# 基于 Win32 的控制软件定时器程序的编写

## Writing Timer Procedure Based On Win32

梅洪　单家元
北京理工大学机电工程学院 100081

**摘要：** 由于控制系统的功能越来越多地通过软件实现，在控制系统的软件编写过程中就经常涉及到各种定时器程序的编写。本文主要介绍几种在 NT 下通过软件获得较高精度定时器的方法，并对其性能作了粗略的分析。
**关键词：** 定时器, Win32, 实时；
中图分类号: T P 391 9: T P 273
**Abstract：** More control system's function lies on software implementation, and high performance timer procedure is always used in those software. The paper introduces how to write and use timer procedure in software based on WinNT.
**Keywords:** Timer, NT, Realtime

## 1. 前言

定时器一般是指在一定时间范围内多次或单次触发的逻辑或物理装置。在控制系统中，定时器是重要的逻辑/物理设备，通常用来周期性地（通常也要求是实时的）执行某一既定动作，如状态报告、扫描设备运行情况或通讯等。由于计算机控制系统的大量应用，早期控制系统中的各种硬件设备的逻辑功能逐渐简化或直接由计算机软件直接实现，使系统的易用性和通用性都大大增强。但同时也带来不利因素，硬件设备的减少使很多原来由硬件时钟完成的定时器或中断功能改由计算机时钟＋控制系统软件实现，使得控制系统软件设计难度增加，稳定性降低。尤其因为进来很多控制软件要求有更好的用户界面转而在 Windows NT 中实现，使这一矛盾更加突出。本文从控制系统软件编写人员的角度，介绍几种较为简单实用的定时器的使用方法。

## 2. 定时器程序的编写

随着计算机技术和控制科学的发展，通用 PC 机大量应用到计算机控制系统中去，控制系统软件也由简单的面板控制发展为操作系统＋应用程序的形式。为满足一般非计算机专业人员的使用要求，软件设计上要求使用方便，界面直观，所以近年来 Windows 的使用越来越多。

作为控制系统软件的支撑平台，一般要求 OS 的稳定性较高且最好可以提供较好的运行效率。Windows NT 尽管使用方便，但在效率上却不尽人意，且对硬

件的操作也不太方便。事实上微软提供有专门用于嵌入式或工控的特殊版本的 Windows NT（包括 Windows NT Embeded、Windows XP Embeded 和 Windows CE），这些版本对硬件的需求较少，且可以定制系统内核（Windows 并不是微内核的 OS），对提供系统运行效率有极大的好处。然而售价太高，现在也只有少数 PDA 类的手持 PC 机上使用 CE，而 NT/XP Embeded 更是少见。好在现在的 PC 机硬件发展迅速，普通的 NT 也可以满足大部分的要求。

## 2.1 Windows NT 定时器编程的一般方法

Windows NT 来源于 VMS，是运行于保护模式的完全的 32 位操作系统。通常 Windows NT 并不允许运行在用户态的应用程序直接存取硬件设备，这样 DOS 时代利用硬件时钟定时中断的办法就行不通了。Windows NT 一般通过 Platform SDK 的形式提供给用户应用程序的开发接口（API），其中提供的几组和时间相关的 API 可以解决这个问题。通过使用这几组 API 和适当调整应用程序的结构，不难满足定时器的要求。另外，工控程序通常是多线程的：一方面是用户界面的要求，程序运行时不至于失去对用户的响应；另一方面也有助于提高 CPU 的利用率，增强程序的性能。

## 2.2 普通定时器

一般常见于 Windows 编程书籍完成定时器功能的是 SetTimer 这个函数。这个函数最早在 16 位的 Windows3.x 上已经出现了，一般用于提供精度要求不是很高的定时。函数原型如下：

```
UINT_PTR SetTimer( HWND hWnd,             // handle to window
                   UINT_PTR nIDEvent,     // timer identifier
                   UINT uElapse,          // time-out value
                   TIMERPROC lpTimerFunc  // timer procedure
                   );
```

