A TIMING SYSTEM FOR CYCLE BASED ACCELERATORS

J. Gutleber, CERN, Geneva, Switzerland
Z. Croflic, J. Dedic, R. Stefanic, Cosylab, Ljubljana, Slovenia

Abstract

Synchrotron accelerators with multiple ion sources and beam lines require a high degree of flexibility to define beam cycle timing sequences. We have therefore designed a ready-to-use accelerator timing system appliance based on off-the-shelf hardware and software that can fit mid-size accelerators and that is easy to adapt to specific user needs. This Real Time Event Distribution Network (REDNet) has been developed under the guidance of CERN within the MedAustron-CERN collaboration. The system is based on the MRF transport layer and has been implemented by Cosylab. While we have used the NI PXIe platform, it is straightforward to obtain receivers for other platforms such as VME. The following characteristics are key to its readiness for use: (1) turn-key system comprising hardware, transport layer, application software and open integration interfaces, (2) performance suitable for a range of accelerators, (3) multiple virtual timing systems in one physical box, (4) documentation developed according to V-model. Given the maturity of the development, we have decided to make REDNet available via the integration partner.

MOTIVATION

Traditionally, timing systems for particle accelerators tend to be solutions that are tailored to a particular project, whether based on custom or on off-the-shelf components. The Proton Ion Medical Machine Study (PIMMS) at CERN [1] defined the characteristics for an entire class of mid-scale synchrotron-based particle accelerators for light-ion cancer therapy. That class of machine requires a high degree of configurability, induced by the need for large amounts of different beam characteristics needed for medical irradiation sessions. Although the motivation to design and implement a main timing system has its origin in the MedAustron project at CERN [2,3], *the goal was to provide a generally usable main timing system appliance for a mid-size class of particle accelerators for industrial and medical applications*. PIMMS-type accelerators are now emerging in several places. They share a set of commonalities, which define the operation concept and required timing functions, which we outline in the following two sections.

OPERATION CONCEPTS

The *task of the main timing system is to distribute events in real-time to front-end controllers of beam-line elements, where they trigger actions, which are associated to particular events. The system enables front-end controllers to carry out the actions in a synchronized fashion*. Hence, we call the system REDNet, Real-Time Event Distribution Network. The point in time at which actions take place determine the beam-generation process. We call a pre-defined sequence of events for a particular process that generates a beam with a certain set of characteristics a *cycle*. A medical irradiation session or an experimental physics beam application consists of a sequence of pre-defined cycles, called a *run,* see Fig. 1.

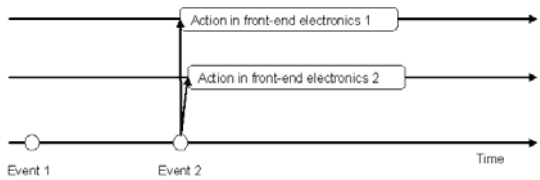

Figure 1: Events trigger actions in frontend controllers in real-time. The actions are responses to events that are used for beam generation purposes.

DESIGN

Overview

The overall design goal was to come to an *accelerator timing system appliance* that can be part of a general Medical Accelerator Control System [4]. It includes all elements spanning from transport to processing hardware, including software drivers, libraries and a generally usable timing system sequencer application that can be integrated with industrial SCADA systems and tools, see Fig. 2.

