Measurement of Real-Time Aspects of Simatic® PLC Operation in the Context of Physics Experiments

Harald Kleines, Janos Sarkadi, Frank Suxdorf, and Klaus Zwoll

***Abstract*—Today, most slow control systems for physics experiments at Forschungszentrum Jülich are implemented with Programmable Logic Controller (PLC) technology and fieldbus systems. In many cases, even deterministic response is required from the PLCs. This raises the question about the real-time performance that can be expected from a PLC. Response-time measurements of Simatic® PLCs—manufactured by the world market leader Siemens—are presented. Influence of program structure and hardware configuration on performance and deterministic behavior of a PLC is discussed.**

I. Programmable Logic Controllers (PLCs) in Experiment Control Systems

TODAY, industrial automation technology is well established in infrastructure systems for physics experiments, e.g., in water or gas supply systems. This leads to the heavy use of Programmable Logic Controllers (PLCs), which typically are the intelligent automation stations forming the core of industrial systems [1]. Main reasons include

- low prices induced by mass market;
- robustness;
- long term availability and support from manufacturer;
- professionality (connectors, conformance to standards,…)

Beyond the scope of pure infrastructure systems, PLCs are increasingly becoming central components of experiment control systems, replacing VME- or PC-based real-time systems [2], [3]. This is caused by the following features of modern PLC families.

- High degree of scalability: Modern PLC families have a wide spectrum of CPU types, that is scalable not only with regard to performance, but also with regard to functionality and form factor. For outdoor or fault tolerant applications special versions are available.
- Extensibility: The modular design of PLCs enables the extension with a wide range of digital and analog I/O modules. Additionally, integrated technology modules are available for different application areas, e.g., stepper motor controllers, servo motor controllers, or PID controllers.
- Extensive communication capabilities: Modern PLCs have at least one integrated communication port and can be extended by a variety of communication controllers for different field and process bus systems, thus enabling connection of other industrial devices. A key issue is the extension of a central PLC system with decentral periphery via special fieldbusses (e.g., PROFIBUS DP), that allows the transparent connection of "unintelligent" I/O-modules. Thus a PLC program can access this decentral periphery in the same way as central PLC periphery.
- Powerful development environment: Modern PLC families come with a homogeneous cross development environment, that supports all the major IEC 1131 programming languages [4]. Typically, representations in instruction list (IL), function block diagram (FBD) or ladder diagram (LD) can be switched dynamically. The development tools allow semigraphical hardware configuration, offer strong debugging mechanisms and allow incremental development by the exchange of blocks during runtime.

Today, in FZ Juelich, all new and advanced experiment control systems are heavily PLC-based [3], as illustrated by the architecture of a neutron spectrometer control system shown in Fig. 1.

Because the world market leader Siemens dominates the European market, Simatic® S7 PLCs are used in FZ Juelich, almost exclusively. The midrange series S7-300® is most popular. The high-end series S7-400® is targeted at applications with extreme performance requirements and supports also multiprocessor configurations. The mini PLC series S7-200® is rarely used, because it got the name S7 by pure marketing reasons and its programming environment is incompatible to the other S7 devices. Instead of the S7-200®, the IM151/CPU serves as a mini PLC. The IM151/CPU is an intelligent controller for the decentral periphery family ET200S®. Also the decentral periphery systems ET200L® and ET200M® are used commonly in Jülich. The SoftPLC WinAC® has only been tested in the Lab, so far.

The responsible planning of PLC-based control systems requires knowledge on their real-time features.

- What is the magnitude of PLC response time, depending on PLC type?
- Can deadlines be guaranteed?
- What programming rules have to be followed?

The paper addresses these issues by measurements at different PLC types of the Simatic® S7 family. The standard IEC 1131 defines a common framework for PLC functionality and programming languages [4], which all the major PLC manufacturers conform to. Thus general results can be generalized also to their PLC families.

Manuscript received May 16, 2003; revised October 1, 2003.

