THE MRF TIMING SYSTEM. THE COMPLETE CONTROL SOFTWARE INTEGRATION IN TANGO

J. Moldes, D. Beltrán, D. Fernández, J. Jamroz, J. Klora, O. Matilla, R. Suñé
CELLS, Cerdanyola del Vallès, Barcelona, Spain

Abstract

Alba [1] is a synchrotron light source under installation located nearby Barcelona. This 3 GeV third generation light source is planned to deliver the first X-rays beam to the users in 2012. The linac was commissioned in 2008, the booster in 2010/2011 and the Storage Ring in 2011. The seven beamlines included in the "Phase One" are being commissioned at the end of 2011.

The timing system is mainly used to synchronously distribute signals to all the equipments of the machine that need them. It is built on top of the MRF [2] timing solution. It is a fast digital, point to point, event based system. The system basically works by sending event codes from one event generation source (synchronized with RF/4 frequency) through a fast, tree structured, bidirectional fibre optics cabling network to many event receivers (about 100), which react to these event codes in different ways, amongst which the most commonly used is to generate a pulse (which characteristics are configurable) in one of its outputs. This pulse is connected to the hardware equipment that physically needs it. There are currently 418 output signals connected to different equipments. In other words, timing provides the "synch" in the synchrotron, keeping synchronized all the key systems (power supplies, RF and diagnostics).

The receivers are also capable of sending event codes up to the event generator when they detect activity in one of their two inputs. The generator can then redistribute this event again in downwards direction to all the receivers. This feature has been used to implement the Fast Interlock system [3]. There are currently 49 interlock signals connected to the system.

Logging of events with globally distributed timestamps is available in the receivers. This feature is specially useful (coordinated with Fast Interlock system) to easily and quickly determine the cause of a sudden beam loss.

The event generator is also capable of distributing up to 8 arbitrary digital signals using another feature called Distributed Bus (DBus).

In combination with other systems, timing will also be the key for getting the desired filling pattern of the machine and for implementing the top-up feature.

LAYOUT

The layout of the system is detailed in Fig. 1. As pointed above, it is basically a tree structured fibre optics network, starting from the event generator (EVG from now on) fanned out by the fanout/concentrators (fanouts from now on) and ending in the event receivers (EVR[s] from now on). This downwards path is the one mainly used. The EVG sends event codes to the EVRs, which react to them mainly by generating a pulse in any of its outputs.

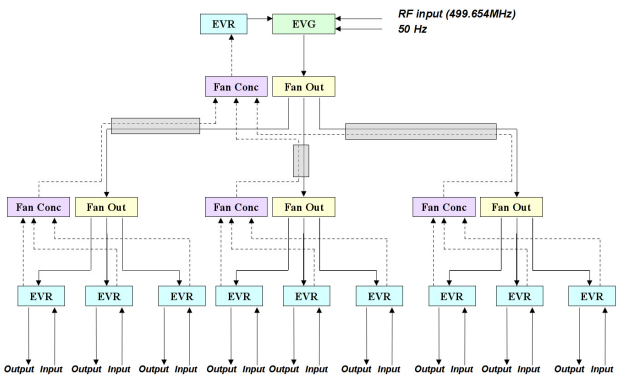

Figure 1: Layout of the Timing System.

However, the communication can also be done in upwards direction, starting from any of the EVRs sending an event up to the EVG. The later can then propagate downwards the event to all the EVRs. This is the core of the Fast Interlock system [3].

THE HARDWARE

Hardware was provided by different companies.

The core of the system, formed by the EVG, EVRs and fanouts, was supplied by MRF [2]. The EVG is the 230 model and almost all the EVRs are also 230 model. In addition to these, a few units of EVRTG300 have been acquired for special purposes, which we will see later.

It was decided to host the EVG and EVRs in cPCI machines, due to their robustness and reliability. Hence, the EVG and EVRs (model 230) were supplied in cPCI 3U form. The 3U cPCI chassis cPCIS-2632 and CPUs cPCI-3840 were supplied by Adlink [4].

The fanouts come in 6U cPCI form. Adlink chassis cPCIS-6418U and cPCIS-6130R were selected to host these cards. Fanouts control is much simpler than EVG and EVRs and hence it is not necessary to use a CPU to manage them. They provide an ethernet connection and an API based on UDP that is enough to control them.

The EVRTG300 units are also 6U cPCI form and were mounted in cPCI-6130R chassis with a cPCI-6965 CPU.

