Soft Real-Time Acquisition in Windows XP

Justin Cinkelj, Matjaz Mihelj and Marko Munih

Faculty of Electrical Engineering,
University of Ljubljana, Ljubljana, Slovenia
{justin.cinkelj,matjaz.mihelj,marko.munih}@robo.fe.uni-lj.si

Abstract — *In the paper we present two possible approaches for a soft real-time acquisition under Windows XP. The first approach is based on a high priority thread and the second one on a local advanced programmable interrupt controller (LAPIC).*
Robustness was evaluated on unloaded and loaded system. In order to assess real-time system performance we analyzed sampling time histograms and maximum timing error. Both approaches perform satisfactory on unloaded system. On loaded system LAPIC approach shows better robustness. Although sampling time histograms show bigger errors on loaded systems than on unloaded systems, maximum timing error does not change significantly for the LAPIC approach.

1 Introduction

1.1 Real-time control and acquisition

Real-time operation means that some particular action as response to a given condition has to happen within given timing constrains. Condition can be fulfilled by event (event triggered systems) or by time (time-triggered systems) [1]. Here we will deal only with time-triggered systems.

An example of a time-triggered real-time system is control of a robot. Current position of the manipulator is first read via encoders. In the next step is desired excitation for actuators calculated by the control algorithm. Finally is excitation applied to power amplifiers, which drive robot's motors. The control algorithm must perform each step of the control loop within specified timing constrains. If timing requirements are not fulfilled, control performance will degrade and in worst case instability can occur [2], [3].

Timing constrains are given in terms of control delay, control period, jitter and transient error [4]. Control delay is the time between a sampling instance and its corresponding actuator output. It should be as small as possible for improved control performance. Control period is rate of actions and should be based on system dynamics. In a single-rate system is the control period equal to the sampling period. A smaller control period does not necessarily improve control performance – instead it should be within a range. Jitter is defined by IEEE as "time-related, abrupt, spurious (false) variations in the duration of any specified related interval" [5]. It arises because of clock drift, branching in the code, scheduling, communications and use of certain

hardware (cache memory) [6], [7], [8]. It should be as small as possible. Transient error can be seen as a special case of time variation. It is related (but not restricted) to hardware errors, which arise due to an internal or external fault. An undetected and uncorrected fault can lead to unpredictable malfunction, a system failure.

Real-time systems can be divided into hard and soft real-time systems. Hard real-time system is the one that must, without failing, generate a response to an event within a specified time window. Robot control requires use of hard real-time system, since malfunction could damage manipulator, other equipment or people. Soft real-time system is the one that should fulfill timing requirements. "Should" means that a system is allowed to occasionally miss the deadline.

Data acquisition systems are usually less susceptible to inexact timing. This makes them candidates for application of a soft real-time system [9], [10]. In an acquisition system timing requirements are described in terms of sampling period and jitter. Sampling period should again be based on system dynamics – Shannon's sampling theorem should be fulfilled. Jitter should be minimized. Figure 1 shows the error introduced by jitter. The sample is assumed to be acquired at moment t(i), but actually it is acquired at t(i)+jitter(i). Consequence is that acquired data lies at point B instead at A. This introduces error of ΔU(i). The same error is introduced if data is acquired at correct time t(i) and an inaccurate (unstable, noisy) sensor reads value at C instead at A. We see that unfulfillment of timing requirement can degrade acquired data the same way as use of an inaccurate measurement device.

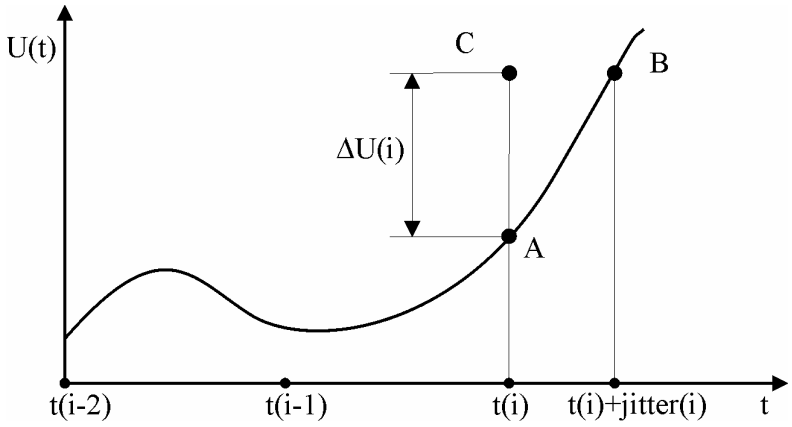

Figure 1: Jitter introduced error

1.2 Measurement in ALLADIN project

The ALLADIN project focuses on development of a measurement and analysis system for decision support in neuro-rehabilitation, in particular in stroke. Final product of the project should help predict final outcome of the therapy in early stages of the therapy and suggest best possible method of therapy.