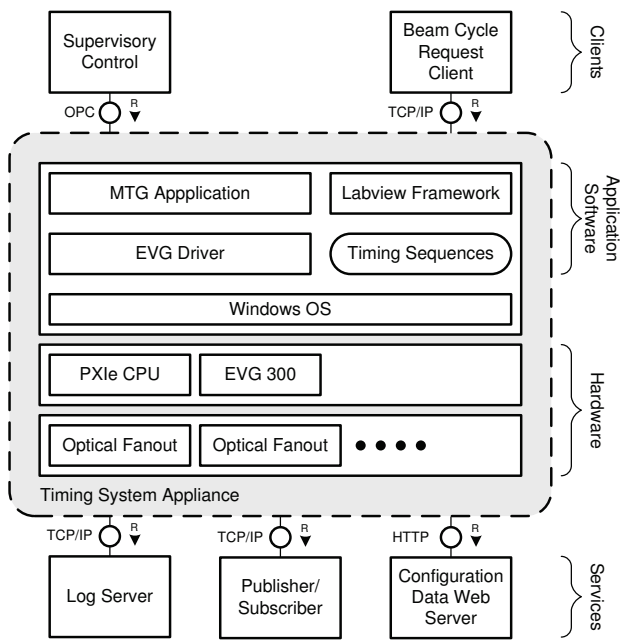

Figure 2: Schematic overview of appliance.

Although a specific hardware, operating system and programming language environment had to be selected, care was taken that the appliance can be integrated

ISBN 978-3-95450-139-7

seamlessly into different environments. Consequently an emphasis was put on using accepted protocols and technologies for supervisory control and configuration. Details on the technologies used for implementation are highlighted in section Deployment.

Functions

The foundation of the system is the timing specification for a sequence of events that can be used for a set of beam cycles. A run is composed from an arbitrary amount of timing sequences, limited only by the memory capacity controller CPU. Timing sequences are specified in an open XML format (see example below). A timing sequence contains a list of events with time and broadcast indications. The appliance caches timing sequences from the configuration Web server. Temporarily, sequences can be overridden using the SCADA interface for machine development and fine-tuning the timing specifications. When a client requests a particular beam cycle, the appliance resolves the mapping timing sequence and starts broadcasting the events in real-time according to their timing indications.

```
<TimingSequence id="ExampleCycle"
  version="1.0.0" UUID="..." status="in work"
  author="..."
  creation="2011-02-09T09:22:30.835593Z" ..>
   <EventTypeIdentifier id="StartCycle" time="0"
   spec_time="0" isDynamic="false" rt="true"
   nonrt="true" acknowledge="false" index="0"
   absolute_time="0" is_primary="true" />
   ...
   <EventTypeIdentifier id="StartAcceleration"
   time="305000" spec_time="310000"
   related_event="10" isDynamic="false"
   rt="true" nonrt="true" acknowledge="false"
   index="12" absolute_time="577242"
   is_primary="true" />
...
   <EventTypeIdentifier id="EndCycle"
   time="479000" spec_time="479000"
   related_event="26" isDynamic="false"
   rt="true" nonrt="true" acknowledge="false"
   index="27" absolute_time="3000744"
   is_primary="true" />
</TimingSequence>
```

Code 1: Example XML timing sequence.

Events are freely specified as human readable names. Up to 250 different user-defined events can be specified without code changes such that the system can be easily configured to use in different accelerators. The same event can be specified multiple times in a single timing sequence. The system is designed for timing sequences with durations up to 1024 minutes and up to 512 events per timing sequence. However, these limits may be changed at deployment time. Two hard assigned events are always present in every timing sequence, one that indicates the start and one that indicates the end of the cycle. Timing events in a sequence must at least be spaced by 1 usec. A single timing sequence can be assigned to multiple different accelerator cycles to reduce the overall amount of required timing sequences. Assignment of timing sequences to cycles is based on a selection of beam characteristics such as particle type, energy, extraction duration, beam size, target beamline and other possible cycle parameters. Each event is tagged with a timestamp at generation time that can be used to correlate data at event receivers. A particular heartbeat event is broadcast at a configurable, regular interval, for instance at 10 Hz. The start of timing sequence generation can be configured with a negative offset with respect to the heartbeat, such that the regularly broadcast heartbeat events occur always at the same time within a timing sequence. This feature serves usually the synchronization between a regularly pulsed injector and the synchrotron. Additional local reference clocks are available as firmware plugin-components for the event receiver cards. These highly precise reference clocks can be phase corrected with respect to the start of a cycle. Examples are provisioning of a 10 MHz clock for backplane synchronization, 200 kHz as clock for a low-level RF system or 100 MHz for a set of synchronously working digital signal processors.

