# Real-Time Measurement Systems Based On LabVIEW Graphical Environment

**Arkadiusz Gancarczyk**, *Silesian University of Technology*
(dr hab. inż. Lesław Topór-Kamiński, prof. Pol. Śl., *Silesian University of Technology* )

**Abstract**

The subject of the present paper is to discuss the problems concerning the measurement of systems fulfilling the requirements of the real time. In the thesis we deal with the application, the nature and the architecture of the systems which are important and can be area of research. In many applications it may be required that a certain periodic schedule be executed within a hard-timed loop. These systems share a common feature in that they interact with devices or processes that themselves operate based on real-world time. Such systems are often termed "hard real-time systems" because their actions must meet time constraints imposed by the application space, rather than by the operation of the measurement or control devices.

LabVIEW graphical programming includes tools for low-level system debugging and precise execution timing so that it can increase the flexibility and functionality of deterministic real-time application. Design real-time application, based on NI LabVIEW, benefit from the rapid application development of graphical programming. This solutions offers the widest variety of high-performance distributed real-time execution and headless operation such as control processing, data logging, real-time data acquisition or data analysis.

## 1. Introduction

Examples of such hard real-time systems are more and more. In the measurement world, complex test systems composed of many electronic instruments operating in concert are used to verify the performance of even more complex electronic or electromechanical devices. In the field of control, combinations of computers, controllers, sensors and actuators collaborate, great importance has regulate and control processes or communications systems operate to pass information from source to destination. A hard real-time system is one that must meet its performance objectives every time and all the time. As soon as one of these systems does not meet one of its performance criteria, it fails. An example of a hard real-time system is a fly-by-wire flight control system, where if the system does not respond to a pilot's commands within microseconds, then the system fails with potentially catastrophic circumstances. Our everyday life seems to be governed by the clock. Real-time application appear in the field such us :

- in-vehicle data acquisition, data logging and control;
- machine condition monitoring and protection;
- embedded system prototyping;
- remote and distributed monitoring;
- embedded data logging;
- custom multi-axis motion control.

In programming these devices, the time which something is to happen is entered as well as the time at which the activity is to cease or in some cases, the duration of the activity. Never is there a list of activities executed based on the speed at which the device operates.

The limitation of existing systems is that their architecture makes it hard to offer control strategies that require dynamic adaptation. For example, deployment of sophisticated control strategies for high accuracy, such as multiple-input and multiple-output control, requires a precise identification of the system under control. In many cases, optimal implementation lacks flexibility and tends to require fairly sophisticated hardware design techniques.

## 2. LabVIEW Real-Time Module

LabVIEW Real-Time extends the LabVIEW graphical development environment to deliver deterministic, hard real-time performance. The LabVIEW RT architecture consists of the following three components: LabVIEW, the Real-Time Development System and the Real-Time Engine. In real-time applications the code-part of the vi requiring absolute reliability and extended duration run time. Develop application (fig.1) will be transfered to the over Ethernet to Real-Time Engine to run on a variety of dedicated hardware targets

with a real-time operating system (RTOS), whereas the user-interface will remain on the LabVIEW PC. When the program has started running, the PC with the user-interface can be turned off and on again at a later time. The communication between the real-time hardware and the originating PC or any other PC, can be achieved in several ways.

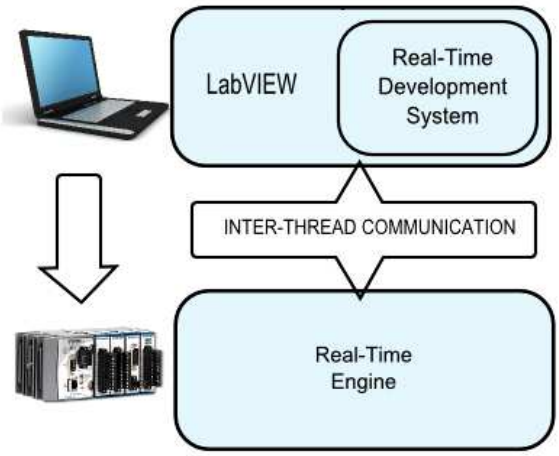


**Fig.1. Components of LabVIEW Real-Time Architecture**

LabVIEW offers two main differences from other solutions: signals are always present and programs are represented as diagrams. Furthermore, causality violations are easily avoided if proposed design principles are followed. Additionally compared to automated and efficient memory management, which prevent memory leaks and a dataflow-based environment. The LabVIEW Real-Time Module provides native tools for debugging application. The Real-Time System Manager can be use to monitor system resources such as CPU and memory usage of your real-time target. With other debugging tools, keep track of memory buffer allocation and the amount of memory consumed by each VI as it downloads to target. Additionally, LabVIEW Execution Trace Toolkit is used to advanced debugging to visualize the task execution of application and control assign the appropriate execution priority to each real-time task. The embedded RTOS then uses a combination of round-robin and preemptive scheduling to ensure deterministic execution of time-critical tasks. Additionally, can be implements multirate applications to include independent tasks running at unique priorities.

