**Krzysztof KOŁEK**

AGH University of Science and Technology

# Windows7 x64 as real-time measurement and control platform

***Streszczenie.*** *Systemy operacyjne z rodziny MS Windows, a w szczególności system Windows 7 w wersji 64-bitowej, dominują na platformie komputerów PC. Artykuł przedstawia cechy Windows 7 umożliwiające realizację systemów pomiarowo-sterujących o stałym okresie próbkowania. Badano punktualność realizacji zadań czasowych oraz zaproponowano algorytm jej poprawy. Przedstawiono wyniki sterowania laboratoryjnym modelem trójwymiarowej suwnicy.* ***Windows 7 x64 jako pomiarowo-sterująca platforma czasu rzeczywistego.***

***Abstract****. MS Windows operating systems, in particular 64-bit Windows 7, are the most popular for PC computers. The paper presents some features of the Windows 7 that can be applied in real-time measurement and control systems running at constant sampling. The jitter of the Windows timers is shown. An algorithm that improves the accuracy of the timer tasks is proposed. The paper describes control results for 3D crane model running in the Windows 7 environment.*

**Słowa kluczowe**: system operacyjny, pomiarowo-sterujący system czasu rzeczywistego, niepunktualność.
**Keywords**: operating system, real-time measurement and control, jitter.

## Introduction

MS Windows family of operating systems covers almost 90% of systems installed at notebook and desktop computers. The trends given in Fig.1 confirm dominant and increasing position of Windows 7, nevertheless still the significant position of Windows XP is noted [1].

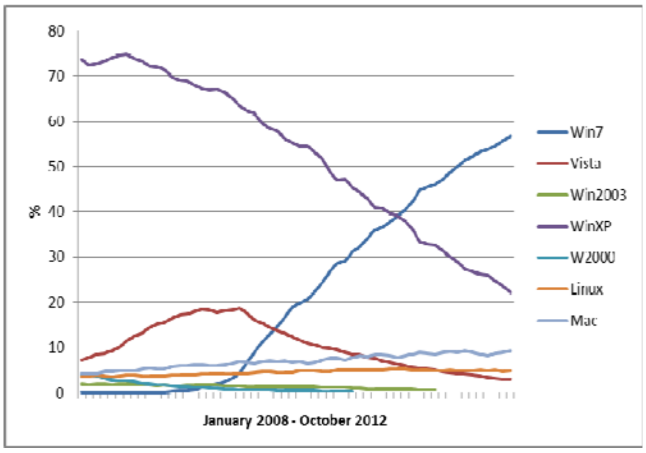

Fig.1. Trends in the use of operating systems

Increasing number of 64-bit systems is second significant trend. As mentioned at *The Windows Blog*, almost 50% of systems run 64-bit editions of Windows 7.

Two main systems, Windows XP and Windows 7, are not designed to be hard real-time operating systems (RTOS) but there is an observable pressure to apply these OSes for real-time control tasks. The systems are widely applied and familiar to the user. Also there exists a lot of software packages applied to design, monitor and analyse measurement and control systems. Running this software at one machine and real-time tasks at another machine seems to be a tempting alternative for expensive commercial RTOSes. Some commercial real-time extensions of the MS Windows kernels are also available: RTX [2] or RTKernel [3]. The extensions are deeply embedded into OS kernel and, in fact, a separate real-time environment is established. Unfortunately, the extensions are usually not compatible with the latest MS Windows kernel versions, in particular the 64-bit systems are not supported.

Let us consider some features of the 64-bit Windows 7, probably the most popular OS within next few years, as a platform for direct execution of real-time measurement and control tasks.

## Requirements of the RT control systems

Basic requirement of the real-time control system is the reaction guarantee to events within a predefined time period, regardless on the time and sequence of events arriving to the computer.

Real-time systems are classified as hard or soft real-time. In the hard system the reaction must fulfil the time constraints. Missing a deadline may lead to serious risk to the controlled plant. In the soft real-time systems some delays are acceptable but the quality of the system degrades in such cases.

Most real-time control systems run in one of two ways. First, responding to external events. Sometimes this mode is supported by an interrupt system. Second operating mode makes use of hardware timers to perform some equally timing distributed tasks.

The first mode is preferred for supporting emergency signals. The second mode is applied in constant sampling period control systems. Also measurement systems, where constant sampling frequency is required, apply constant timing events. Only the parameters of constant sampling period systems are analysed in this paper.

Unfortunately, constant sampling is just an idea and in the real sampling one can observe some deviations to ideal sampling time points.