The user can enqueue any pre-defined event asynchronously into an active run by indicating a relative time delay with respect to another event. The timing system will then include the user event in the next cycle and distribute it as part of the pre-defined timing sequence. This feature lets automated procedures easily insert events to trigger measurements at precise points in time dynamically.

Timing sequences are pre-defined in released configurations that are kept on a Web server. Upon startup of the main timing system generator, the system determines if the latest released set of software and timing sequences is locally cached and updates its configuration if needed. A user can download additional timing sequences at any time into the timing sequence and define to which beam cycles the newly provided sequence matches.

The system has four modes of operation: in service mode, the software and firmware can be updated and system tests can be performed. In machine physics mode the system can be used to perform runs and play timing sequences, which are not marked for clinical or quality assurance use. In this mode, the user can freely download new timing sequences, change and add associations of timing sequences to beam cycles. In quality assurance mode any modification of the pre-loaded, released configuration of timing sequences and mapping of sequences to cycles is unchangeable. In clinical mode, the same constraints as for quality assurance mode apply and only requests to play cycles that are marked for clinical mode are accepted.

While a timing sequence is played out, the system already accepts a request for a next sequence, such that fully pipelined operation without dead-time between cycles can be achieved. The client can choose if the next timing sequence starts as soon as the current sequence ends or if the system should wait for an additional start request. Once a sequence of events is generated, it can be aborted or a request to fast-forward to a specific timing event in the sequence is possible. The latter function

ISBN 978-3-95450-139-7

permits operators to stop a running timing sequence, by ensuring that the accelerator completes the cycle in a well-defined manner.

One physical system implements five virtual timing systems, called timing execution slots that can be operated concurrently [5]. A client requests a timing system slot with the request to start a run. Consequently, the frontend controllers that will eventually receive events are configured to work with a particular timing slot before the run is started. This functionality permits to perform multiple tasks such as beam commissioning, hardware commissioning, software development or testing of medical frontend systems concurrently without the need to purchase and re-wire multiple timing systems.

Real-time distribution of events from generator to receivers is unidirectional broadcast via an optical fanout. Message integrity is verified by CRC16 checking at the receiver and the client-side software moves to failed state if an error is detected. Frontend controllers can, however, acknowledge event reception to the generator via TCP/IP messages over the control system's Ethernet infrastructure. If acknowledgements are not received within a deadline, the virtual timing system moves to failed state. The design foresees the extension to postpone cycle execution until the event is acknowledgement by a configured number of front-end controller applications. In addition, the XML format permits defining the emission of events with GPS timestamp at 100 nsec precision also via a non-real time publisher/subscriber mechanism over Ethernet. Such, user interface applications can easily display and correlate data that have been acquired as result of event reception in front-end controllers.

The timing system scales to several thousands of event receivers since the broadcast network can be built from multiple fanout stages. A single level fanout configuration accommodates 100 possible event receivers. For MedAustron, only 30 frontend controllers with real-time event reception capability were required.

Performance

The real-time performance of the timing system has been matched to the PIMMS class of particle accelerators, but is suitable for a variety of different accelerators. At an injection energy of 7 MeV the revolution time in the synchrotron is 2 µsec. At highest extraction energy at 800 MeV the revolution time is 0.1 µsec. Timing events are distributed to destinations in real-time such that two devices receive the same event within a time window of less than 1 µsec. In addition, devices that do not have processing capabilities, such as selected power converters or oscilloscopes, can receive signals corresponding to timing events as electrical or optical pulses directly via front-panel connections. This functionality allows high-precision synchronization of two event receivers down to 10 nanoseconds as shown in the figure below. The total latency from event emission to event response generation at the receiver side is 200 nsec + 5 nsec/m fiber, effectively indicating to the user that the absolute timestamp of an event is close to the time at which an action occurs in the front-end controller. The constant latency between event distribution and receiver-side response generation is, however, not an issue since all events, including the start of a cycle is shifted by this constant time. On the receiver, event identifier, cycle identifier, timing slot, run number and a 100 nsec GPS timestamp of event emission time are made available to applications via a software API, see Fig. 3.