The fibre optics cables were supplied by R&M [5].

In the next section we will have a more detailed view of the main hardware components (EVG, EVRs and fanouts) and their main capabilities. Thought extensive, it is not a

complete description of these devices. See [2] for full information.

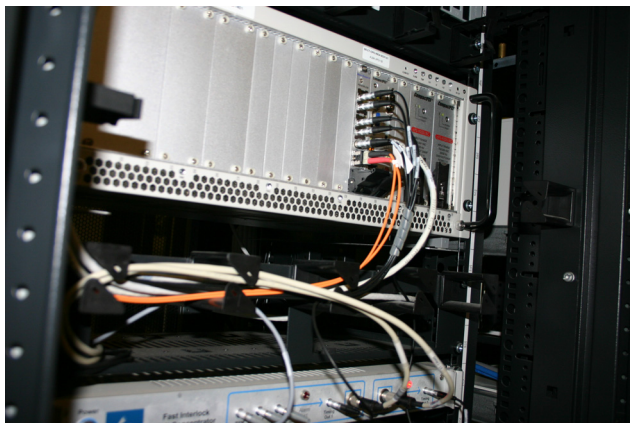

Figure 2: cPCI chassis with CPU and 1 EVR.

Figure 2 shows an example of use of the system. We can see a cPCI chassis with a CPU and an EVR. The four outputs of this EVR are used in this case to provide timing to 8 BPM units. We can also see two inputs coming from a Fast Interlock concentrator [3], in this case coming from any of the 8 BPM units in case the beam orbit gets out of limits. The two orange Rx/Tx fibre optics cables are also visible in the figure.

Event Generator

The EVG is responsible for creating and sending out the events to the EVRs.

Events are sent out by the EVG as event frames (words), which consist of an eight bit event code and an eight bit distributed bus data byte. The maximum event transfer rate is derived from the external RF/4 clock. The optical event stream transmitted by the EVG is phase locked to the clock reference.

There are several ways of generating the events:

- Trigger events. A single event is fired by an internal configurable counter.
- Sequence events. This is the feature used in our case. The sequence is a table like structure, with pairs timestamp - event, so the users can define exactly the time to wait until the next event is sent. The sequence itself is triggered by an internal configurable counter. Receivers can be grouped by subsystem to react only to some events. This way, users can fire different events that trigger only a given subsystem, without affecting the rest.
- software events: any event can manually be sent on user request.
- events received from the upstream EVRs (used for the Fast Interlock System).

In addition to events the EVG enables the distribution of eight simultaneous signals sampled with the event clock rate. This feature is called distributed bus. Distributed bus signals may be provided externally or generated on-board by programmable counters.

Event Receivers

EVRs decode timing events and signals from the optical event stream transmitted by the EVG.

The EVRs lock to the phase event clock of the EVG and are thus phase locked to the RF reference.

The EVRs basically convert the data got from the EVG to hardware outputs. Every info packet got from the EVG is 16 bits long, and is divided into two parts:

- Event code (8 bits).
- DBus bits (8 bits). Any output of the receiver can be configured to follow any of these bits. When the value of this bit is 1, the output is set to high, and it is set to low when the value is 0. This means that we can reproduce up to 8 arbitrary digital signals in any part of the machine at clock frequency resolution.

Two mapping RAMs (only one can be active at a time) are provided in order to configure how the EVR will react when receiving a given event. This RAM contains a table like structure in which event codes are associated to different actions, amongst which the most common ones are:

- Trigger a pulse in a given output. The delay since the event is got is configurable, as well as the width. Both width/delay can be set in 8 nano seconds steps.
- Set a given output to high. A delay is configurable.
- Reset a given output to low. A delay is configurable.
- Log the event. Events can be stored with globally distributed timestamps into a log memory.
- Forward the event downstream.

Two external hardware inputs are provided to be able to take an external pulse to generate an internal event. This event can also be transmitted in upwards direction to the the master EVR and from this to the EVG for distributing it downwards to all the EVRs. This is the core of the Fast Interlock system [3]: any interlock signal connected to any receiver can be distributed to the whole machine in 4.2 micro seconds.

EVRTG300 units add to the above a special feature. A fine grain delay of 10 pico seconds steps can be set in any of its outputs. This will be used for being able to inject any bucket of electrons in any position of the storage ring. It will also be used for the kicker magnets delay fine tuning and for providing timing to the streak camera.