Data is collected by multi channel isometric force/torque measurement using JR3 force/torque sensors (JR3 Inc., Woodland, California, USA). Each of 8 sensors transfers data to a personal computer (PC) as digital stream with 8000 samples per second. Digital data are received by JR3 receiver boards. The receiver boards provide host PC with processed data. Data are not buffered. Since receiver boards store only last sample, use of a real-time acquisition is required.

The acquisition system is running on Windows XP operating system (OS) (Microsoft, Redmond, Washington, USA), mainly because physiotherapists as the end users are familiar with it. However Windows OS does not support a real-time operation. Because of that a soft real-time support for Windows OS was implemented.

Required sampling frequency is 100 Hz. It is lower that sensor sampling frequency because in this measuring system forces are generated by a human, meaning that frequency range is below 30 Hz. Jitter has to be under 1 ms. The acquisition computer is expected to be loaded during acquisition with video playback, reading and writing files to and from hard disk and with network communication.

In the paper are presented two approaches for a soft real-time acquisition. One is based on a high priority thread and second one interrupts from LAPIC timer. Robustness was tested on an unloaded and a loaded system. Sampling time histograms and maximum timing error was used to assess performance of the implemented soft real-time acquisition systems.

2 Methodology

2.1 Hardware platform

Soft real-time operation was analyzed on 2 different computers. Table 1 lists their main characteristics.

	CPU type	CPU frequency	Motherboard	System bus frequency
System 1	Two Pentium III	500 MHz	ASUS P2B-DS	100 MHz
System 2	Pentium 4 with hyperthreading	2.8 GHz	Gigabyte GA-81848P	200 MHz

Table 1: Hardware platform

2.2 Implemented drivers for real-time work

In order to have a maximum control over the OS the real-time support was implemented in a driver. Driver does not need to use unpredictable Win32 API [11] and have a direct access to the hardware. Two approaches have been tested, the first one based on a working thread [12] and the second one on interrupts, generated by the local advanced programmable interrupt controller (LAPIC).

The first driver creates a working thread with the highest possible priority - HIGH_PRIORITY. The thread then waits in a busy-wait loop until the moment arrives to acquire a sample. Busy-wait loop would ideally last whole acquisition period. It controls the sample period and has also effect of a buffer (a trading delay for jitter) that minimizes jitter [4]. A thread with HIGH_PRIORITY priority can not be preempted by other threads. That requires use of a dual processor PC, otherwise the PC will appear to “freeze” until acquisition ends. Interrupts (both interrupt service routines - ISRs and deferred procedure calls – DPCs) can preempt the thread [13].

The second driver uses a local advanced programmable interrupt controller (LAPIC), located on a CPU. The LAPIC contains a timer, which can be programmed to deliver one shot or periodic interrupts to the CPU [14]. This timer runs with the frequency of the system bus and is not used by the OS. To program the timer it is re-

quired to bypass Windows OS and work directly with the hardware. Highest priority interrupt level (interrupt vector in range 0xF0 to 0xFF) is used, so that the LAPIC timer interrupt can interrupt interrupts of lower priority [15]. Delay can still occur if interrupts are disabled by reset of flag IF in EFLAGS register (use of CLI instruction).

Again a dual processor machine has to be used, this time because Windows disables the LAPIC on an uniprocessor systems. After that the LAPIC remains disabled until reboot [15].

2.3 Procedures

Real-time performance was evaluated on an unloaded and a loaded system. On loaded system a compressed video and 3D openGL graphics were simultaneously played, a file was copied from the network to the local disk and two instances of Matlab were using remaining processing time. Graphics was used because DMA transfers made by the graphic card can occupy the system bus when real-time task needs to transfer data over the system bus. This can cause delay in execution of the real-time task [16], [17]. Copying of a file causes interrupts from the network card and the hard disk and also DMA transfers.

Loaded system presents much worse conditions than those expected during use of the developed acquisition system. With loaded system we are trying to simulate worst case conditions. If system performs well under heavy stress, we can expect with higher probabillity that it will works on lightly loaded system too. This is to necesarilly to gain some safety margin.

Requested sampling frequencies were 5 and 10 kHz. Length of a single acquisition was 30 s. Time was measured with the RDTSC instruction – corresponding timer runs with the frequency of the processor core.

2.4 Analysis

Analysis is based on a maximum jitter and on histograms of the sampling period. Histograms are also summarized in tables which show percentage of samples outside of a given tolerance.

3 Results

Table 2 shows that both drivers perform well on unloaded systems. Almost no samples have jitter bigger than 10 μs.