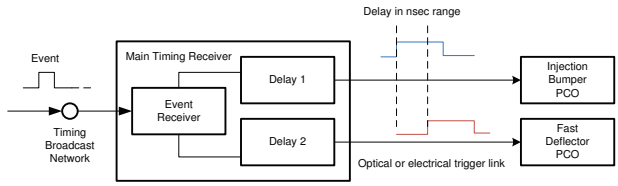

Figure 3: High-precision synchronization of beamline elements.

For simple development setups it is possible to emulate the presence of a complete timing system with a software application in a front-end controller that hosts a timing receiver card. It permits to run a locally loaded timing sequence and exercise the behaviour of applications and hardware devices that require event triggering without the need for the entire control system or the physical presence of the main timing system generator system. Generation of electrical and optical pulses on the front-panel outputs and on the front-end controller bus backplane can also be directly requested via calls to the software API. This function permits testing the most basic application and hardware functions during development time without the need for timing sequences and defined events.

DEPLOYMENT

The deployed timing system is composed from a Symmetrikom SyncServer S350 reference clock, an OCXO 10 MHz to 100 MHz frequency multiplier, a National Instruments 3U 8 slot PXIe crate with PXIe CPU, an MRF [6] PXIe EVG card, a 6U cPCI crate and 6 MRF optical fanout cards. The timing system broadcast network uses standard Gigabit Ethernet OM3 fiber-pairs and OM1 single fibers are used for optical triggers from timing receiver cards to power converters. The system is installed in one 19" rack.

The application software is written in Labview on top of the 64 bit MS Windows 7 operating system. The MRF EVG 300 card features a Virtex 5 FPGA, which is programmed in VHDL. The timing system generator application is based on a common Labview framework that can also be used to program frontend controllers on PXI and PXIe platforms using Labview and Labview/RT. Currently, non PXI/PXIe FECs are integrated via timing receiver card front-panel TTL pulses and a low-latency datagram over the PXIe CPU's serial output. However, making receiver libraries available on a different operating system is a mere implementation effort, since the design is well-documented.

Configuration, state and mode transition changes and monitoring are performed via the OPC protocol. Timing

ISBN 978-3-95450-139-7

events can also be published along with run number, cycle number, timing slot number and GPS timestamp via a proprietary, documented TCP/IP application layer protocol to a publisher/subscriber system. Publishing and subscription API libraries exist for C, C++, Labview and C#. Application and system log messages are distributed to a log4net based log server via the originally log4j developed UDP based log protocol. Log message pre-filtering and selection of log messages to file or network are configurable in the appliance to keep log message traffic under control during operation. Timing sequences, framework, driver and application software releases are picked autonomously from a Web server via the HTTP protocol. Graphical user interfaces for testing exist as Labview panels and ready-to-use operation panels exist for the SIEMENS/ETM WinCC OA SCADA tool. If the latter tool is used, data logging, alarming and on-the-fly modification of timing sequences for machine development are available in WinCC OA as scripts and datapoint element configurations, see Fig. 4.