The authors are with Zentrallabor für Elektronik, Forschungszentrum Jülich, D-52425 Jülich, Germany (e-mail: h.kleines@fz-juelich.de).

Digital Object Identifier 10.1109/TNS.2004.828504

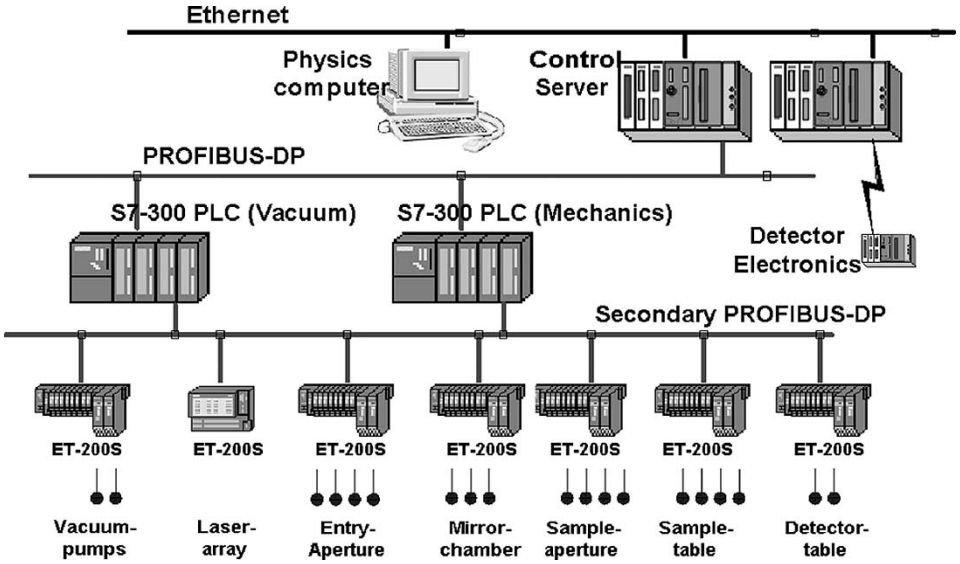

Fig. 1. Control system architecture of the neutron spectrometer KWS3.

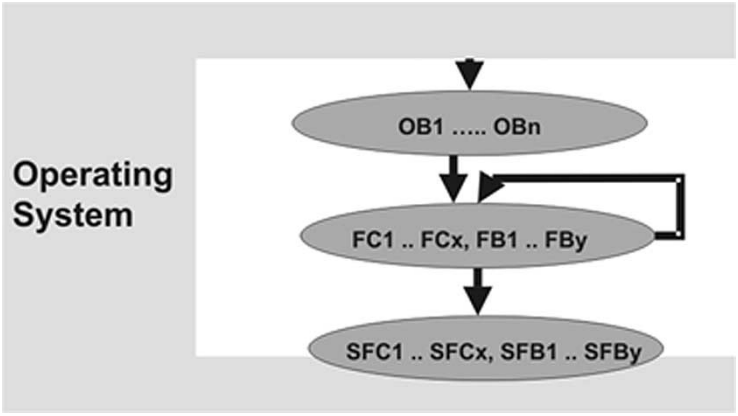

Fig. 2. Block calling hierarchy.

II. SIMATIC® S7 PROGRAMMING MODEL

Classical real-time applications in research are implemented with real-time kernels like OS-9 or VxWorks, that follow an asynchronous parallel programming approach, as defined in POSIX [5], for example. The software developer structures his program in tasks according to the logical structure of the problem to solve. These tasks are executed quasiparallel by the operating system, and the execution is basically event-triggered. By assigning priorities to the tasks the programmer gives hints to the operating system about the desired execution order. Thus the programmer does not have to plan the scheduling details. On the other hand it is difficult to understand the execution order and to decide, if a specific task can meet its deadlines.

