ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUCPL04

SwissFEL TIMING SYSTEM: FIRST OPERATIONAL EXPERIENCE

B. Kalantari, R. Biffiger, Paul Scherrer Institute, Villigen, Switzerland

Abstract

SwissFEL timing builds on Micro Research Finland (MRF) event system products. Requirements for performance and new features have pushed MRF timing components to its newest generation (300 series) providing active delay compensation, conditional sequence events, topology identification and other interesting added features. However employing available hardware functionalities to implement complex and varying operational demands and provide them in the control system is its own challenge. In this paper after brief introduction to the new MRF hardware, we concentrate on operational aspects of the SwissFEL timing and related control system applications. We describe a new technique for beam rate control and explain how we provide assistance to machine protection system (MPS) using the same technique. We show how a well thought software design supports a clear interface between software layers and allows the use of the low level abstraction in different applications with reduced development effort. We also discuss our pulse marking (timestamping) method and its interface to beam synchronous data acquisition system. Further we share our experience in timing network installation, monitoring and maintenance issues during commissioning phase of the facility.

INTRODUCTION

SwissFEL [1] is a 740 m long FEL facility with final energy of 5.4 GeV, currently under commissioning at Paul Scherrer Institute in Switzerland. The first pilot experiment is scheduled before end of 2017. The nominal repetition rate of the SwissFEL machine is 100 Hz. A large number of various devices and systems distributed across the facility require precise timing to achieve synchronous controls and data acquisition. The SwissFEL timing system uses with MRF [2] timing modules. Two different kinds of modules exist: Event Master (EVM) modules and Event receiver (EVR) modules. EVM modules can be used as an event generator and as a fan-out. These timing modules are connected in a tree structure where one EVM is used as root node (top level event generator) and other EVMs are used as branch nodes (fan-outs). EVRs are used on the leaves of the tree. The nodes are connected via fibre optic links. The root node EVM generates global events and synchronous data and sends them down the tree. EVRs decode received event stream and use them to generate hardware triggers and/or process software objects. A reference clock at 142.8 MHz synchronizes the event system with the master oscillator of the accelerator.

OPERATIONAL REQUIREMENTS

Beam Rate Control

In order to have stable beam operation at reduced rates (below 100 Hz), subsystems such as RF and lasers must be triggered at all machine pulses. Reduced beam rates are achieved by delaying the actual triggers of those subsystems typically by a few microseconds in selected machine pluses. For RF systems this added delay stops acceleration in that machine pulse. As illustrated in Fig. 1, RF event is distributed at every machine pulse independent of the beam rate. The trigger to the RF subsystems is generated after a specified local delay. The delay value *T* at machine pulse *N* causes the acceleration (Fig. 1 on the left) and generates beam. Delay value *Td* instead, is specified relative to the RF phase such that it stops acceleration which leads to the machine pulse with no beam (Fig.1 on the right).

This is the principle for beam rate control in all machine timing modes. This method also avoids acceleration of dark current and keeps the systems' operation stable by continuous triggering.

MPS Assistance

Machine protection at SwissFEL has its own implementation which is independent of the timing system. However timing system assists MPS as a second level protection. The requirement is that upon generation of certain MPS alarms beam rate should be instantly reduced to a configurable value. In other cases the beam must be completely stopped. This avoids destroying expensive devices such as undulators or screen monitors. The method to stop beam or reduce its rate is the same technique as for beam rate control that we explained in the previous subsection. The difference is that the information to delay RF trigger is not known ahead of time. This information can become available at any time during the machine pulse and it needs instant reaction on changing the delay value. Therefore it cannot be achieved by a software method (i.e. reprogramming of the delay value) and requires a hardware solution.

Flexible Event Rates

Already at the time of collecting requirements it became clear that we should maintain several event rates since not all systems will run at full rate during the commissioning or later for other machine setups like machine development, etc. A good example is RF subsystem where new RF stations are being commissioned while other stations are already in their routine operation. The new stations require independent repetition rate settings and usually are operated at lower rates for some time. There also several other examples such as slow diagnostics devices, etc.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUCPL04

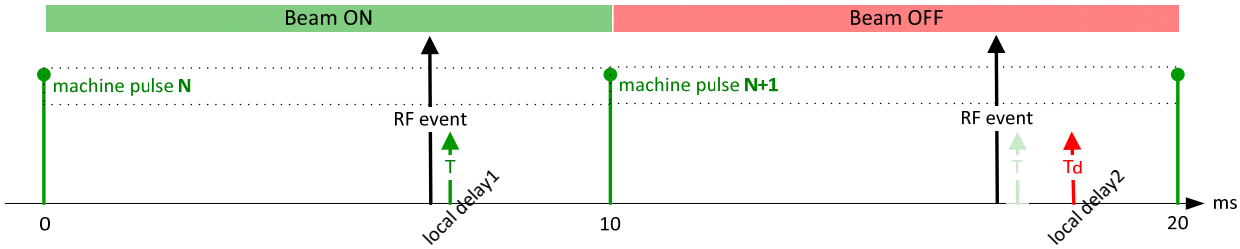

Figure 1: In machine pulse with beam (green on the left) gun RF is triggered with delay **T** after RF event. In the pulse with no beam (red on the right) gun RF is triggered with delay **Td** after RF event.