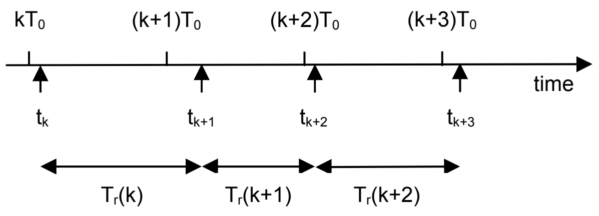

Fig.2. Exact and real sampling times

In figure 2 the exact sampling time points for sampling period $T_0$ are marked as $kT_0, \ldots (k+3)T_0$. They differ from the real sampling point $t_k, \ldots t_{k+3}$ leading to varying sampling periods $T_R(k)$, $T_R(k+1)$ and $T_R(k+2)$. The deviation of duration of real sampling periods is called jitter.

There are different sources of the deviation of real sampling point:

- time jitter of the quartz oscillator. Stability of the quartz elements usually equals to a few ppm so this factor can be neglected in our considerations,
- jitter of the network communication in distributed systems,

- jitter in the communication with external I/O devices, e.g. connected by the USB link [4],
- differences in calculation times in each sampling, leading to different jitters for sampling and acting,
- also some jitter is introduced by the operating system. Each OS contains critical sections, which have to be executed in an uninterruptable way. At the beginning of the critical section the OS disables interrupts, disabling also the source of the time pattern for the sampling periods.

Time jitter continuously changes the sampling period and may lead to loss of stability. A controller can compensate the jitter by reading the duration of the sampling from precise timer and at each sampling period introduce a new sampling value to the calculations. Unfortunately, also switching stable controllers does not guarantee that the system remains stable. Jitter compensation and the stability analysis are described in many papers [5].

**Jitter in a commercial RTOS**

The real-time operating system is optimised to guarantee low jitter values. As an example the Xilkernel is considered [6]. The Xilkernel is a small and scallable microkernel dedicated to MicroBlaze and PowerPC 405 processors embedded in the Xilinx reconfigurable circuits. The features of the Xilkernel are typical for RTOSes: POSIX compatible API functions, real-time scheduling algorithms, real-time synchronization, interprocess communication services and precise timers.

To measure the jitter parameters the Virtex-4 FPGA equipped with PowerPC processor was used. The board was driven by the 100MHz clock. The processor served a single timer with the 10ms period. A precise timer, implemented as part of the FPGA, was used for time measurements. The histogram of the duration of the sampling periods of experimental data is shown in figure 3.

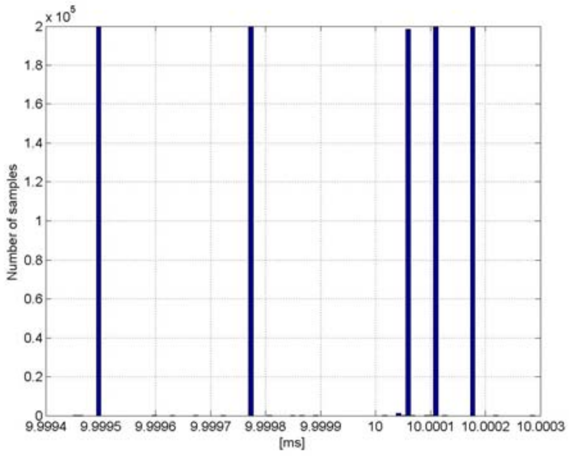

Fig.3. Jitter histogram of the Xilkernel real-time kernel

Figure 3 shows the histogram of one milion samples. Each subsequent sample corresponds to the length of time sampling. One can observe that the sampling times are very close to the reference period equal to 10 ms. The deviations among all sampling times are less than 4 microseconds. Some quantitative values of the collected data are shown in table 1.

**Time source in Windows 7**

The x86 processors, since Pentium architecture has been introduced, contain a 64-bit Time Stamp Counter (TSC). It counts processor clock pulses. Although the TSC may not be directly accessible by user applications Windows 7 API contains two functions to read the frequency and value of this counter:

- *QueryPerformanceFrequency* – returns frequency in counts per second. At most of PCs it returns the value of 1688496. It allows time measurements with the 592 nanosecond resolution,
- *QueryPerformanceCounter* – returns current counter value.

In the following experiments the TSC API functions are used to measure all timings.