The programming mechanisms in PLC systems are totally different, because they follow the older approach of synchronous programming [6]. Here, the execution of tasks is completely time-triggered, and the programmer has to organize his program into tasks according to the time, when a task has to run. So he must plan the execution order himself, which is more complicated but also gives more control.

In Step7, the development environment of the S7, all code exists in blocks, as defined in IEC1131. Tasks are represented by Organization Blocks (OBs). OBs are the schedulable items, that are called by the operating system of the PLC at certain events, e.g., when a timer expires or an error occurs. Thus, the OBs are the interface of the operating system to the user program. As indicated in Fig. 2, OBs can call Functions (FCs), which are blocks that correspond to functions in a procedural language. FCs can call other FCs or system functions (SFCs), which correspond to operating system calls in a POSIX environment. Function Blocks (FBs)/System Function Blocks (SFCs) are FCs/SFCs with an assigned data block for static function data.

A "normal" PLC program is contained in OB1, which is called cyclically by the operating system, as indicated in Fig. 3. Before OB1 is called the operating system transfers data from the input modules to a memory area called process image table. After OB1 has been called, data from the process image table is copied to the output modules. The indirect access to I/O-modules via the process image table reduces access time and increases consistency.

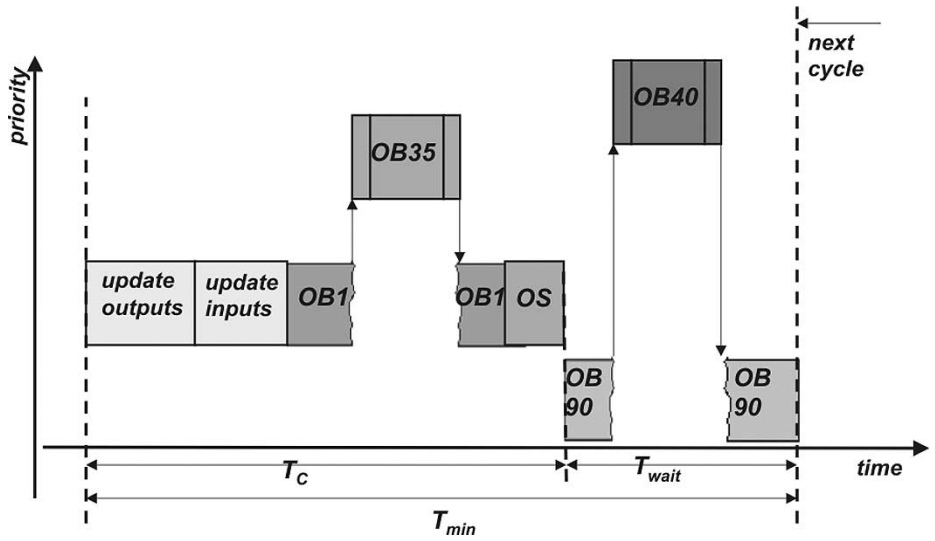

Fig. 3. Execution of main program scan cycle OB1.

The execution time of OB1 is monitored, and if a preconfigured maximum $T_{\max}$ is exceeded, the time error OB80 is called.

On S7-400® and WinAC® also a minimum $T_{\min}$ for the cycle time of OB1 can be configured. If the execution time T_C for OB1 is less then $T_{\min}$, the background OB90 is called, which has the lowest priority. The priority of all other OBs increases with its number. Only on S7-400® and WinAC® this default priority can be changed. Each OB can be interrupted by OBs with a higher priority. Table I lists the possible OBs. Availability of OBs depends on the CPU type. If more OBs of a certain type are required, a more expensive CPU has to be bought.