Fanouts

These devices are mainly hardware devices and have few configurable settings. They have a dual function:

- The 8-way fan-out receives the optical event signal through a fibre connected to the uplink RX port. This signal is fanned out to all eight downstream ports.
- The concentrator receives signals from up to eight EVRs or downstream concentrators and forwards the signals upstream.

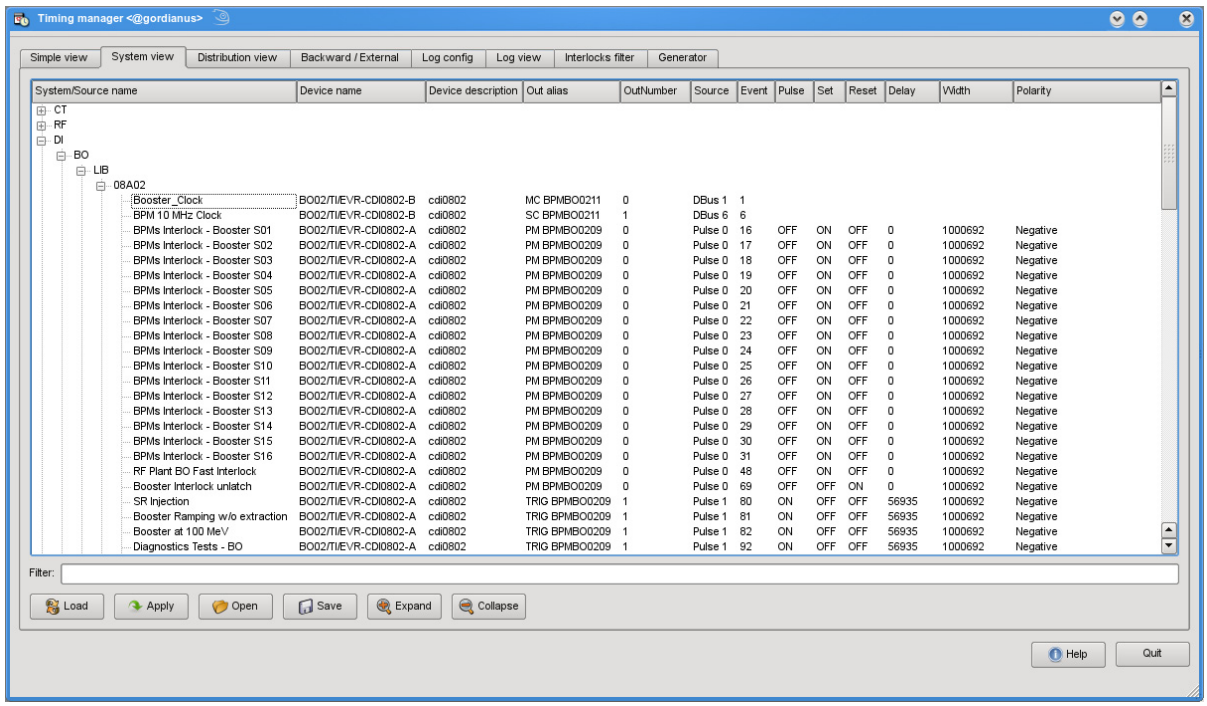

Figure 3: Timing manager GUI, the main users interface with timing system.

THE SOFTWARE

The software for controlling the system was developed from the lowest level up to the highest user GUIs level. In the middle we had to develop the core of the system: the tango device servers in order to integrate all the components into tango [6].

Drivers and API

Linux device drivers for the EVG and the EVRs had to be developed. Drivers for kernel 2.6 were developed by R. Suñé and J. Pietarinen from MRF [2].

In addition to this, a binding for python has been developed in order to use the manufacturer's API, which is written in C++. Swig [7] has been used for this purpose.

Tango Device Servers for EVG, EVR and Fanouts

Tango device servers have been developed to control all the features of the EVG, EVRs (both 230 and 300 models) and fanouts. These device servers have been written in python, using the above mentioned *pythonized* API and the python binding for Tango: PyTango [6]. They allow to control all the features of a single unit. In addition to this, the tango device server allows to permanently store in Tango database the configuration of a given unit and reload it when the device server is started. This permits the permanent storage of the configuration of the whole timing system.

Tango Device Servers for Configuring and Monitoring the System

In order to be able to control all the units in a centralized and convenient way, the TimingManager device server was developed. This device allows to set/retrieve the configuration of all the EVRs using a simple XML format and also configure/retrieve the logs.