The challenge is to allow flexible rate setting of events but in synchronous manner. For instance, for generating beam in a machine pulse, among others, laser and RF must be present. Assume RF event rate at 10 Hz and laser rate at 5 Hz. This leads to a beam rate of 5 Hz given that these events meet each other at the same pulse. Otherwise there will be no beam. In practice this becomes more complicated due to existence of several related events as well as various different rates to maintain.

Beam State Flag

At every machine pulse the beam state should be reliably provided to many systems. This allows control and data acquisition systems to react accordingly. For instance, in most cases the data from pulses without beam are not interesting and shall be dropped.

Pulse Marking

Every machine pulse must be identified uniquely. This is also known as pulse marking. Since SwissFEL is a pulsed accelerator, the generated machine and experiment data can differ from one pulse to the next. Therefore correlation studies on the acquired data become an essential requirement. Such correlation will be only meaningful if the data affinity to the machine pulse is known.

HARDWARE FEATURES

Requirements of the SwissFEL timing system led to the development of the 300-series of MRF timing modules. The event clock has been increased to maximum of 142.8 MHz corresponding to 2.9 Gbps. One new feature of the 300-series is the active delay compensation. The goal of active delay compensation is to have a uniform and specified propagation delay of events in the timing network. This is achieved by measuring the round trip delay of each timing network segment and sending a data packet down the tree which contains the accumulated measured delay value of its path. Each node in the tree adds the delay of its segment and its internal delay. The EVR at the end receives this measured path delay and it delays delivery of the received event stream to achieve the specified target delay.

Another new feature is masked sequencer events. Sequencer is a table programed by software. Each row in the table consists of a delay, an event code and a mask field. When the sequencer is triggered, the event codes are sent out after elapsing of the defined delays. By using the mask fields, software or hardware input signals can control whether or not the event code should be sent without the need to reprogram the sequencer table. A combination of mask and enable bits (each 4 bits) are available. Mask bits inhibit sending of events while enable bits allow sending of events. If mask and enable fields are left to zero sending of the event is only controlled by its delay defined in the table.

The 300-series introduces gated trigger output feature on the EVRs. Using this feature it is possible to set up two completely independent trigger configurations on the same output and activate (or deactivate) them using available gates. These gates themselves are controlled by means of other events as configured by user.

Finally topology ids have been introduced. Each node in the timing distribution tree has a topology id. A node gets its topology id from its parent. This is a unique 4 byte value. Each EVM downstream port sends out the topology id to the child connected to that port. This is done by shifting the own topology id left by four bits and assigning the output port to the four least significant bits. The root of the tree has no parent and therefore its topology id is zero. Using this scheme, it is possible to construct the complete timing distribution tree by reading the topology id of all nodes.

IMPLEMENTATIONS

One consequence of maintaining flexible event rates is that event pattern changes at every machine pulse. Since EVM hardware has two event sequencers and we need to keep up with 100 Hz repetition rate, in each machine pulse one sequencer is playing back events while the other one is being programmed. The software process in charge must consider all event settings defined by user, which can be modified at any time, and decide in each machine pulse which events shall be programmed for distribution in the next pulse. The Timing Master Application (TMA) is the software we have designed to implement this process.

TMA is divided into two distinct layers. A lower layer which abstracts event object interface and an upper layer which takes care of combining all event object settings to fulfil operator's demanded beam rate. In the following we describe these two layers.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUCPL04

Event Object Abstraction

Event object abstraction allows all relevant controls on individual events independently. This abstraction provides the following interfaces:

- Enable/Disable
- Event code
- Rate (1, 5, 10, …, 100 Hz)
- Offset (pulse ID modulo)
- Delay (from start of the machine pulse)
- Multi-shot controls
- Priority
- Polarity (normal, inverted)
- Masking (soft- or hardware)

Another important parameter used by event object is the *pulse Id*. It is a global counter of machine pulses since the start of SwissFEL. The pulse id is maintained by TMA and distributed by the timing system. The pulse Id serves two purposes: pulse marking and for decision making whether to send an event in the current pulse or not.