Time-of-day interrupt OBs are started at a preconfigured time, e.g., end of a shift, whereas time delays interrupt OBs are started at the expiration of a one-shot-timer. Cyclic interrupt OBs are started with a fixed frequency. The time interval and the phase offset can be configured with a granularity of 1 ms. Hardware interrupts OBs are started by a an event at an input or function module, e.g., detection of the rising edge of a digital signal. This functionality is only available with so-called "High Feature" input modules. Synchronous error OBs are started by errors in the user program, whereas asynchronous error interrupt OBs are started by PLC faults, like power failure, module failure or time errors. A time error occurs, when an OB cannot meet its scheduled start time, and is an unique feature of PLCs.

TABLE I
OB TYPES OF A SIMATIC® S7

background cycle	OB90
main program scan	OB1
time-of-day interrupts	OB10,..,OB17
time delay interrupts	OB20,..,OB23
cyclic interrupts	OB30,..,OB38
hardware interrupts	OB40,..,OB47
multicomputing interrupt	OB60
asynchronous error interrupts	OB80,..,OB87
startup	OB100, OB101
synchronous error interrupts	OB121,OB122

TABLE II
PLCS UNDER TEST

PLC CPU	**IM151/ CPU**	**314C-2DP**	**CPU 412-2**	**CPU 414-1**
PLC Series	ET200S	S7-300	S7-400	S7-400
T(float addtion)	~42µs	~4µs	~0.6µs	~0.7µs
Cyclic interrupts	OB35	OB35	OB32, OB35	OB32,33,34,35
HW interrupts	OB40	OB40	OB40, B41	OB40,41,42,43
Price in 2002	~500 €	~1400 €	~2000 €	~2700 €

III. REAL-TIME PERFORMANCE MEASUREMENTS

A. Performance Evaluation Goals

A key issue of real-time performance is the reaction time to external events. Because PLC systems basically conform to a synchronous programming model, this is directly determined by the cycle time T_C, which has to be analyzed for the free running cycle OB1 and cyclic interrupts; e.g., OB35. To determine the application area of PLC classes, the minimum of T_C has to be measured for different PLC types. The actual value of T_C in a specific application depends on the amount of code in the cyclic OB, of course.

Deterministic behavior of a PLC is determined by the jitter of T_C, which is of primary interest for cyclic interrupts.

With "High Feature" input modules, also hardware interrupts are possible. Here the minimum response time T_R, which is defined as the time to activate OB 40, and its jitter has to be measured.

In order to get a complete picture, the measurements have to be conducted for a mini PLC, a midrange PLC and a high-end PLC. Table II shows the CPUs, that have been selected for the measurements in this paper. The time for a floating point addition has been measured by repeating it 10^6 times, in order to convey an impression of their relative performance, which differs considerably.

Because of the distributed nature of PLC-based systems, the impact of PROFIBUS communication to response time is an im-

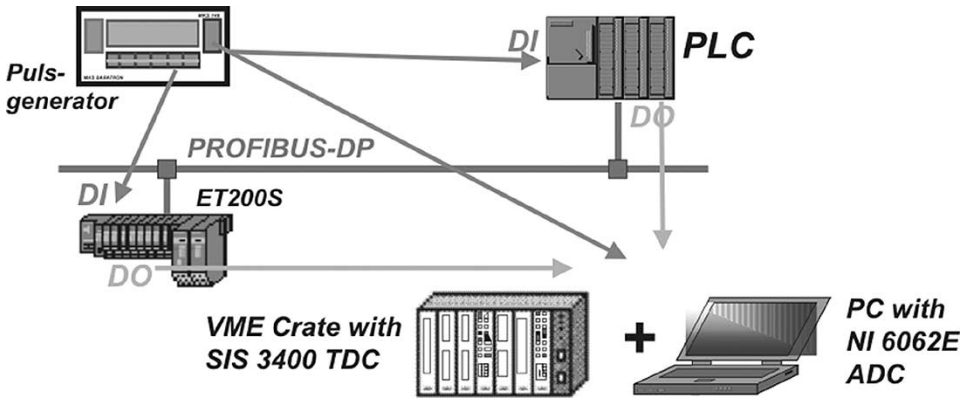

Fig. 4. Experiment setup.