此函数的工作原理大致是根据 nIDEvent 向相应的窗口消息队列中加入一条 ID 为 nIDEvevent 的 WM_TIMER 消息，等到窗口消息处理函数处理 WM_TIMER 消息时，根据相应的 TimerID 执行不同的代码；或者如果最后一个参数 lpTimerFunc 不为 NULL，则以类似 hook 的方式将指定窗口的 WM_TIMER 消息处理函数换成用户自定义的 TimerProc。最后，需要用 KillTimer 清除定时器释放系统资源。有一点需要注意的是 SetTimer 必需用在有窗口消息处理函数的程序中，这意味着控制台（Console）程序无法直接使用 SetTimer。具体的例子可以参考相应的书籍。

之所以称之为低精度计时器，就是因为 WM_TIMER 消息是以同步方式在消息队列中等待处理，优先级较低；另外，SetTimer 的时钟源实际上是 PC 机日时钟（由 8253/8254 的频率 1.1932MHz 除以 65536 分频得到）提供的，只能达到 18.2Hz，因此函数的第三个参数在小于 18.2 时实际上是没有意义的。综合这两个因素，SetTimer 提供的定时精度和频率都非常有限。

另外，通过改写 8253/8254 分频数的方法在 NT 下也不是很容易实现。一方面牵扯到端口直接读写的问题，需要编写驱动程序，实现较复杂；另一方面，IRQ0

对 Windows 非常重要，很多 Windows 的操作都依赖日时钟（如双击鼠标键的间隔控制等），稍有不慎则导致系统崩溃。

## 2.3 高精度计数器

如果把定时器看成按照“计时－触发”方式工作，那么一个精度足够的计数器也可以完成定时器的功能。在 MSDN 中，QueryPerformenceCounter 被表述成用来计算程序计算性能的函数，其原型如下：

```
BOOL QueryPerformanceCounter(LARGE_INTEGER *lpPerformanceCount);
```

它一般需要和 QueryPerformanceFrequency 配对使用。前者得到一个只与时间有关的类似流水号的且不断增加的值，后者则得到每秒记下的点数。根据系统总线和 CPU 频率的不同，QueryPerformanceFrequency 得到的值是不同的(比如一台 CPU 是 1.5GHz 的机器上这个值是 3579545)，但对同一台计算机总相同。这样，我们利用前后两次 QueryPerformanceCounter 的差除以此频率值，就可以得到两次计时之间的时间差。下面是一个例子：

```
...
LARGE_INTEGER t1,t2,f;
QueryPerformanceFrequency(&f);
QueryPerformanceCounter(&t1);
While(((double)(t2.LowPart – t1.LowPart))/((double)(f.LowPart)) < SomeValue )
    QueryPerformanceCounter(&t2);
//timeout
//do other things
...
```

把这些代码放到定时线程中去，很容易就可以得到一个高精度的定时器，并且可以达到毫秒级的精度。当然，这是在假定两个函数的精度都足够高且计算机速度很快的情况下。但是他们计数的精度还是总可以保证的。

## 2.4 多媒体定时器

为了给多媒体设备（如 MIDI 序列发生器等）和程序提供高精度的定时器，微软在 32 位版本的系统里提供了一组所谓的“多媒体定时器”API，可以根据具体计算机的性能最大限度的提供高精度定时。

这一组 API 中常用的是 timeSetEvent 和 timeKillEvent，原型如下：

```
MMRESULT timeSetEvent(UINT uDelay, UINT uResolution, LPTIMECALLBACK lpTimeProc,
                      DWORD_PTR dwUser, UINT fuEvent);
MMRESULT timeKillEvent(UINT uTimerID);
```

这一对函数的用法与 SetTimer/KillTimer 类似，首先用 timeSetEvent 启动定时器，TimeProc 用来指定用户的处理函数。与 SetTimer 不同的是，timeSetEvent 在控制台程序和窗口程序中都可以运行。用 Spy++观察后发现，timeSetEvent 执行后（若成功）会启动额外的线程，猜测这是 timeSetEvent 可以同时运行在控制台和窗口程序中的原因。

