

Applications of Timer of Data Sampling Software

Hongyan ZHUO*, Chengliang GE, Zhiqiang LIU, Jiaru ZHANG

Institute of Applied Electronics

CAEP, P.O. Box 919-1011

Mianyang 621999, China

*E-mail: 526756092@qq.com

+* Corresponding author

Abstract—Based on periodic inspection and scanning, data sampling software usually used *WM_TIMER* and *MM_TIMER*. *MM_TIMER* is a multimedia timer. There are differences between *WM_TIMER* and *MM_TIMER*, such as implementation methods, *PRI* and precisions of time. Herein, the great differences are compared with experiments. Its precision will be reached at 1ms with applications of *MM_TIMER*.

Keywords—periodic inspection and scanning; precision of timer; *MM_TIMER*; *WM_TIMER*

I. INTRODUCTION

Up to today, there are two methods of data sampling system based on periodic inspection and scanning. One method is using embedded system to achieved microsecond precision by hardware clock. Another method is using A/D card to achieved millisecond precision. In the second method, most of users developed applied software based on Windows system. And for such multitask and multiuser system of Windows, developers cannot operate the rock-bottom hardware. Then, developers achieved periodic data sampling by STK of applied software [1]. *MM_TIMER* and *WM_TIMER* are usually used timers of VC by developers. And there are great differences of precision between two timers. So, the methods and precisions of the two timers are presented in this paper.

II. DESIGN AND IMPLEMENTATION OF *WM_TIMER*

WM_TIMER is a timer which is provided by Windows. The method is easy and simple. Developer can transfer *SetTimer()* function to allocate a timer to application program. While a time interval is specified, Windows system sends a *WM_TIMER* message every specified time interval. Then the program can implement events in real-time.

The procedure of implementation is listed as initialing of *SetTimer()*, closing timer with *KillTimer()*, sending message queue, sampling data in response function of *WM_TIMER*.

Function of *SetTimer* is defined as below.

```
UINT_PTR SetTimer(
    UINI_PTR nIDEvent,
    //identification code of timer
    UINT nElapse,
    //time interval with unit of ms
    Void(CALLBACK*
    lpfnTimer)HWND,UINT,UINT_PTR,DWORD)
);
```

Wherein, parameter *lpfnTimer* is the callback function of timer message. If it is NULL, *WM_TIMER* sends

message queue of program and will be disposed by *OnTimer* member fuctions. The return value 0 means fault.

For a data sampling system, codes of data sampling with *CFormView* basic class is listed below.

```
//setup timer under pushbutton of 'Data Sampling'
void CTestView::OnButtonRun()
{
    // TODO: Add your control notification handler code here
    SetTimer(1,20,NULL);           //sampling periods are 20ms
}
//implementation of data sampling
void CTestView::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    ...
    writing functions of data sampling
    ...
}
//close timer under pushbutton of 'Stop Sampling'
void CTestView::OnButtonRun()
{
    // TODO: Add your control notification handler code here
    KillTimer(1);
}
```

III. DESIGN AND IMPLEMENTATION OF *MM_TIMER*

MM_TIMER is a multimedia timer provided by Windows system. Within VC software, its difficult than *WM_TIMER*. The main procedure is to setup parameters of *MM_TIMER*, to transfer *timeGetDevCaps()* function to decide supported minimal precision and maximal precision, to startup timer to do applications, to release sources by deleting timer. The more used functions and means are listed below [2,3].

```
timeBeginPeriod( ) //setup minimal resolution of timer
timeEndPeriod( ) //eliminating minimal resolution of timer built by ascending function
timeSetEvent( ) //producing time of allocated period
timeKillEvent( ) //deleting events of front timer
```

The interface functions of *MM_TIMER* are included in DLL of *mmsystem.dll*. And VC++ provides response head files. The detail implementation methods are listed below.

(1)Setup resolution of Timer

MMRESULT *timeGetDevCaps(lpTIMECAPS&ptc, UINTcbtc)* transfers *timeGetDevCaps()* function to judge the minimal resolution and maximal resolution. Then it transfers *timeBeginPeriod(UINTuPeriod)* function to setup resolution of timer. The parameter *UINTuPeriod* is the value of timer resolution.

(2) Setup events of timer

MMRESULT *timeSetEvent(UINT uDelay, //uDelay: interval of sampling*

UINT uResolution, // uResolution: precision of time, its default value is 1ms

LPTIMECALLBACK lpTimeProc, //lpTimeProc: user defined callback //function

DWORD dwUser, UINT fuEvent); //dwUser: user provided callback function

// UINT fuEvent: mode of event, triggered by TIMER period

(3) Declaring of whole callback function

Void CALLBACK TimerCallBackProc(UINT wTimerID, UINT mMsg, DWORD dwUser, DWORD dw1, DWORD dw2)

Wherein, *wTimerID* is sign of timer. *mMsg* is reserved. *dwUser* is parameter used by user. *dw1* and *dw2* are reserved.

(4) user defined message disposal function

The message disposal function defined by user is to receive messages from *MM_TIMER*.