	Tolerance [μs]					
	50	10	5	1	0.5	0.1
System 1 Thread	0.00	0.02	0.04	0.06	0.07	11.49
LAPIC	0.00	0.00	0.00	0.04	0.13	1.02
System 2 Thread	0.00	0.01	0.02	0.07	0.21	1.37
LAPIC	0.00	0.01	0.02	0.48	0.66	2.99

Table 2: Percentage of jitter outside of the tolerance, unloaded systems at 10 kHz

In Figures 2 and 3 are shown timing histograms for data acquired on the loaded system 1 with the sampling frequency of 10 kHz by the working thread and the LAPIC

driver, respectively. For the thread frequently samples with jitter of a few 100 µs occur. but most of the time is jitter bellow 100 µs. For the LAPIC driver is maximal jitter 11.3 µs, and typically below 5 µs.

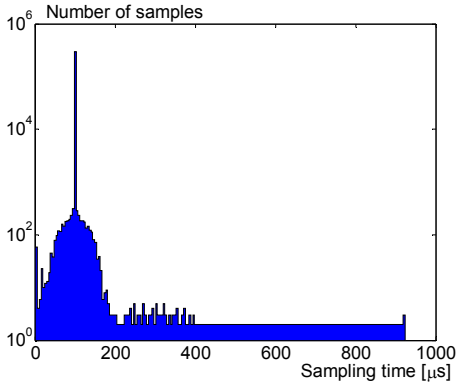

Figure 2: Loaded system 1, thread histogram at 10 kHz

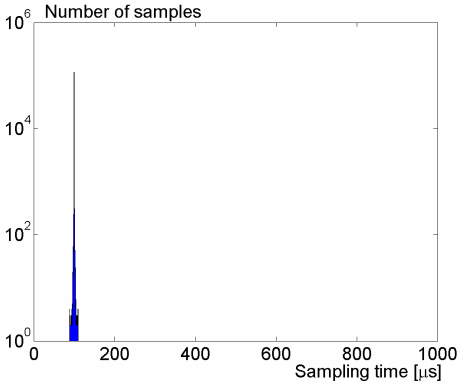

Figure 3: Loaded system 1, LAPIC histogram at 10 kHz

Figures 4 and 5 show the same timing histograms, but now for the loaded system 2. LAPIC has maximal jitter of approximately 40 µs, typically below 10 µs. Thread has worst jitter over 200 µs, typically up to 100 µs. Thread is again worse then LAPIC.

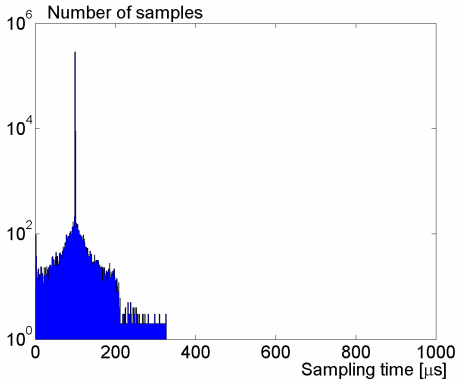

Figure 4: Loaded system 2, thread histogram at 10 kHz

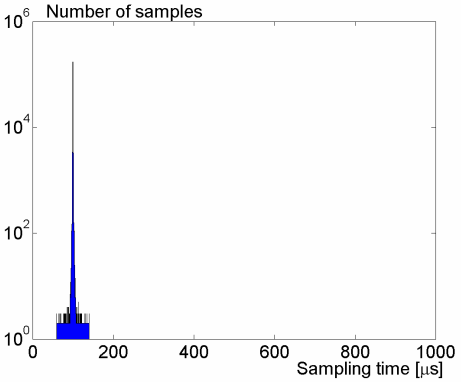

Figure 5: Loaded system 2, LAPIC histogram at 10 kHz

Distribution of sampling times shows on loaded systems better robustness of the LAPIC approach. Table 3 shows significant percentage of samples outside of 10 µs tolerance for the thread driver, but almost none for the LAPIC driver.

	Tolerance [µs]					
	50	10	5	1	0.5	0.1
System 1 Thread	0.15	0.99	1.17	1.35	1.37	12.66
LAPIC	0.00	0.00	0.01	1.03	4.85	36.51
System 2 Thread	0.46	1.39	1.66	1.99	2.78	15.87
LAPIC	0.00	0.01	0.03	1.91	7.94	26.15

Table 3: Percentage of jitter outside of tolerance, loaded systems at 10 kHz

Table 4 shows maximal jitter for all tests. Better performance of the LAPIC driver is seen once more. The thread driver on a loaded system has a much larger jitter than on an unloaded system. The difference unloaded – loaded system is for the LAPIC driver small and statistically not significant. Difference between 5 kHz and 10 kHz case is also not significant, except for the LAPIC driver on the unloaded system 1.

	unloaded		loaded	
	5 kHz	10 kHz	5 kHz	10 kHz
System 1 Thread	45.2	53.6	973.9	823.6
LAPIC	3.4	11.2	12.6	11.3
System 2 Thread	20.1	27.0	184.6	227.2
LAPIC	45.2	46.6	35.2	40.6

Table 4: Maximum jitter in μs

4 Discussion

The LAPIC driver does not only give a lower average jitter than the thread driver, but is also less sensitive to the loading of the system. Jitter of the LAPIC driver could be further minimized by the use of a buffer (trading delay for jitter), implemented as a busy wait loop.