One feature of LabVIEW is that the real-time runtime environment and the simulation environment use the same compiler and scheduler, which facilitates embedded control systems development, since usually in this domain extensive simulations are performed, including hardware in the loop tests, before a first prototype.

## 3. Example architecture real-time control system

One of the most popular in the last time to design real-time measurement system are programmable automation controller (PAC). Examples is CompactRIO which bases on new reconfigurable I/O (RIO) technology and is combines the processing power and flexibility of a field-programmable gate array (FPGA) with the reliability of a real-time processor. Include three components:

- processor to execute LabVIEW Real-Time applications for reliable real-time operating system;
- reconfigurable embedded chassis with programmable FPGA core that can be accessed and configured using LabVIEW graphical development software;
- hot-swappable industrial I/O modules with built-in signal conditioning for direct connection to a variety of sensors and actuators.


**Fig.2. Components of CompactRIO**

ComapctRIO is attractive solution to design hard real-time system which require timely decisions to be made based on incoming data. For instance, I/O device samples an input signal and sends it directly to memory. Then, the processor must analyze the signal and send the appropriate response to the I/O device, records data over time and sends results to PC station to measurement system of operator. In this application, the software must be involved in the time-loop. Detailed knowledge of the specific hardware and topology providing develop application to direct access to each I/O module for precise control and flexibility in timing, triggering, and synchronization. For design hard real-time control system, very important is choose appropriate architecture program which depend from specification control system and possibility elements of dedicated hardware. Figure 3 showed all possibility components which can be used for develop control system. By analyzing controller operations, can break down the system into smaller components, each responsible for a specific task in the overall application. Some of these components are ready-to-run as part of the machine control reference architecture, while others must be developed as part of the design and implementation of a specific machine control application.

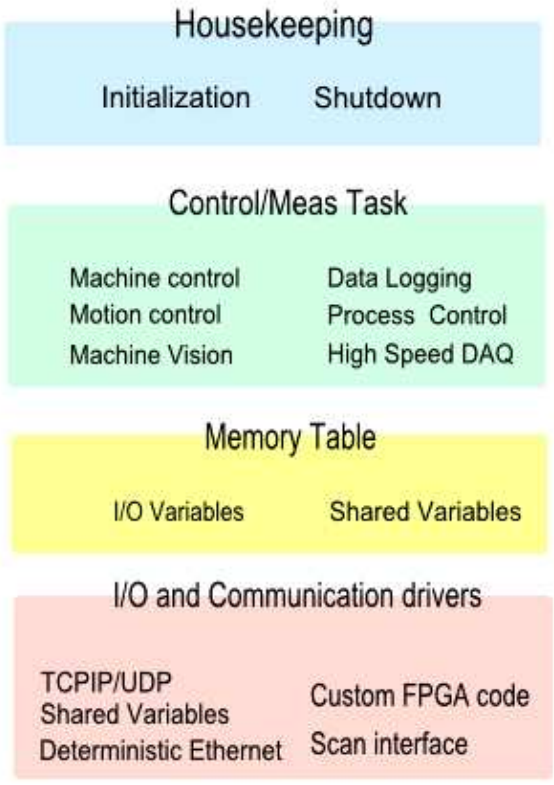


**Fig.3. Controller Architecture and Components**

Knowing an architecture to acquire data and perform analysis, control and develop applications by programming is not sufficiently. Invaluable is effective management all available resources. It will be able to effectively:

- determine if a real-time solution is appropriate for a given problem;
- implement a deterministic and reliable application;
- reduce the jitter in a real-time application;
- choose an appropriate communication method;
- benchmark application;
- control timing, synchronization and priority of operations;
- create deterministic control and simulation solutions on the NI LabVIEW platform.

Building complex systems requires an architecture that allows code reuse, scalability and execution management. The time require to take action after an event is known as responsiveness and different control applications have different tolerances, varying from microseconds to minutes. Most industrial applications have responsiveness requirements in the milliseconds to seconds range. An important design criterion for a control application is the required responsiveness because this determines the control loop rates and affects I/O signals, processor and software decisions. Most controllers use a single processor to handle all control, monitoring and communication tasks. Because there is a single resource (processor) with multiple parallel demands, therefore need a way to manage the demands that are most important. LabVIEW make possible setting critical control loops to a high priority. Due to that compactRIO is a full-featured controller that still exhibits good determinism and responsiveness.

For real-time solution, improving software performance is indispensable. Most performance bottlenecks are caused by architectural and design limitations that require significant amounts of recoding to overcome. The following, however, are a couple of techniques that have been used successfully. Improving software is realizing by:

- performance profiling and algorithm improvements;
- control and management memory;
- timing and scheduling;
- I/O optimization techniques ( for example FPGA programming for compactRIO );
- application architecture and multitasking design.