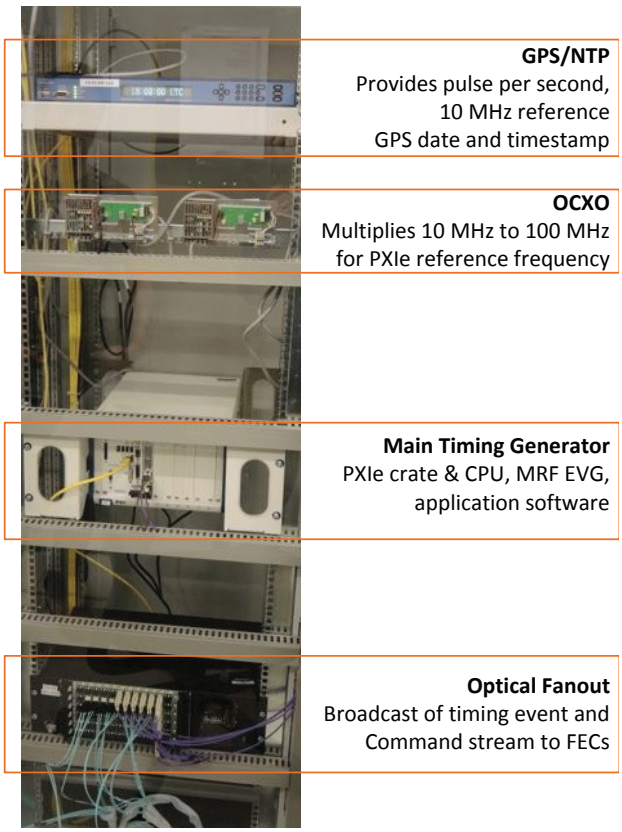

Figure 4: Appliance as delivered and deployed.

FUTURE WORK

The current implementation includes the possibility require acknowledgements for individual timing events. If the indicated number of acknowledgements is not received by the timing system generator within a specified timeout, generation will be stopped by the system. The implementation's focus was on the functionality rather than on performance. The acknowledgement mechanism is therefore currently available over TCP/IP only. Particular application may, however, require low-latency acknowledgements within microseconds and the possibility to pause timing sequence generation rather than aborting it. The system design has been foreseen to accommodate those additional features that can be implemented if the arises. In addition, the design foresees to pause cycle execution rather than moving to failed until acknowledgements have been collected. Supervisory control protocol is currently implemented via the National Instruments OPC/Shared Variable engine. It is suggested to upgrade the software to Labview 2012 and switching to OPC/UA protocol to improve performance and stability.

CONCLUSIONS

Based on the PIMMS design, an accelerator main timing system for a mid-size class of synchrotron accelerators requiring large amounts of different beam cycles, no dead-times between cycles and a large amount of flexibility without a need to intervene at hardware or firmware level has been devised. Test operation has shown good performance and robustness. We have therefore decided to make the CERN IP and design available through an integration partner, Cosylab (Ljubljana, Slovenia). The system is based on the MRF timing system transport layer. Current implementations by Cosylab are available for PXI and PXIe systems with Labview software support for front-end controller applications. The design is, however, flexible and timing receiver cards for other platforms such as VME exist such that accommodating different hardware and operating system platforms are a mere implementation task.

REFERENCES

[1] L. Bodano, M. Benedikt, P. Bryant et al., PIMMS study, CERN-PS-00-010-DI.
[2] M. Benedikt, U. Dorda, J. Gutleber et al., "Overview of the MedAustron design and technology choices", Conf. Proc. C100523(2010) IPAC-2010-MOPEA20.
[3] M. Benedikt, A. Fabich, "MedAustron – Austrian hadron therapy centre", Nuclear Science Symposium Conference Record 2008, IEEE, p.5597-5599, 19-25 Oct. (2008).
[4] J. Gutleber, R. Moser, "The MedAustron Accelerator Control System: Design, Installation and Commissioning", in proceedings of ICALEPCS 2013 conference.
[5] R. Stefanic et al., "Timing System Solution for MedAustron; Real-Time Event and Data Distribution Network", Proc. ICALEPCS 2011, WEPMN015.
[6] J. Pietarinen, "MRF Timing System", Timing Workshop CERN, February (2008).

ISBN 978-3-95450-139-7