When interpreting the results, we have to be aware that a different loading of the system could significantly change results. Loaded system was only a simulation of a worst case load. Timing can be also affected by a prolonged masking of interrupts in ill-behaved drivers. This would degrade performance of both approaches.

The LAPIC driver has to use an interrupt vector, which is not used by the Windows. If chosen interrupt vector is used by the OS, some interrupts will be delivered to the LAPIC ISR instead to the OS. Such situation will most likely cause a crash. Selection can be verified by trial and error, but it can not be guarantied that it is not used by only rarely occurring interrupt.

Acknowledgments

This work was supported by the ALLADIN project, funded by the European Commission under the 6th Framework Programme, IST Contract No.: IST-2002-507424, and by Ministry of Higher Education, Science and Technology, Republic of Slovenia.

References

[1] M. Törngren. Fundamentals of implementing Real-Time Control Applications in Distributed Computer Systems. *Journal of Real-Time Systems*, 14(3): 219-260, May 1998.

[2] P. Marti, J.M. Fuertes, G. Fohler, K. Ramamritham. Jitter compensation for real-time control systems. *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, pages 39-48, London, UK, December 2001.

[3] P. Marti, R. Villa, J.M. Fuertes, G. Fohler. On Real-Time Control Task Schedulability. *Proceedings of the European Control Conference,* pages 2227-2232, Porto, Portugal, September 2001.

[4] M. Sanfridson. Survey of Techniques for Handling Timing Problems in Distributed Control. *Compendium of Papers SNART'99*, Linkping, Sweden, August 1999.

[5] *The new IEEE Standard Dictionary of Electrical and Electronic Terms*. Institute of Electrical and Electronics Engineers Inc, New York, USA, 5th edition, 1992.

[6] J. Liedtke, H. Härtig, M. Hohmuth. OS-Controlled Cache Predictability for Real-Time Systems. *Proceedings of the Third IEEE Real-Time Technology and Applications Symposium*, pages 213-227, Montreal, Canada, June 1997.

[7] M. Di Natale, J. A. Stankovic. Scheduling Distributed Real-Time Tasks with Minimum Jitter. *IEEE Transactions on Computers*, 49(4):303-316, April 2000.

[8] C.-G. Lee, K. Lee, J. Hahn, Y.-M. Seo, S. L. Min, R. Ha, S. Hong, C. Y. Park, M. Lee, C. S. Kim. Bounding Cache-Related Preemption Delay for Real-Time Systems. *IEEE Transactions on Software Engineering,* 27(9):805-826, September 2001.

[9] EA. Clancy. A PC-based workstation for real-time acquisition, processing, and display of electromyogram signals. *Biomedical Instrumentation & Technology*, 32(2):123-34, March-April; 1998.

[10] H. Shimakawa, H. Ohnishi, I. Mizunuma, M. Takegaki. Acquisition and service of temporal data for real-time plant monitoring. *Proceedings of the Real-Time Systems Symposium*, pages 112-118, December 1993.

[11] J. W. S. Liu. *Real-Time Systems*. Prentice–Hall Inc., New Jersey, USA, 1st edition, 2000.

[12] C.-H. Lin, H.-H. Chu, K. Nahrstedt. A Soft Real-time Scheduling Server on the Windows NT. *Proceedings of the 2nd USENIX Windows NT Symposium*, pages 149–156, Seattle, Washington, August 1998.

[13] A. Baker and J. Lozano. *The Windows 2000 Device Driver Book: a guide for programmers*. Prentice-Hall Inc., New Jersey, USA, 2nd edition, 2001.

[14] H.-P. Messmer. *The Indispensable PC Hardware Book.* Addison-Wesley, Boston, USA, 4th edition, 2002.

[15] IA-32 Intel® Architecture Software Developer's Manual. http://developer.intel.com/design/pentium4/manuals/index_new.htm

[16] S. Schönberg. Impact of PCI-Bus Load on Applications in a PC Architecture. *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, pages 430-440, Cancun, Mexico, December 2003.

[17] T.-Y. Huang, J. W.-S. Liu, D. Hull. A Method for Bounding the Effect of DMA I/O Interference on Program Execution Time. *Proceedings of the 17th IEEE Real-Time Systems Symposium,* pages 275-287, Washington DC, USA, December 1996.