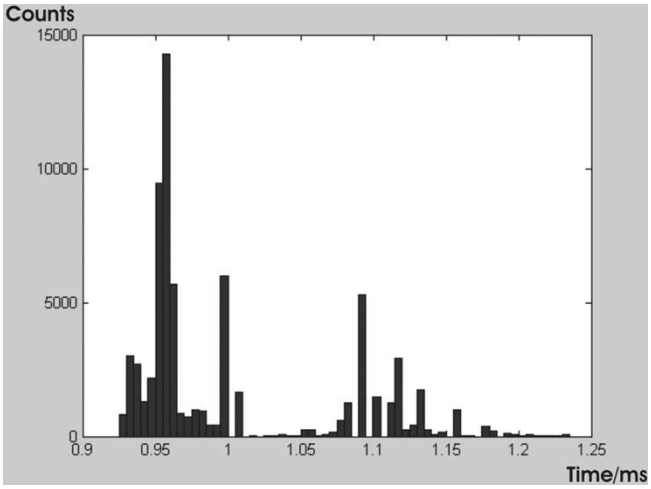

Fig. 5. Histogram of OB1 cycle time at IM151/CPU.

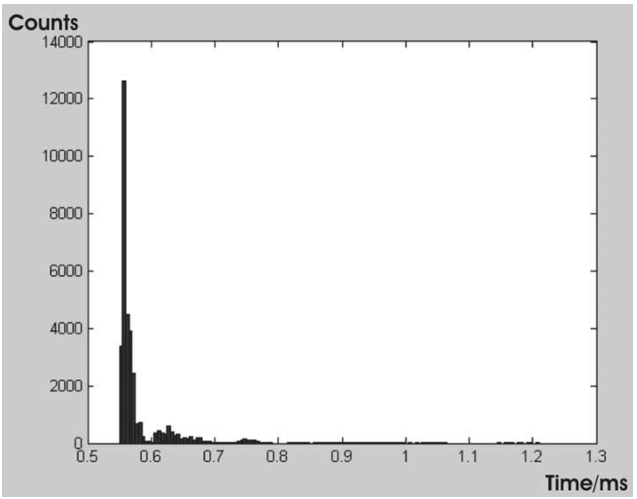

Fig. 6. Histogram of OB1 cycle time at CPU314C-2DP.

portant issue. Thus the additional delays introduced by communication as well as the additional jitter have to be analyzed. But presentation of communication-related measurements would go beyond the scope of this paper and will be covered by a future publication.

The analysis of the response time as a function of the system load, e.g., induced by harddisk activity, communication or background computing, is a key issue on conventional POSIX-like real-time systems. This is not an issue on PLC systems because of their synchronous cyclic operation. Even the communication on fieldbusses like PROFIBUS DP V0 or AS-Interface is cyclically, thus inducing a constant load. For asynchronous type of communication, e.g., TCP/IP, intelligent communication controllers are used, thus offloading the CPU. An exception from this rule is the MPI (Multipoint Interface, a proprietary fieldbus), that is integrated in each CPU. But in Jülich MPI is only used for programming.

B. Measurement Scenario

According to Fig. 4 the inputs of the PLCs under test (listed in Table II) are connected to a pulse generator. OB40 is activated by a rising edge of the input signal and toggles a output signal. The output of the pulse generator and outputs of the PLC under test are connected to the National Instruments analog input module NI6062 E. The signals are sampled with a frequency of 100 kHz. Matlab code has been developed that detects rising edges in the sampled signals, computes the required time differences and forms an histogram of the measured data. Thus the distribution of the response time T_R of the PLC can be measured. The distribution of the cycle time T_C is measured in an analogous way. Alternatively, the signals are connected to the TDC module SIS 3400 from Struck Innovative Systems. Thus the correctness and the sufficient precision of the measured data could be verified.