The decision making is done as follows: if expression "*(pulse Id modulo rate) - offset == 0"* is true, the event has to be sent out. In each machine pulse all event objects are evaluated according to this logic and all events which have to be sent out are written to the event sequencer. Actually all events are written to the event sequencer in every pulse but the events which shall not be sent out have their event code replaced by 0 which tells the hardware not to send out the event. This is done for performance reason because it saves the software from recalculating relative event delays which is necessary if the sequence of events changes for each pulse.

An event object has two variants: *normal* and *soft*. Their only difference is in the settings of the event code value: the former has a non-zero event code and the latter has its event code set to zero. Consequently the normal event object directly controls generation of a hardware event code while soft event object is used to control other event objects. This is very useful in situations where we need to evaluate some conditions based on which transmission of other events must be controlled.

Delay specifies relative (global) delay of each event with respect to the start of the machine pulse. This can be anywhere between 0 to 10 ms range with resolution of the event clock (7 ns in SwissFEL).

Multi-shot allows configurable pattern generation for event transmission that is played on demand. It takes rate, offset and polarity also into account.

Polarity specifies the logic to send an event. For example, an event object at 10 Hz rate with inverted polarity sends its event code for 9 consecutive pulses and omits it for one pulse.

Masking allows overwriting of the TMA send decision. Setting the mask of an event object sets the mask bits in the sequencer table described in the hardware feature section. Consequently, this gives software or hardware inputs the means to overwrite the send decision. Hardware input is logically OR'ed with the software setting.

Beam Rate Control

The beam rate control is managed by upper layer of TMA. This layer provides the high level operator's interface. The available knobs are for example: RF fire1, RF fire2, laser as well as the desired beam rate. Based on these settings TMA should decide and control in which pulses beam generation is allowed and in which ones it must be stopped.

Stopping the beam is implemented based on the delayed trigger mechanism explained in the requirement section and illustrated in Fig. 1. To achieve this, we have a dedicated event which we call "RF-Off" event. A soft event object controls generation of this event by manipulating the masking input.

More precisely, triggers for the individual RF stations are delayed by a specified amount whenever RF-Off event is distributed.

For convenience and practical reasons we have provided a separate button, besides desired beam rate, that turns off the beam permanently when it is activated. This causes distribution of RF-Off event at all machine pulses until it is deactivated whereupon the last beam rate is restored. The same approach is used to permanently turn off the beam by delaying the triggers of the gun laser system (instead of RF). This is done using another dedicated event code (hence another event object) which we call Laser-Off. Turning off the beam with this method is desired when machine experts want to study the dark current of the machine.

MPS assistance is achieved by applying the MPS alarm signals to the RF-Off event object which overrides operator's settings for the beam rate. The high severity MPS alarm is directly connected to the EVM input which controls the RF-Off event masking. The states of the less critical alarm signals are read via a digital input card which is fed to the TMA to reduce the beam rate to specified settings.

Beam State Flag

It is important for certain systems to know if in a machine beam was generated or not. For example, data acquisition systems can decide to drop or keep the acquired data based on this information.

Beam state in a machine pulse is determined by knowing whether RF-Off or Laser-Off events were distributed in that pulse. If any of these events is distributed in a pulse no beam was generated, otherwise beam was generated in that machine pulse.

This information is evaluated in one system that checks arrival of these two events and provides the beam state to other systems.

Pulse Marking

The pulse Id is the global reference marker that uniquely identifies each machine pulse. It is generated at the master timing system and transmitted to all EVRs using

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUCPL04

event system data buffer feature. Every system with an EVR receives the pulse Id. In systems that use standard MRF products, i.e. VME or PCIe EVR, when new pulse Id is received in the data buffer, a control system PV gets updated through operating system interrupt. This PV is in our case an EPICS channel. Beam synchronous readout software uses this PV and tags it to all data collected in that pulse. Since there is no 64-bit (long long) data type through standard EPICS records available, we decided to represent pulse Id by "ao" at master (EVM) and "ai" record type on receiver side (EVR) which have double precision floating point value. The pulse Id must be able to accommodate whole expected life span of the Swiss-FEL facility.

