分类号: TN911.3

文献标识码: A

文章编号: 1673-629X(2013)02-0044-05

doi: 10.3969/j.issn.1673-629X.2013.02.011

Windows Platform Application Timer Research Based on Visual C++ 6.0

WANG Peng-fei, WANG Peng

(China Airborne Missile Academy, Luoyang 471009, China)

Abstract: Visual C++ 6.0 (VC 6.0) is one of the most popular development environments for Windows platform. Therefore, most of interface control of measurement & control system between embedded system and PC were also developed via VC6.0. However, since high-precision timing is often necessary for embedded system, and Window operation system (OS) isn't real-time OS. How to design timer to meet different embedded system requirement becomes a vital process for VC 6.0 development. In this paper, all kinds of timer design strategies were introduced and presented, following with their comparison. Finally, according to one project demand, based on CPU time stamp counter technique, a microsecond-level timer was presented. Whole project has been validated, therefore this times strategy's effect was also proved.

Key words: timer; precise timing; windows platform; high precision

0 引 言

随着信息和自动化测试技术的进步, 利用计算机实现的数字控制技术已经广泛应用于工业生产中。典型的测控系统通常由 PC 机、嵌入式系统板卡, 以及上位机程序组成, 而这些测控系统在采集数据、输出信号和刷新屏幕过程中都需要定时器, 尤其是有实时性要求的数据采集控制系统, 其对高精度定时器的要求就更为迫切^[1]。

VC6.0 作为微软公司于 1998 年推出的可视化编

程工具, 长期以来占据着 Windows 平台下主流开发环境的地位。虽然此后微软又推出 VC6.0 的更新版本, 如 VS2005、VS2008, 直至最新的 VS2011, 但 VC6.0 仍以其强大的生命力在 Windows 平台下占有重要席位^[2], 文中也正是在 VC6.0 下完成各种定时器的设计。

普通的 Windows 系统定时器每秒钟只能产生 18 个定时信号, 其定时精度仅为数十 ms 数量级^[3,4], 这种精度只可满足一般桌面应用程序的定时需求, 但在面向测控、嵌入式系统接口控制等领域时, 其弊端就会充分暴露, 而且随着时间变化, 这种定时方式带来的累积误差会急剧增大, 当定时中断要完成的任务较多时, 两者的差值将更大。从 Windows 的工作原理上分析, 其程序的运行机制实际上是消息循环机制, 如果某进

程已经占用了当前 CPU, 整个消息队列中的其他所有消息就不得不处于挂起状态, 进而使这些程序得不到及时响应, 最终导致定时精度的降低。因此, 有必要对 Windows 下不同精度的定时器设计进行研究。文中详细说明了 VC6.0 平台下毫秒级和微秒级定时器的实现方法及其误差范围。

1 常用定时方法

1.1 Windows 系统自带定时器

该方法主要是采用 SetTimer API 函数, 其原型及说明如下:

```
UINT_PTR SetTimer(  
    HWND hWnd, // 窗口句柄  
    UINT_PTR nIDEvent, // 定时器 ID  
    UINT uElapse, // 时间间隔, 单位为毫秒  
    TIMERPROC lpTimerFunc // 回调函数, 可留空  
);
```

使用示例: SetTimer(m_hWnd, 1, 500, NULL); 该函数调用即实现了一个每 0.5 秒触发一次的定时器。此外, 如果是在 MFC 程序中使用, 由于 SetTimer 已经被封装在 CWnd 类中, 调用时可不指定窗口句柄, 如 UINT SetTimer(1, 100, NULL) 即可。

该函数的返回值为此定时器的 ID 号。

使用 SetTimer() 申请定时器资源后, 其中断响应程序为 OnTimer() 函数, 用户只需要在该函数中加入处理代码即可。如果不再需要定时器, 用户可通过 KillTimer() 函数调用删除该定时器及占用资源。