同样，函数的第一个参数 uDelay 指定的时间间隔也是以毫秒为单位的。这意

味着理论上可以达到 1 毫秒的精度。

程序退出时同样需要用 timeKillEvent 释放定时器资源。

## 2.5 三种方法的测试和比较

一般情况下，控制系统软件的定时要求都比较高，往往达到毫秒级的精度。这样，采用 SetTimer 显然是不可行的，和前面提到的相似，SetTimer 不仅不能获得高频率的定时器，即使是较低频率也无法保证定时器触发的间隔准确，测试的结果很乱。而使用 QueryPerformanceXXX 族函数族或多媒体函数则可以达到较好的精度。

测试方法如下：

定时器精度为 1ms，分别连续 10 次计时 1s、2s、3s（通过 QueryPerformanceCounter 函数），再计算出误差，假设 QueryPerformanceCounter 计算得到的是真实时间。

结果如下表：

| 总时间 | 1s | 2s | 3s |
|---|---|---|---|
| 高精度计数器定时 | -0.01482311% | -0.01501029% | -0.01509503% |
| 多媒体定时器 | -0.01509448% | -0.00020952% | 0.00009573% |

注意假定的前提条件是 QueryPerformanceCounter 误差忽略不记。

从实验结果来看采用多媒体定时器似乎更好一些，但是受实验条件的限制，这里得到的也只是粗略的结果，但可以认为两种方法满足精度的要求，最终的优劣判断还需要结合具体的应用程序才有意义。注意，采用多媒体定时器时，1s 测试的误差较大，原因是多媒体定时器需要启动额外的线程，导致一定的时间开销。一般定时器连续工作的时间都会远超过测试的时间。

（受篇幅限制，测试用程序未加入，需要可另外索取）

# 3. 其他方法

以上介绍的是 Windows NT 中几种定时器的编程方法，主要通过操作系统的 API 完成相应的功能，也就是尽量通过软件完成功能。当然，也可以通过其它的一些小技巧达到目的，比如通过计算并利用_asm{nop}指令的运行时间来获得高精度的定时器。但这些方法都有一定的局限性，即当硬件性能较差或系统同时运行的程序较多时，定时的精度就较差。对于一部分运行环境较为苛刻的控制系统程序而言，采用专门的硬件中断电路是一个较好的选择。但这样作系统的成本自然也较高。

如果开发能力较强，也可以选择自行编写硬件驱动程序的办法来获得高质量的中断源当做定时器。

在不增加硬件的情况下，可以直接接管系统的 IRQ0。这里包括两部份内容：一是改写 8253/8254 的分频系数并编写自己的中断服务程序，二是在调用自己的中断服务程序后，根据新的分频系数计算日期和时间，并调用原 0 号中断的入口程序。注意在中断处理程序中不要进行密集计算或大量的 I/O 操作。还有一个方法就是对 PC 的 CMOS 实时钟进行编程，也可以获得最高至 8192Hz 的时钟频率，

但可选的频率是固定的几种，不是很灵活。具体的驱动程序可以使用 Numega 的 DriverStuio 等开发，请参考相应的资料。

## 4. 结论

根据上面的介绍，一般控制系统软件的精度较高的定时器是比较容易获得的。借助强大的硬件能力和良好的程序逻辑，并不需要太多的计算机专业知识也可以获得一个快速响应的稳定的控制系统程序，使实时控制称为可能。

参考文献

[1] Microsoft Corporation, MSDN. Microsoft Corporation, 2001
[2] Deltatau Data Systems Inc, PMAC Manual. 1997
[3] Charles Petzold, Programming Windows.Microsoft Press, 1998

作者简介：

**梅洪**(1978-)，男，北京理工大学 2000 级硕士生，研究方向计算机仿真(68912418, bsdguru@163.com)

**单家元**（1967－），男，北京理工大学机电工程学院教授，博士，研究方向为导弹控制制导、半实物仿真、分布交互仿真，现任计算机仿真分会理事

北京理工大学，６８９１２４１８或６８９１１９２１

手机.13041123200,

email: bsdguru@163.com