COMISSIONING EXPERIENCE

Modular Software and Documentation

Since we knew that we will be a very small team for supporting time system users (only two engineers plus one two man-years of external support), we planned already at design time to provide tools and documentation to enable engineers at different expert groups integrate, setup and do first-level troubleshooting of EVRs on their own. This was not an easy task as the EVR is a complex device and its correct configuration for the end user is not so trivial. We aimed therefore for a very modular software design and good documentation. The modular software was given partly by the way we handle EPICS IOC modules at PSI where a modules contains driver, templates, default substitutions and default start-up configurations. As such the user needs only to make a call for loading the EVR module which in most cases can go with default settings. We provide two types of documentation to the users: user's manual, which contains exhaustive instruction to setup EVR, and a tutorial document that contains basic example to popular use cases and also application notes. Both refer to the configuration parameter of the EVR software module and accompanying standard EVR GUI's which expose a unified interface for all form-factors (VME and PCIe) to the user. Our experience up to now proves that a good though, modular control system software together with standardized GUI and good documentation is worth the effort.

Network Diagnostic

SwissFEL timing has a tree distribution topology which consists of up to five cascaded layers with one EVM as the top master and several distribution stations along the machine. Currently we have the following components in production:

- 67 EVM
- 45 PCIe EVR
- 152 embedded EVR
- 142 VME-EVR

Soon after taking the system into operation, we realized that tracking of the connection chains of the optical patches using labels , excel sheets or other manual methods are error-prone and not maintainable over time. Connections do not remain static on the long run, for example due to swapping of the patches on EVM ports for quick fix of a bad signal level or due to wrong patching after maintenance work. Without a good diagnostics tool troubleshooting becomes a tedious task in rather large networks. When a component (e.g. EVR) cannot lock to the upstream signal or a port on an EVM is defected, connection path must be known precisely in order to identify source of the problem and all other connected component that might be affected by this troubleshooting.

For this reason we proposed MRF to implement a mechanism in which every component in distribution tree knows his parent and the port of the parent it is connected to. Based on our proposal MRF implemented the topology identification as the following:

As described in the section of hardware features the event system provides topology information. For each component the topology ID is available through a register which can be read by control system software [3].

At PSI we have developed "TopoTool", a tool that fetches the device names from inventory and then retrieve topology ID from the control system at runtime.

Figure 2: Screenshot of the tool that captures and displays the actual timing network interconnect on the fly

All nodes that cannot be read, e.g. missing topology id or event link is down, or nodes that cannot be localized in the tree (e.g. due to a missing parent) end up in unsorted root node. A screenshot of this tool is shown in Fig. 2.

TopoTool can store the captured topology and compare the runtime captured topology with previously stored one. Under the column "TopoID stored/actual" in the screenshot, the topology ID's that have changed since the last capture is displayed in red. The compare function can also identify newly added or removed nodes (e.g. EVR or EVM) to the system. Furthermore by clicking on a node

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUCPL04

in the tree the expert GUI for corresponding device is launched where all status and properties of the device can be accessed.

External Super-period

In SwissFEL there are several subsystems that use different subharmonics of the machine reference clock. Coincidence frequency of these subharmonics is known as super-period. The timing system also distributes the super-period to all EVRs.

Initially we used to generate the super-period inside the EVM using a counter which was driven by the event clock. The resulting super-period was distributed via the distributed bus signal mechanism. Additionally, triggering of event sequencer (100 Hz repetition) in timing system is synchronized with this super-period.

With this scheme the event clock, the super-period and the events had a common phase. Due to the fact that the super-period counter started on a random event clock edge after reset, some systems in SwissFEL that depend on the two clocks had to be readjusted after every timing master restart. To solve this problem we are now using an external super-period which is derived reference clock. The relation of these clocks remains exactly the same, even if the timing master restarts. Furthermore, EVM measures the relative phase between external super-period and the event clock and allows selection of sequencer trigger with resolution of 90 degree of the event clock.

Delay Compensation

The MRF 300-series product support active delay compensation. Delay compensation is achieved by measuring the propagation delay of events from the delay top master EVM through the distribution network up to each EVR. At the last stage the EVR is aware of the path delay through the network and adjusts an internal FIFO depth to match a programmed target delay value. The delay measuring loop runs at a few KHz rate and compensate for the varying delay that is generated through the path due to varying conditions of all components on the way (including temperature effects).

At SwissFEL we have estimated a higher boundary which covers the largest distance between the master and the furthest EVR in the facility. The target delay of all EVRs is set to the value of 5000 ns. With the same target delay, all EVRs deliver their event stream at the same time. Starting from day one the delay compensation was active at SwissFEL and machine operation is relaying on this. However we have not yet done a thorough study of this mechanism due to lack of available machine time during the commissioning and also the short time available to reach the project milestones. We plan to do this at next possible time slot of the machine development. One case study can be to turn off the delay compensation for an EVR in a subsystem and see how the system drifts.