C. Measurements of the Main Program Scan Cycle OB1

Figs. 5–7 show the distribution of T_C for OB1 measured on the first three PLC in Table I. There was no other activity on the system than OB1, which only contained code for toggling a digital output directly without using the process image table.

The minimum for T_C and its jitter are caused by operating system activities and get better with increasing performance of the PLC under test. Although me value and variance of T_C for the S7-300® are much better than for IM151/CPU the worst case is comparable. CPU412-2 is almost determistic, basically taking two discrete values. This is not necessarily caused by CPU behavior, because at these frequencies the behavior of the digital outputs gets significant, too. This caused artifacts on the CPU414-1 where we measured a minimum cycle time of 0.2 ms. But in this situation the time between two state changes

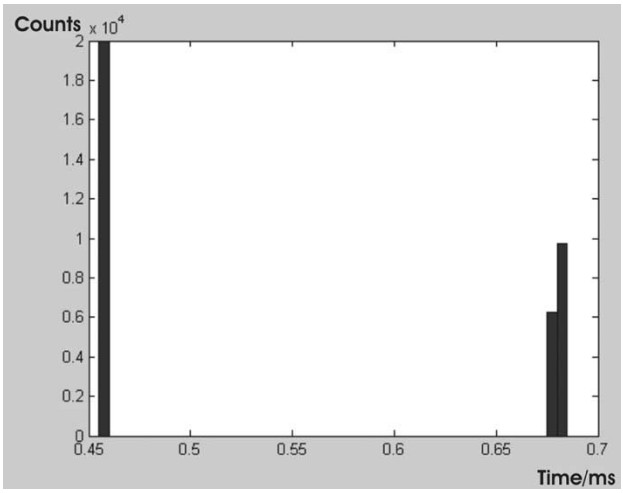

Fig. 7. Histogram of OB1 cycle time at CPU412-2.

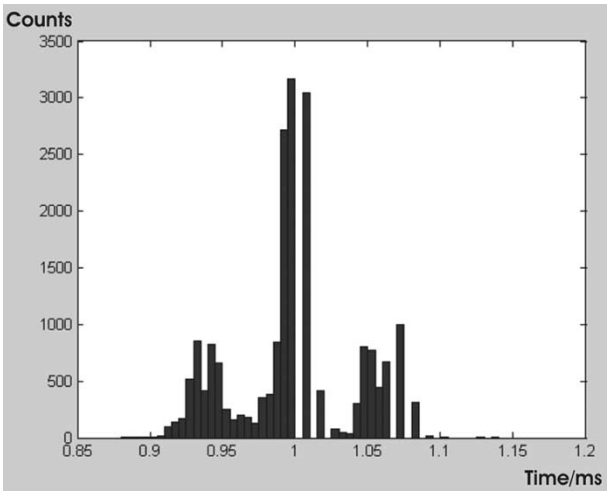

Fig. 9. Histogram of OB35 cycle time at CPU314C-2DP.

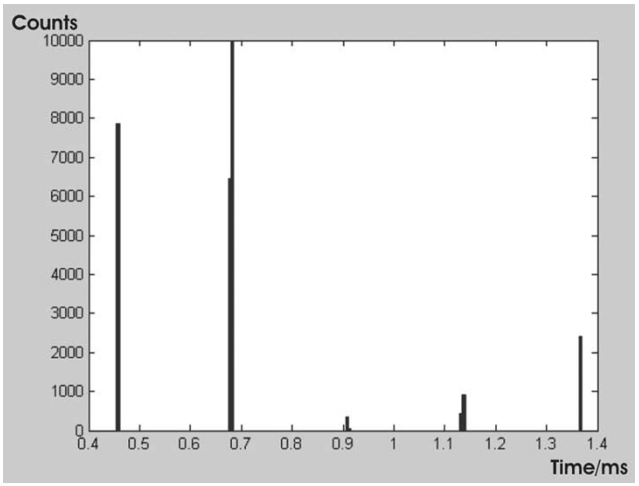

Fig. 8. Histogram of OB1 cycle time at CPU412-2 with background load.