**Process and thread priorities in Windows 7**

Priority management in Windows 7 is divided in assigning priorities to processes and to tasks. There are the following process priorities, presented in the ascending priority order:

*IDLE_PRIORITY_CLASS,*
*BELOW_NORMAL_PRIORITY_CLASS,*
*NORMAL_PRIORITY_CLASS,*
*ABOVE_NORMAL_PRIORITY_CLASS,*
*HIGH_PRIORITY_CLASS,*
*REALTIME_PRIORITY_CLASS.*

The highest possible priority preempts all threads, including important system tasks. If user task operates at the *REALTIME_PRIORITY_CLASS* priority level it may cause unstable system behaviour.

The following priority classes can be asssigned to the threads:

*THREAD_PRIORITY_IDLE,*
*THREAD_PRIORITY_LOWEST,*
*THREAD_PRIORITY_BELOW_NORMAL,*
*THREAD_PRIORITY_NORMAL,*
*THREAD_PRIORITY_ABOVE_NORMAL,*
*THREAD_PRIORITY_HIGHEST,*
*THREAD_PRIORITY_TIME_CRITICAL.*

The higherst priority is at the bottom of the list. The assignment of the priorities to the threads is performend by the *SetThreadPriority* function.

It is not recommended to use the REALTIME_PRIORITY_CLASS in experiments therefore, the HIGH_PRIORITY_CLASS is applied. In the experiments the priorities of threads are set to *THREAD_PRIORITY_TIME_CRITICAL.* Such selection guarantees fast, but still safe, execution of the test application.

**Timer sources in Windows 7**

Windows 7 contains two types of software timers that can be applied to trigger periodical time events: the WM_TIMER message and the multimedia timer. The resolution and minimum period of both timers is 1 milisecond.

To start the first timer type a thread calls the *StartTimer* function. The operating system, when the timer period expires, sends the WM_TIMER message to the thread’s message queue. Than the massage is processed by a callback function specified during the *StartTimer* call. The callback functions runs a control or measurement task in the presented case.

Messages in the queue are handled in the order in which they appear. It means that the time when the controller function is started is disturbed even by low-priority system events like mouse movements.

The jitter histogram of one million samplings is given in figure 4. One can notice a huge deviation of sampling times. The desired sampling period was set to 10 ms but real sampling periods are distributed between 2 and 32 ms.

Also the multimedia timer allows scheduling periodic events. The advantage of the multimedia timer over the

WM_MESSAGE method is that the *timeGetDevCaps* function in the multimedia library detects the highest resolution of the timer for the given PC. Also the multimedia timer does not rely on the message queue to trigger the timer event. Instead it calls directly a callback function. The *timeSetEvent* function starts a timer and establishes a connection between the timer event and the callback function.

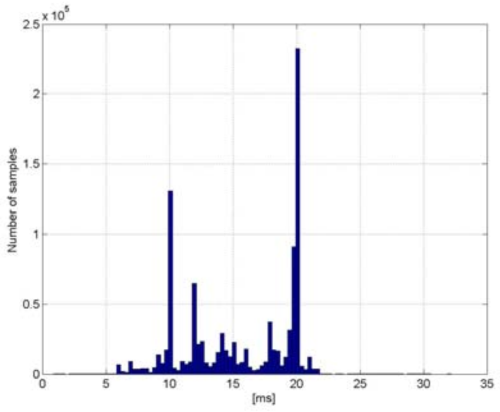

Fig.4. Jitter histogram of the WM_TIMER timer

Figure 5 shows the jitter histogram of the multimedia timer for the periodic task. The period is set to 10ms. One can notice that the samples are much closer aggregated than the samples in figure 4. But also some slower and faster samples appear with error higher than 1ms. Occasionally periods longer than 15ms are observed. Such disturbances appear mostly during intensive disk and network operations.

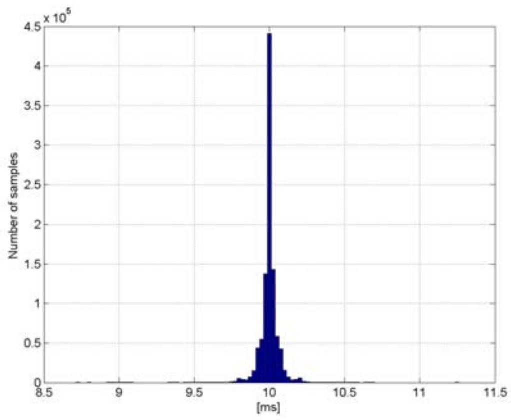

Fig.5. Jitter histogram of the multimedia timer

**Modified timer**

An example of disturbance of the timer period is shown in figure 6. When the operating system performs some time critical operations the timer procedure is delayed. It results in a longer period duration. In the next timer cycle the system generates shorter period to keep the average value of the timer period at the given level.