总体上看, 该方法使用很简单, 而且可以支持多个定时器并行处理。在定时器定时时间到时, Windows 系统会向应用程序发送 WM_TIMER 消息, 并由用户在 OnTimer() 函数中处理。但其缺点也很明显: 当程序定时过程中遭遇到系统忙时, 很可能会导致定时时间已到但定时器仍未响应的结果。此外, 该方案受限于系统时钟, 定时精度很低, 最好情况下也只能达到数十毫秒^[5,6]。因此, 这种定时思路虽然具有使用简单的优点, 但一般只能用于某些精度要求不高的系统中。

1.2 sleep() 函数

该方案采用 sleep() 函数达到延时的目的, 它的定时最小单位也同样是毫秒。在 MSDN 文档中, 该函数的最小可用参数为 1ms, 但经实际测试, 该函数的最小延时精度在 30ms 以上, 即 sleep(1) 和 sleep(30) 实现的延时效果是相同的。因此, 该方法精度也非常低, 而且采用 sleep 函数的一个重要缺点是在延时期间不能处理其他消息, 如果延时时间过长, 其表现形式如同系统死机, 此时 CPU 的占用率也非常高。

总体上看, sleep() 函数也只能用于精度要求不高

的延时程序中。

1.3 GetTickCount() 函数

GetTickCount() 函数可返回从计算机操作系统启动后到被调用时刻所经过的毫秒数, 其返回值是 DWORD 类型。因此, 利用其返回值, 即可实现定时功能, 下面给出了一个示例代码, 实现了 200ms 的精确定时功能^[7]。

```
DWORD Start_t, Stop_t;  
Stop_t = GetTickCount();  
While (1)  
{ Start_t = Stop_t; // 上一次的中止值变成新的起始值  
do  
{  
    // 用户自定义代码  
    Stop_t = GetTickCount();  
}  
while(Stop_t - Start_t < 100); // 检测定时时间是否已到  
}
```

GetTickCount() 函数的定时精度要比 SetTimer() 和 GetTickCount() 函数相对较高, 可适用于要求更高一级的应用场合。

1.4 timeGetTime() 或 timeGetSystemTime() 函数

timeGetTime() 和 timeGetSystemTime() 均为多媒体库函数。首先介绍 timeGetTime() 函数, 它返回了从系统启动所经过的时间, 其函数原型为:

```
DWORD timeGetTime(void)
```

该函数的最大计数周期为 2^{32} ms (约 49.71 天), 如果达到了最大计数周期, 会从零重新开始累加。用户可通过查询方式来建立定时循环, 进而控制定时事件。该函数使用也相对简单, 同时定时精度亦可满足一般场合需求。但是, 在采用轮询的方式进行查询以实现定时功能时, 会在相当程度上增加程序资源开销, 如果本来资源就很紧张, 很明显该方法是不宜采用的^[8,9]。

timeGetSystemTime() 的功能和用法类似, 不再详述。

1.5 多媒体定时器

Windows 中提供了高精度定时器的底层 API 支持, 使应用程序可以得到周期性的时间中断, 无论运行什么应用程序, 操作系统都能在高精度多媒体定时器事件到来时打断该程序而先去调用定时器的回调函数^[10,11]。

在 Windows 的多媒体扩展库中, 提供了一系列定时器操作函数, 分别说明如下:

timeGetDevCaps(): 用于获取定时器的服务能力参数;

timeBeginPeriod(): 用于设置定时器的定时精度;

timeSetEvent(): 用于设置定时间隔并启动定时器;

timeKillEvent():用于删除定时事件;

timeEndPeriod():用于释放定时器资源。

下面给出使用该方法的步骤:

1)调用 timeGetDevCaps()函数,获取定时器服务能提供的最大和最小事件周期等参数;

2)调用函数 timeBeginPeriod(),设置定时器的最小计时精度,可由用户根据使用场合自行设定;

3)调用函数 timeSetEvent(),初始化并启动定时器事件;然后明确定时器的周期、精度、以及回调函数名称;

4)不需要使用定时器时,调用 timeKillEvent()函数,以删除定时器事件;