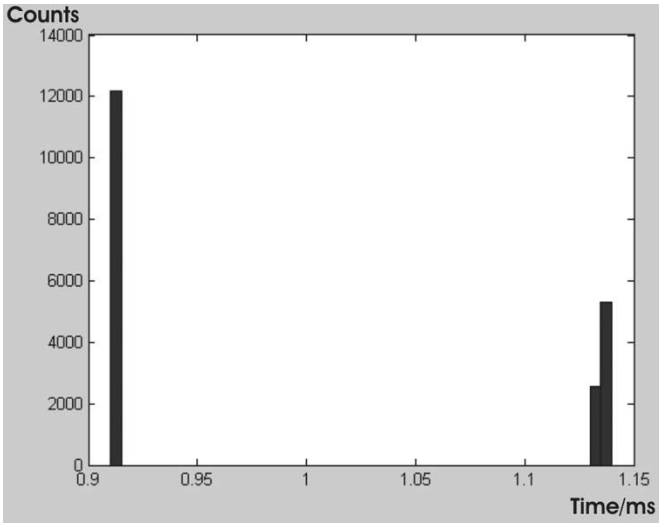

Fig. 10. Histogram of OB35 cycle time at CPU412-2.

of the output was several milliseconds with a extremely high jitter. When we increased the minimum duration of OB1 at the CPU414-1 to 1 ms, the time between output changes became much lower, and cycle time on the CPU was consistent with the speed of the outputs. This illustrates, that I/O modules have to be selected carefully. Standard modules have delays in the order of milliseconds, because of protection circuits, additional electronics to reduce electromagnetic noise, filters for stable read of switches, etc.

As expected, Fig. 8 shows that the jitter of T_C for OB1 measured at CPU412-2 increases with constant background load induced by OB35 (called every ms). Again the distribution of T_C is almost discrete, which indicates that there must be some internal cycle of about 0.2 ms. Because of the synchronous operation of a PLC, the maximum of T_C can be estimated when the duration of each OB is known. The results show that OB1 is not adequate for applications, which require a fixed scan rate, e.g., in a control loop. Section III-D examines if a cyclic interrupt OB can meet these requirements.

D. Measurements of the Cyclic Interrupt OB35

Figs. 9 and 10 illustrate that the cyclic interrupt OB35 has an extremely low jitter, compared to OB1. Again CPU412-2 exhibits an almost discrete distribution of T_C. Cyclic activation of tasks with a frequency of 1 kHz at a precision better than 0.1 ms is a unique feature of PLCs, that is even not possible with LynxOS on a Pentium II platform [7]. The smallest possible OB35 cycle time of IM151/CPU is 2 ms, because of the low performance of this CPU.

E. Hardware Response-Time Measurements

Figs. 11 and 12 show T_R, the activation time of OB40 with a rising edge of the input, including all hardware-related time fractions, measured at CPU314C-2DP and IM151/CPU. Because we had no "High feature" input for the S7-400®, we could not measure this metric for the S7-400® series. The mean value of T_R and its variance measured at IM151/CPU are unsatisfactory, even for a low-end PLC. The values of T_R

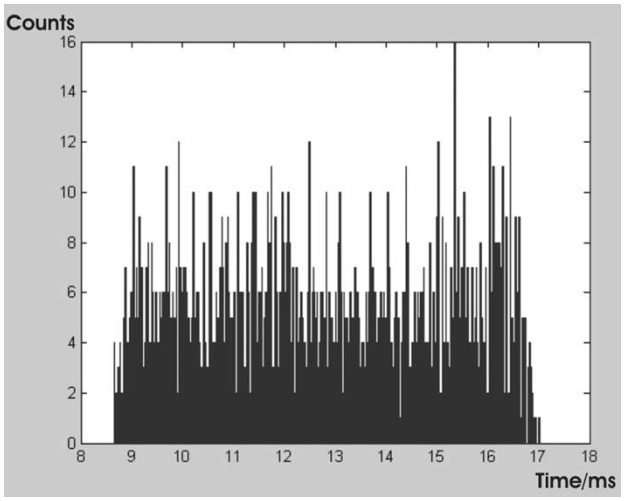

Fig. 11. Histogram of OB40 response time at IM151/CPU.