IV. APPLICATIONS

A. Experiments

Above methods are applied in one big system. The curves of two timers are showed in figure 1 and figure 2. Figure 1 is curves of sampled data with applications of *WM_TIMER*. The precision of *WM_TIMER* is 15ms. But, there is 'Losing Second' phenomenon. The maximal losing second is about 2s-4s. As showed in figure 1, the real sampling time is 18s and figure 1 has only 16s. The 2s is lost.

Figure 2 is curves of sampled data with applications of *MM_TIMER*. The precision of *MM_TIMER* is 1ms. And, there is no 'Losing Second' phenomenon. As showed in figure 2, the real sampling time is 18s and figure 2 has data with time duration of 18s. Time is not lost.

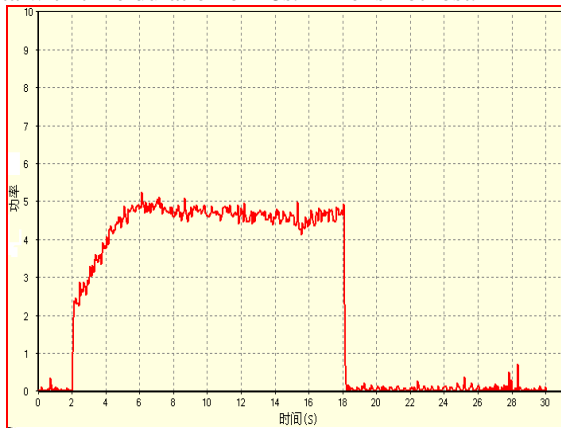


Figure 1. Data sampling curves of *WM_TIMER*

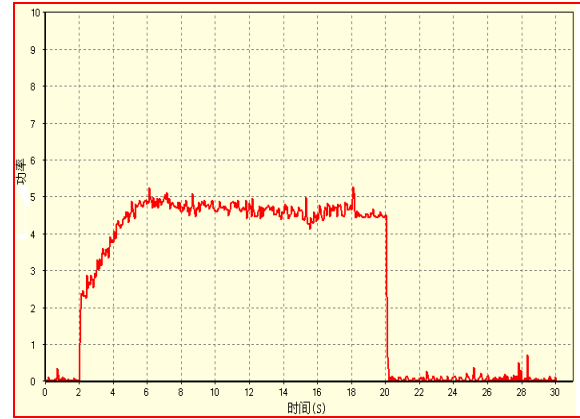


Figure 2. Data sampling curves of *MM_TIMER*

B. Analysis

After doing many experiments, it can be found that *WM_TIMER* is a simple extending of IBM PC hardware and timer logic built by ROM BIOS. ROM initializes the timer chip of Inter8254 to produce interrupt 08H. Frequency of interrupt 08H is about 18.2Hz. BIOS updates two timer variables of *TIMER-LO* and *TIMER-HI* with interrupt 08H. Hence, the maximal fault of *WM_TIMER* is that its resolution is 55ms [4]. That is, the program can receive 18 messages per second at best. And the *PRI* of *WM_TIMER* in Windows system is very low. While there are several *WM_TIMER* events, Windows system combines the events. Then there will be some events lost by Windows.

For mechanism of implementation of *MM_TIMER*, *MM_TIMER* provides hardware interrupt. *MM_TIMER* has its independent thread to transfer its callback function. It has high *PRI*. Its precision can be achieved to 1ms [5].

In the applications, it is watchful that it must be transfer *MM_TIMER* before end of the thread. Otherwise, the system will be done. And after application of *MM_TIMER*, the timer and its functions must be deleted. Otherwise, the system will be changed slowly and slowly.

V. 5 CONCLUSIONS

By studying of applications, some conclusions are listed in table 1. From table 1, applications without requirements of high precision can apply *WM_TIMER*. And for an application with high sampling frequency and precision, the *MM_TIMER* is needed.

TABLE I. COMPARISON OF TWO TIMERS

	<i>WM_TIMER</i>	<i>MM_TIMER</i>
Coding	Concision and Conveniency	More codes
Precision	55ms in general	Maximal value is 16ms Minimal value is 1ms
<i>PRI</i>	Low	High
Sampling frequency	It is easy to appear 'Losing Second' while the sampling frequency is high.	It is not easy to appear 'Losing Second'.

REFERENCES

- [1] LI Jing, YANG Jun-wu, QIAN Xu. Precise control of sample frequency with multimedia timer, *Computer Applications*, 2000, 12 (12):67-68.(in Chinese)
- [2] ZHENG Cun-hong, HU Rong-qiang, ZHAO Rui-feng. Data Acquisition System Programming with Visual C++, *Computer Applications and Study*, 2002, 4:103-104,108. (in Chinese)
- [3] YU Yong, LEI Zhi-yong. Realization of real-time measurement and control system under Windows, *Modern Electronic Technology*, 2005, 5(196):22-23. (in Chinese)
- [4] WANG Wei, XU Guo-hua. Applications of multimedia timer in industrial control, *Micro-machine and Applications*, 2001, 12:8-9, 10. (in Chinese)
- [5] GUAN Hong-wei, GAO Wei, CHEN Bao-xue. Solution of the real time data acquisition based on Windows environment, *Applied Science and Technology*, 2004, 31(4):35-37. (in Chinese)