UPCOMING APPLICATIONS

Complex Triggering Application

The Aramis beamline of the SwissFEL has three end stations. At each end station there are various devices and subsystems which need to be triggered in a specified sequence. This is typically required by pump/probe experiments where one pulse excites (pumps) a sample system to a certain state which is subsequently probed after a well-defined delay with a pulse.

Control of such process requires defining a complex triggering pattern that goes beyond the capabilities of the TMA. Furthermore such trigger sequences are only of interest by specific end-station and not by the rest of the machine or even another end-station. As such it makes sense to implement a separate application that runs locally for each end-station. Since event system is very limited in number of distinct event codes (8-bit codes), using local event generation provides a bigger event code range for each end-station as they provide independent downstream. Figure 3 shows a typical local distribution of timing at SwissFEL machine and end-stations.

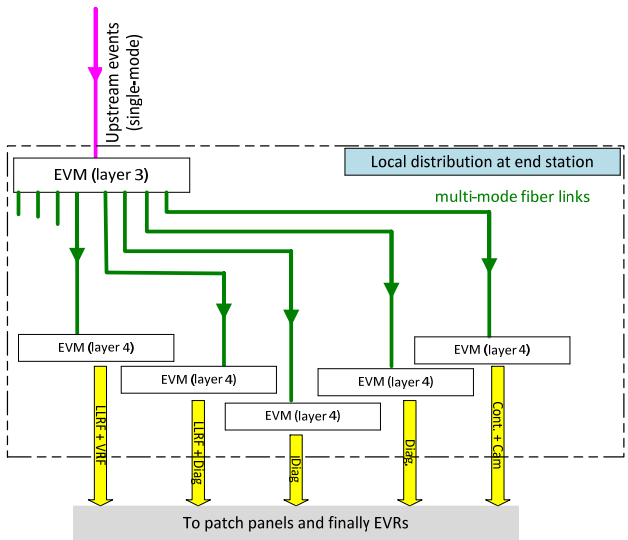

Figure 3: Typical local timing distribution system at SwissFEL machine or experimental end-stations

As shown, an EVM on layer 3 receives the upstream events via a single-mode optical link from SwissFEL master and redistributes it to several downstream EVMs via multi-mode links which are connected to EVRs in various subsystems. Layer 3 EVM generates local events in addition to forwarding of the global ones. Locally generated events should provide very flexible triggering sequence required at the end stations. The scientists need to be able to define an arbitrary sequence of different events. This is specified using a table with many entries. Each entry defines an event code and a delay value. The delay is in machine pulse units and specifies after elapsing of how many machine pulses the corresponding event code shall be emitted. Delays in all entries of the table are

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUCPL04

defined relative to the delay in first entry. The sequence can be started in three different ways: a) on a button press, b) at a specified pulse Id or c) upon reception of a global event. The software application which implements this functionality is called Complex Triggering Application or CTA for short. A subset of the event code space is assigned to local events generated by the CTA. This event code subset can be the same for all end-stations due to their separation. The implementation of CTA consists of two parts. The first part is a pure EPICS application running on the IOC in charge of the local EVM (layer 3 in Fig. 3). The second part is a client application implemented in pyQt and PyEpics which allows the users to define their sequence of interest.

The IOC side of CTA makes use of event objects, described in previous section. It maintains one event object for each local event. The client application transforms the sequence defined in the table to a set of arrays, one array for each local event. These arrays are written to the CTA IOC. When sequence execution is started, each array is used to drive the enable/disable input of one event object. As CTA iterates over the arrays to stimulate the event objects, the sequencer is programmed in each machine pulse to play the specified sequence of events.

Upstream Communication

The event system at SwissFEL is capable of upstream communication. There have been already discussions about applications where a reliable and synchronous distribution of the measured beam or machine parameter is required. Such information, which might be generated in principle in any subsystem, can be sent timing system upstream towards the master and redistributed globally to all systems. At the moment this is on our low-priority task list.

The design of segmented data buffer in 300-series among others was a result of discussions we had with MRF in order to find a way that individual EVRs do not overwrite each other's data when they send their data upstream. The segmented data buffer therefore provides possibility to assign each segment to a specific applications (and subsystems) and keep the communication protocol very simple.

CONCLUSION

We shared our experience and issues we had to handle for timing system during the commissioning of the SwissFEL. We presented new features of the MRF hardware and described how we employed them to fulfil operational requirements of the SwissFEL accelerator and its experimental stations.

REFERENCES

[1] PSI, https://www.psi.ch/swissfel/

[2] MRF, http://mrf.fi/

[3] MRF Delay Compensation Products Technical Reference 207.