5)最后调用函数 timeEndPeriod(),删除此前通过函数 timeBeginPeriod()建立的最小定时器精度。

这种方法能保证定时中断得到实时响应,最高定时精度可达到 1ms,但使用时要注意的是该方法占用的资源较多。

2 高精度定时器设计

2.1 基于系统计数定时器

VC6.0 提供了系统函数 QueryPerformanceFrequency()和 QueryPerformanceCounter(),编程者可通过这两个函数调用直接访问系统计数器^[12]。

首先介绍这两个函数的原型:

```
BOOL QueryPerformanceFrequency ( LARGE_INTEGER * lpFrequency );
```

```
BOOL QueryPerformanceCounter ( LARGE_INTEGER * lpPerformanceCounter )
```

该方法的使用方法介绍如下:在定时之前,首先调用 QueryPerformanceFrequency()函数,以获取机器内部定时器的时钟频率 fCLK。然后在需要严格定时的事件前后均调用 QueryPerformanceCounter()函数,将两次调用函数的 * lpPerformanceCounter 变量计数差与时钟频率 fCLK 相除,则可得到两次调用函数期间所经历的精确时间。

2.2 基于 CPU 时间戳定时器

该方案是最为底层的解决方案,它是采用的 CPU 内建硬件寄存器。事实上,在目前使用所有的主流 CPU 芯片中均有一个时间戳计数器,该计数器以 64 位 Unsigned Int 的格式记录了自启动后经过的时钟周期数。它以 CPU 主频为基准计数,而考虑到现在的 CPU 主频已经得到了突飞猛进的发展,普遍都在数 GHz 以上,因此,该计时器的计量精度极高,理论上可以达到纳秒级的精度^[13]。

要访问该时间戳计数器,传统的 API 和 MFC 函数是不直接支持的。但可以通过汇编指令 RDTSC 实现。

该指令可将时间戳计数器的值保存在 EDX:EAX 寄存器对中。而在 Windows 系统下,EDX:EAX 寄存器对正是 C++ 程序保存函数返回值的专用寄存器。因此,只需要把这条指令看成一条普通的函数调用即可。

读取时间戳计数器的参考 C++/汇编混合代码:

```
__int64 GetTSC_cycle() //返回 CPU 当前时钟周期数
{
    __int64 Cnt;
    __asm RDTSC;
    __asm mov DWORD PTR Cnt, EAX
    __asm mov DWORD PTR (Cnt + 4), EDX
    return Counter;
}
```

为了使用 GetTSC_cycle 函数完成精确定时功能,首先应获取系统的中央处理器 CPU 的真实频率,一种方便且常用的方法具体介绍如下:

1)调用 QueryPerformanceFrequency(),首先得到系统内部自带定时器的基准时钟频率 f1;

2)调用 QueryPerformanceCounter(&S1),读取系统计数值;

3)调用 t1 = GetTSC_cycle()读取当前 CPU 的时间戳计数器值;

4)调用 Sleep(2000)函数延时约 2 秒,这是为了保证计算样本足够多,用户也可设置为其它合适的时间;

5)调用 QueryPerformanceCounter(&S2),读取高精度计数值;

6)再次调用 t2 = GetTSC_cycle(),读取当前 CPU 时间戳计数器值;

7)CPU 频率等于: $(t2 - t1) \times f1 / (S2 - S1)$ 。

利用上述方式得到 CPU 的频率后,利用时间戳计数器,即可设计出所需的定时器。由于该方案直接采用 CPU 的时间戳计数器进行定时,因此其精度是各种方案中最高的,可达纳秒级,适合高精度定时场合。

3 定时器精度综合比较

前文给出了 VC 平台下各种常用的定时器设计方法,并分析了其各自的工作原理。但为了详细比较各定时器的性能,还需要对不同方案的定时间隔进行测量。由于 CPU 时间戳寄存器精度最高,因此选用该方案来测量各定时器的性能。

以下详述测试方法:

1)启动待测量定时器,设定定时器服务程序;

2)在每次进入定时器服务程序时,均读取 CPU 时间戳计数器的值并记录在专用数组中,定时器运行 1500 个周期后停止;