To overcome the jitter of the period the periodic timer can be replaced by the single shot timer with a period $T_S$ shorter than $T_0$ (see Fig.7). For the $kT_0$ time the timer is set to be triggered on $kT_0+T_S$. When timer event $t_{k+1}$ triggers the callback function the TSC counter is read until the time $(k+1)T_0$ is reached. Also the priority of the timer task is raised to the *THREAD_PRIORITY_HIGHEST* level to disable the operating system to preempt the callback function.

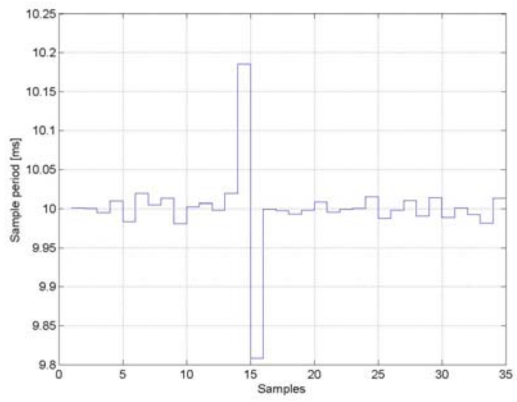

Fig.6. Disturbance of the timer event

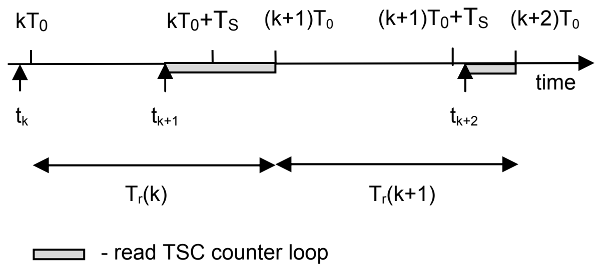

Fig.7. Operation of the modified algorithm.

If the difference $T_0$-$T_S$ is higher than maximum disturbance of the timer event the modified algorithm cancels the jitter of the sampling period.

The jitter histogram of one milion sampling periods of the modified timer is presented in figure 8. The sampling period $T_0$ has been set to 10ms and the $T_S$ to 7ms.

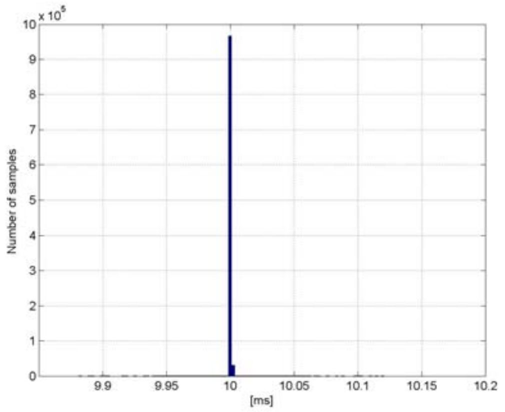

Fig.8. Jitter histogram of the modified multimedia timer

Table 1. Jitter parameters

| | Xilikernel | WM_TIMER | MM timer | Modified MM timer |
|---|---|---|---|---|
| Average period [ms] | 9,9999 | 15,6 | 9,9999 | 10,0003 |
| RMS [ms] | 2,54e-4 | 4,34 | 6,03e-2 | 2,53e-3 |
| $N_{10\mu s}$ | 100% | 1 | 37,59% | 99,71 % |
| $N_{100\mu s}$ | 100% | 7 | 93,71% | 99,91 % |
| $N_{1ms}$ | 100% | 22% | 99,96% | 99,999% |

Table 1 summarises parameters of the investigated timers. In all experiments the period is set to 10ms and one million samples is acquired and analysed. The average value and the RMS error of the period duriation are given. Also the number of periods with the duration error less than 10μs ($N_{10μs}$), less than 100μs ($N_{100μs}$) and less than 1ms ($N_{1ms}$) is shown.

**Real-time experiments**

The 3D crane setup is selected as an experimental model for real-time experiments (see Fig.9). The setup consists of a bridge and a cart moving along the rails. Also the length of the string of the payload is controlled. There are five measurements in the system: position of the bridge, position of the cart, length of the paylod string and two angles of the string attached to the payload. The aim of the experiment is to move the payload to a reference position and to damp the oscillations of the payload caused by motion of the bridge and trolley.