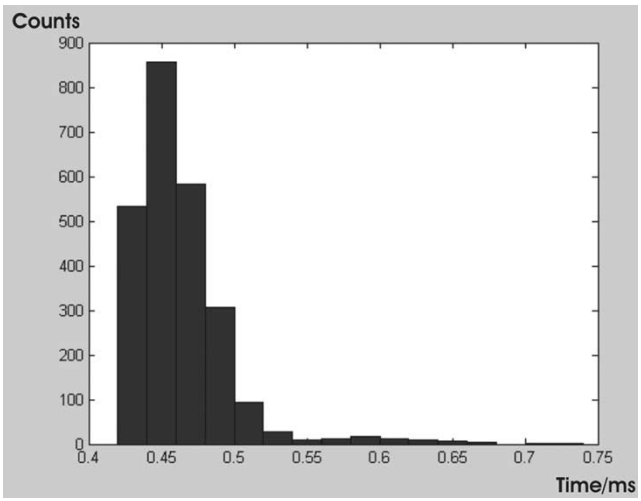

Fig. 12. Histogram of OB40 response time at CPU314-2DP.

measured at CPU314C-2DP are about five times worse than a POSIX system on Pentium II platform [7], but are sufficient for typical PLC application scenarios.

IV. Conclusion

It was shown, that usage of PLCs in physics experiments offers numerous advantages. They operate in a synchronous cyclic way with high predictability. With regard to their real-time properties, they are adequate for application scenarios, that require a deterministic response time in the order of a few milliseconds. Response times of 1 ms or even less require a very careful selection of hardware components. When deterministic response times of less than 0.5 ms are required, PLCs don't seem to be appropriate. Because of their low jitter cyclic interrupt OBs are most appropriate for applications requiring a fixed scan rate, whereas response time can be minimized by using hardware interrupt OBs. The main cycle OB1 is not appropriate for RT operation.

Future work will concentrate on the SoftPLC WinAC®, which is available on the PC platform under WindowsNT, with and without the Venturecom RTX real-time extension, as well as on a MIPS platform under WindowsCE. An additional focus of interest is the new isochronicity mechanism of the S7-400®, based on cycle synchronization. This feature shall allow reading and writing of I/Os in decentral periphery systems synchronously with the CPU cycle.

References

[1] R. J. Lauckner and R. Rausch, "Integration of industrial equipment and distributed control systems into the control infrastructure at CERN," in *Proc. ICALEPCS'97*, Beijing, China, Nov. 1997.

[2] P. Chevtsov, "PLC support software at jefferson lab," in *Proc. PCaPACs2002*, Frascati, Italy, Oct. 2002.

[3] H. Kleines, K. Zwoll, M. Drochner, and J. Sarkadi, "Integration of industrial automation equipment in experiment control systems via PROFIBUS—Developments and experiences at forschungszentrum jülich," *IEEE Trans. Nucl. Sci.*, pt. I, vol. 47, pp. 229–233, Apr. 2000.

[4] K.-H. John and M. Tiegelkamp, *SPS Programmmierung Mit IEC61131-3*. Berlin: Springer-Verlag, 2000.

[5] B. O. Gallmeister, *Programming for the Real World, POSIX.4*: O'Reilly & Associates, 1995.

[6] H. Kopetz, *Real-Time Systems. Design Principles for Distributed Embedded Applications*. Boston, MA: Kluwer Academic, 1997.

[7] K. M. Obenland, "POSIX in real-time," *Embedded Systems Programming*, vol. 4, Apr. 2001.