3)统计数组中相邻元素之差,此即为采用该定时器实际的定时间隔。

本测试中,将定时间隔设置为 1ms,各定时器测

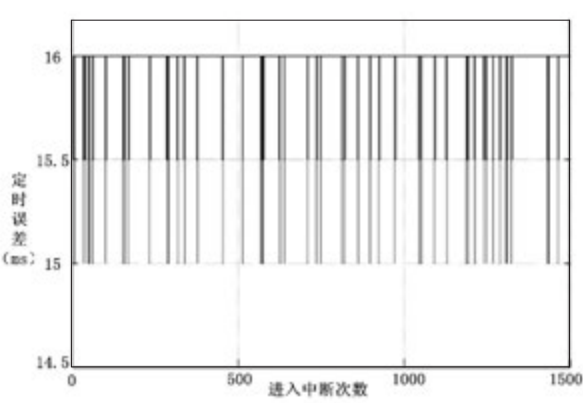


图1 常规定时器测量精度

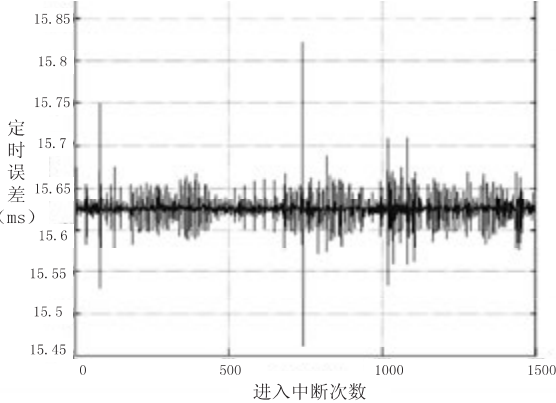


图2 sleep 定时器测量精度

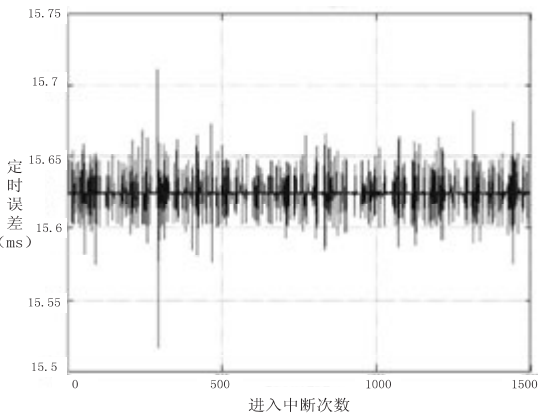


图3 GetTickCount() 定时器测量精度

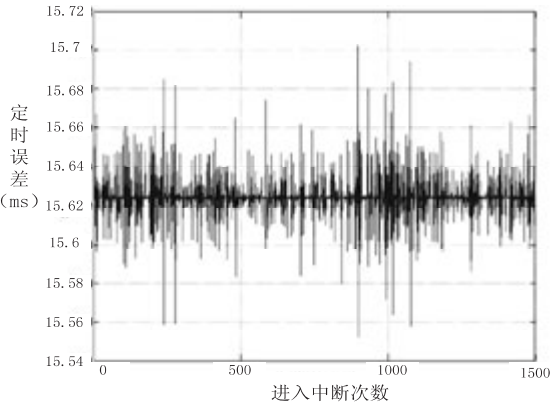


图4 timeGetTime() 定时器测量精度

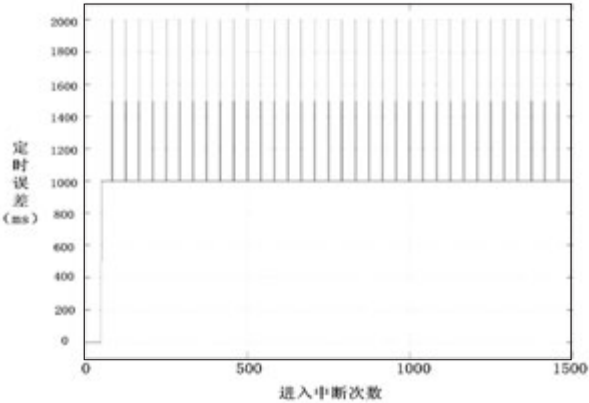


图5 多媒体定时器测量精度

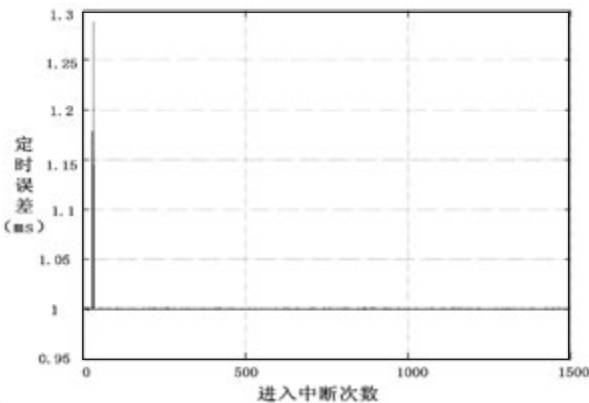


图6 基于系统计数定时器测量精度的实际间隔时间如图1~6所示,图中,横轴代表进入定时器中断次数,纵轴则代表实际定时时间。

从图中可以看出,常规定时器的最小定时时间间隔大约16ms,而sleep、GetTickCount和timeGetTime定时器的精度也都在15ms左右,这些定时器显然不能用于精确定时。多媒体定时器的性能同样不稳定,在开始的大约50个周期内其定时时间接近于0,此后定时器方能基本开始稳定工作,但即使如此,仍存在周期性的波动,波动周期大约40ms此时定时时间间隔可达2ms。同样地,在使用基于系统计数定时器和基于CPU时间戳定时器时也存在不稳定的现象,但持续时间明显短于多媒体定时器。此后,配合外部测试设备,又对这两种定时器进行了单独测试,证实这两种高精度定时器的定时误差分别为1μs和0.1μs。