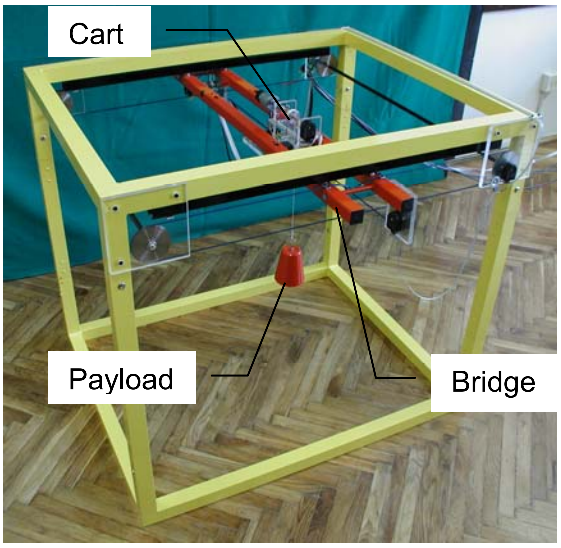

Fig.9. Experimental 3D crane setup.

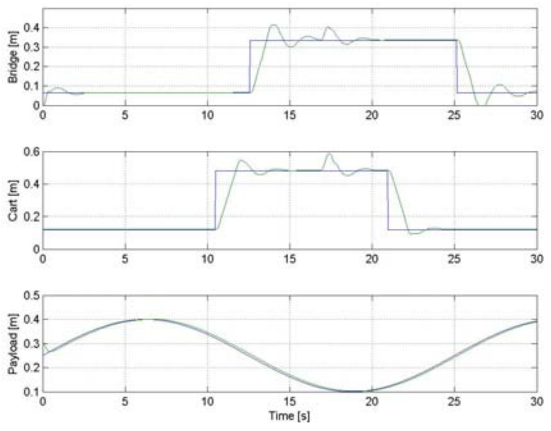

Fig.10. Positions of the bridge, cart and payload

The Windows 7 64-bit system is applied to control the 3D crane model. The software is running the modified multimedia timer algorithm. The results of the experimet are given in Fig. 10 and 11. Figure 10 illustrates the positions of the bridge, cart and the length of the payload rope. The positions are depicted in the green colour, the reference position is plotted as blue lines. Figure 11 shows the angles of the payload. The system is manually disturbed in 16th second. One can notice the system response to the reference position modifications and to disturbances.

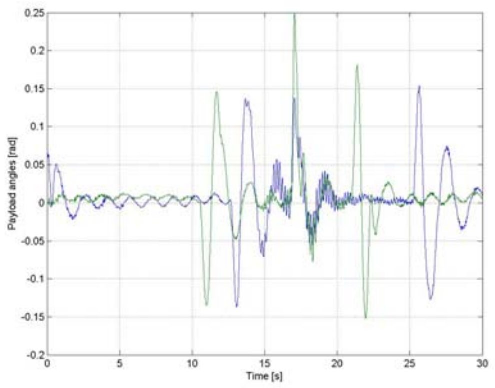

Fig.11. Angles of the payload

**Conclusions**

The currently most frequently applied systems for PC computers, Windows XP and Windows 7, are not equipped with the dedicated real-time measurement and control properties. The Windows priority management, task preemption rules and processing of timer events are not designed to be used in real-time setups. Nonetheless, as the systems are popular, users generate the pressure to apply Windows directly in measurement and control applications.

The presented multimedia timer processing algorithm tunes some parameters of the control process to improove the accuracy of the constant sampling period control tasks. The control task is executed every 10ms and controls the laboratory crane system. The task runs directly in the 64-bit Windows 7 environment.

It is important to emphasise that the presented timer algorithm does not transform the Windows into the hard real-time system. Critical control plants require real-time operating systems. Their principles of operation differ significantly from the Windows operating rules. However, the timer algorithm seems to open the way to apply the Windows in soft real-time measurement and control architectures for plants that tolerate some jitter in control algorithms.

REFERENCES

[1] OS Platform Statistics, http://www.w3schools.com/browsers/browsers_os.asp

[2] http://www.intervalzero.com/soft-control-architecture.htm - TRX homepage.

[3] http://www.on-time.com/rtkernel-32.htm - RTKernel homepage.

[4] Augustyn J. Bień A., Real time performance of USB interface in embedded control and measurement systems, Electrical Review, 2009, 85 nr 7, p1-7.

[5] Marti P., Jitter compensation for real time control systems, Proceedings of the Real-Time Systems Symposium, 2001.

[6] XILKERNEL Users Guide, http://www.xilinx.com, 2006

***Author***: *dr inż. Krzysztof Kołek, AGH University of Science and Technology, Institute of Automatics, al. Mickiewicza 30, 30-059 Kraków, E-mail: kko@agh.edu.pl.*