A TimingMonitor device server was developed for monitoring the system health. This device is continuously monitoring the status of the EVG, EVRs and fanouts to find any problem that may occur.

THE GRAPHICAL USER INTERFACES

EVG and EVR Expert GUI

Expert GUIs for the EVG and EVRs (both models included) were developed. These GUIs communicate respectively with the EVG and EVR tango device servers in order to set/retrieve all the settings of the unit. These GUIs are meant to be used by expert users only.

Timing Manager GUI

This GUI is the users main interaction point with the system. Fig. 3 shows this application. This GUI runs of top the TimingManager device server.

Users provided the initial configuration of the system in a spreadsheet, basically specifying for each output of each receiver which event or events would trigger that output and also the configuration of the pulse/set/reset to be triggered. For each receiver they also provided a tag, which was used for splitting it and building the tree structure shown in Fig. 3, hence allowing users to group EVRs by subsystem and easily locate any output.

The first three tabs, show basically the same information but sorted in different ways. As shown in the figure, every line shows the output information, formed by

the tree node (including event name), the EVR tango device name, a user editable description for that EVR, a user editable alias of the output, the output number, the source for the output (DBus number or event code), pulse/set/reset (exclusive), pulse delay and width (the later no applicable for set/reset) and polarity. Delay, width and polarity are user editable.

The difference between "Simple view" and "System view" is that the former shows only one line per output (different events can trigger the same output), while the later displays the same line as many times as events trigger it. "Distribution view" show the same information as the "System view", but in this case the tree is built from the event code down to the EVRs, hence allowing users to know which outputs of which EVRs are triggered by a given event.

"Backward/External" allows the configuration of the backward events, used by the Fast Interlock system [3].

"Log config/Log view" allows the configuration of the logging of events in any EVR and retrieving/inspecting the contents of the logs of all the EVRs.

"Interlocks filter" allows to enable/disable forwarding of a given event in the main top EVR of the system (the one on the top in Fig. 1). This is useful for preventing a given interlock to be propagated downstream.

"Generator" tab allows users to enable/disable the EVG sequencer and editing it's contents.

The open/save buttons allow users to load/save the configuration of the whole timing system.

Timing Monitor GUI

This GUI is built on top of the TimingMonitor device server, which is continuously monitoring the system and warns about any problem it detects. A snapshot of the GUI can be seen in Fig. 4.

Figure 4: Timing monitor GUI.

The aspects that are monitored:

- EVR violation. All the EVRs can detect a problem in the fibre optics link it receives.
- Direct heartbeat. A heartbeat event is periodically sent by the EVG. A time out alarm is set by any EVR if it doesn't receive that event for a predefined time.
- Reverse heartbeat. Every EVR is instructed to send a heartbeat periodically up to the main EVR.
- EVR Alive. It checks the tango state of every EVR device server.
- Fanouts. It checks the status of the fanouts.

CONCLUSION

The development of the control of ALBA timing system has been quite a big effort. It has proved to be a very useful tool for commissioning, during which the fine tune of the timing of any of the key elements of the machine was possible with simply a two click operation. The save/open feature allowed the saving/restoring of any configuration of the whole timing system with a 3 click process. The permanent storage of all the settings in tango database allows a transparent recover after a system shutdown (i.e. for maintenance): all the settings are automatically restored on power up.

The timestamped logging feature also proved to be a valuable tool for commissioning, allowing to easily and quickly find the root of the problem when beam was suddenly lost.

The Fast Interlock system [3] has also proved to be the fastest system for interlocking the whole machine.

The control system has provided the users a very simple tool which abstract them from the big complexity of the underlying system.

As part of the tango project [6], all the code and its dependencies are free software and can be freely and easily reused by the community with relatively little effort.

CONTRIBUTIONS

Many people has contributed to this project. Specially remarkable is the outstanding effort in developing a big part of the software by R. Suñé before leaving Alba. Thanks to J. Pietarinen [2] for his excellent support for the devices he provided. Many thanks to O. Matilla and D. Beltrán, who were the main designers of the system itself. Thanks also to J. Jamroz for hardware support.

REFERENCES

[1] CELLS-ALBA: http://www.cells.es
[2] Micro-Research Finland: http://www.mrf.fi
[3] O. Matilla et Al, "The Alba Timing System. A know architecture with a Fast Interlock System Upgrade".
[4] Adlink: http://www.adlinktech.com
[5] Reichle & De-Massari: http://www.rdm.com
[6] Tango: http://www.tango-controls.org
[7] Swig: http://www.swig.org