在某雷达信号处理机测试设备中,要实现PC机与某VPX系统板卡的高精度数据采集,且定时误差不能超过1μs。因此,采用时间戳定时器方案,很好地完成了数据定时采集任务,该系统目前已经通过设备验收,这也证实了时间戳定时器方案的有效性。

4 结束语

VC6.0作为目前最为流行的Windows环境开发平台,在各种测控系统、实时数据采集系统中均得到了广

泛应用,而在这些系统中,通常需要与嵌入式系统板卡进行各种高精度定时数据采集、处理等操作。因此,需要在 VC 中方便地实现高精度定时器。

因此,文中对 VC 平台下各种定时器的设计进行了总结,并对其精度进行了详细测试。结果证实:通常的 VC 定时器方案定时误差均在数十毫秒以上,根本无法满足高精度定时要求。而如果对定时精度要求较高,则需要采用 CPU 自带的硬件寄存器完成定时功能,如文中 2.1、2.2 所示,此时其精度可达微秒级甚至更好。在某雷达信号处理机测试设备中,也正是采用 2.2 所示的时间戳定时器方案才达到了设计要求,并最终通过系统验收。

总体上,Windows 系统常规定时器使用简单方便,但定时时间越短误差越大,适合于定时时间长,精度要求低的场合。sleep、GetTickCount 和 timeGetTime 定时器精度与常规定时器性能相当。多媒体定时器可获得最高 1 ms 的定时精度,但需占用较多资源。基于系统计数定时器和 CPU 时间戳的定时器具有微秒级的定时精度,且具有很高的稳定性,相应地,其缺点是资源占用率高,比较适用于对定时精度要求高的程序场合,如实时仿真和数据采集等。而且,事实上以上各种方法并不仅局限于 VC 平台本身,因而在不同的软件开发平台下,各种定时方法均有其各自的优越性,具体应用时应根据实际的需要合理地进行选择,在定时器精度和系统开销之间取得较好的平衡。

参考文献:

[1] 卓红艳,赵平. 基于 VC++ 的实时数据采集系统中定时

器的使用与比较[J]. 现代电子技术,2007,18(2):129-132.