Multitasking application is more challenging than debugging a single-tasking application for the simple reason that, in a multitasking application, more that one thing is happening at a time. Tasks interact with one another reading and writing global memory, acquiring and releasing synchronization objects and sometimes in those places where the activity of one task affects the activity of another, the two tasks can trip one another up. Add more tasks to the mixture which has more opportunity for trip-ups.

## 4. Multicore programming in real-time application

Solutions based on LabVIEW Real-Time offer variety of real-time hardware targets that contain an embedded processor running a real-time operating system for maximum reliability and deterministic performance. PXI industrial platform or desktop PC, dedicated to RTOS, takes additionally advantage of multicore performance for create deterministic applications. LabVIEW Real-Time Module implement high-performance real-time applications on multi-core systems and take advantage of high-performance multi-core processor technology. To further increase the performance and reliability of a real-time system, LAbVIEW Real-Time can be by programming assign to specific processor cores and dedicate them to execute a time-critical control thread and isolate it from less important thread that run on different cores (fig.4). Multithreading extends the idea of multitasking into applications, so can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel.

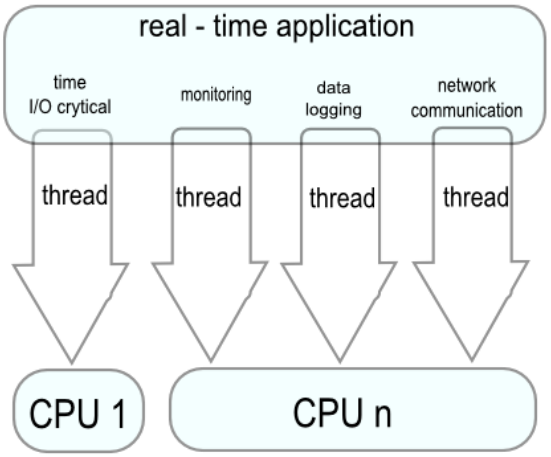


**Fig.4. Dedicating thread real-time system to a specific processor**

Applications that take advantage of multithreading have numerous benefits, including the following:

- more efficient CPU use;
- better system reliability;
- improved performance on multiprocessor computers.

So, it becomes expedient task to investigate hardware and software solutions for scheduling of real-time applications on a multithreaded processor. Scheduling technique is put on hard real-time constraints for one or more threads and a virtually unbounded number of soft real-time threads. The concept should be extended for use in a multi-core processor. This includes a balanced assignment of threads to cores, thread migration between cores and power saving by switching off cores separately.

The scheduling not only distributes computation time to threads, the scheduling decision must also enclose the temporal assignment of resources to threads.

## 5. Conclusions

Performance measures for real time systems did not receive much attention. An assessment of the few existing benchmarking methods for real time systems will reveal that they are highly inappropriate. Taking a closer look into the fundamental issues of real time systems it will become clear, that for hard real time systems qualitative characteristics are much more important than quantitative measures. Rigorous methods for design and implementation of safety critical real-time systems are vital to avoid loss of human lives or severe economic losses. Unfortunately, many of these systems are designed and evaluated using ad-hoc techniques. There are, on the other hand, relatively well developed theories for modeling and analysis of timing and reliability. These theories are, however, seldom applied in industry for system development, mainly because of the simplifying model assumptions and lack of appropriate tool support. Typical real-time system development is a multistep process that includes code programming, debugging, compiling, downloading and deploying. LabVIEW Real-Time is unique because it creates traditional real-time applications but takes advantage of all the benefits of LabVIEW for Windows graphical programming development environment. As real-time software is developed, performance must be measured at each step of the way. The execution speed of each component needs to be measured to ensure that they meet their performance criteria. During integration, the performance of the system needs to be continually measured for the same reason. High fidelity software performance measurements may be achieved by using a combination of source code instrumentation and hardware data collection. In addition to performance analysis, this technology may also be used to monitor dynamic memory usage, multitasking and multithreading behavior. Good area of research will be search appropriate the qualitative design methodology to narrow the gap between research results and industrial practice in evaluation and design of real-time systems measurement based on new solutions such us LabVIEW.

## Bibliography

[1] Gancarczyk Arkadiusz: *System pomiarowy czasu rzeczywistego do zdalnej diagnostyki układu hamulcowego pojazdu szynowego*, Mat. Konf. „Podstawowe Problemy Metrologii", Sucha Beskidzka, 2009.

[2] Gancarczyk Arkadiusz: *Mobile System on LabVIEW Environment Based on CompactRIO*, The Seventh International Students' Workshop Control & Information Technology – IWCIT'08, Glwice, wrzesień 2008, str 8-13;

*[3]* National Instruments *www.ni.com/crio*

*[4]* National Instruments *www.ni.com/rt*

*[5]* National Instruments *www.ni.com/multicore*

**Author:**

mgr inż. Arkadiusz Gancarczyk
Politechnika Śląska
Instytut Metrologii, Elektroniki i Automatyki
ul. Akademicka 10
44-100 Gliwice
tel. (032) 232 28 00
email:
*arkadiusz.gancarczyk@pols.pl*