[2] Zhou Xuejun. Applied Mechanics and Materials[J]. Applied Mechanics and Materials,2011,93(9):1795-1800.

[3] Zhu Yongguang, Sun Zhengshun, Zhao Nanyuan. Video Capture Program Based on Visual C++ and the Application of Timer[J]. Computer Engineering and Applications,2002,20(12):128-132.

[4] Tang Hongzhong, Huang Huixian, Yin Lin. Application of VC++ + DLL Timer in Design of Industrial Control Software[J]. Ordnance Industry Automation,2003,18(6):781-786.

[5] 郭占社,孟永钢,苏才钧. 基于 Windows 的精确定时技术及其在工程中的应用[J]. 哈尔滨工业大学学报,2005,18(12):38-41.

[6] 姚 晔,胡益雄. VC++ 应用程序精确定时方法的实现[J]. 计算机系统应用,2001,22(9):192-195.

[7] 何 斌,韦 工. 基于多媒体时钟的定时控制[J]. 舰船电子工程,2006,26(4):4-8.

[8] 洪锡军,李从心. Windows 下高精度定时的实现[J]. 计算机应用研究,2008,14(3):147-150.

[9] 何 斌,耿春萍. 多媒体时钟解决实时控制系统的定时[J]. 飞行器测控学报,2006,25(6):189-191.

[10] 李福华,李悦丽,段巧雄. Windows 环境下多串口控制软件设计中精确定时的实现[J]. 电子技术,2010,14(6):213-216.

[11] 刘春风,田延岭. Windows 操作系统下的软件定时器的设计与应用[J]. 机电一体化,2004,10(5):39-42.

[12] 毕 业,史忠科. Windows2000 下高精度定时器设计与实现[J]. 工业仪表与自动化装置,2007(1):53-56.

[13] 韩志勇,李先国. Windows 操作系统下高精度计时研究[J]. 计算机工程与设计,2005,24(9):236-239.

(上接第 43 页)

[3] Page L, Brin S, Motwani R, et al. The PageRank Citation Ranking:Bringing Order to the Web[R]. Stanford: Stanford InfoLab. ,1999.

[4] 曹 林,韩立新,吴胜利. 元搜索引擎排序技术综述[J]. 计算机应用研究,2009,26(2):411-414.

[5] Regenwetter M, Tsetlin I. Approval voting and positional voting methods: inference, relationship, examples[J]. Social Choice and Welfare,2004,22(3):539-566.

[6] 盛宪锋,山 岚. 基于元搜索引擎的专业式智能网络信息检索系统[J]. 计算机工程与设计,2004(1):69-73.

[7] Weiss O S. Conceptual Clustering Using Lingo Algorithm: Evaluation on Open Directory Project Data[C]//Proc of IIP-WM. [s. l.]:[s. n.],2004:369-377.

[8] 严莉莉,王倩倩,孟 杰,等. 基于聚类的个性化元搜索引擎设计[J]. 计算机技术与发展,2007,17(4):186-188.

[9] Bartell B T, Cottrell G W, Belew R K. Automatic Combination of Multiple Ranked Retrieval Systems[C]//Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland:[s. n.],1994:173-181.

[10] 胡升泽. 个性化元搜索引擎若干关键技术研究[D]. 长沙:国防科学技术大学,2008.

[11] 孟 星. 基于 Agent 的自适应信息检索系统技术研究[D]. 西安:西安电子科技大学,2009.

[12] 王 忠,程 磊. 基于元搜索引擎的个性化 Web 信息采集[J]. 计算机工程与设计,2009(13):3117-3119.

基于Visual C6.0的Windows应用程序定时器研究

作者: [王鹏飞, 王鹏](#)
作者单位: [中国空空导弹研究院, 河南 洛阳 471009](#)
刊名: [计算机技术与发展](#)
英文刊名: [Computer Technology and Development](#)
年, 卷(期): 2013(2)

本文链接: http://d.g.wanfangdata.com.cn/Periodical_wjtz201302013.aspx