A GENERIC TIMING SOFTWARE FOR FAST PULSED MAGNET SYSTEMS AT CERN

C. Chanavat, M. Arruat, E. Carlier, N. Magnin, CERN, Geneva, Switzerland

Abstract

At CERN, fast pulsed magnet (kicker) systems are used to inject, extract, dump and excite beams. Depending on their operational functionalities and as a result of the evolution of controls solutions over time, the timing controls of these systems are based on different hardware architectures that result in a large disparity of software solutions. A Kicker Timing Software (KiTS), based on a modular hardware and software architecture, has been developed with the objective to increase the homogeneity of fast and slow timing control for fast pulsed magnet systems. The KiTS uses a hardware abstraction layer and a configurable software model implemented within the Front-End Software Architecture (FESA) framework. It has been successfully deployed in the control systems of the LHC and SPS injection kickers, the SPS extraction kickers and the SPS tune measurement kickers.

INTRODUCTION

A kicker system must meet two requirements that are contradictory i.e. a high deflection strength and a short rise time of its magnetic field. Both properties are, for a given operational voltage, proportional to the product of the electrical current passing through the magnet and its length.

In order to meet these contradictory requirements, the length of the magnet is reduced by splitting the system into several independently powered magnet modules. Generally, the powering circuit of a magnet module consists of a set of well identified hardware components:

- A resonant charging power supply (RCPS) used to charge a pulse generator in order to decrease the time interval between two successive pulses and reduce the number of faulty shots;
- A line type pulse generator based on a pulse forming network (PFN) supplying quasi rectangular current pulses with variable length and amplitude;
- Up to three high voltage fast switches used to transfer in a controlled way the energy stored in the PFN to the magnet and to adjust the pulse length;
- A coaxial transmission line (TL) connecting the generator to ferrite type magnets;
- A ferrite magnet, built as lumped parameter delay lines and working in ultra-high vacuum;
- A termination resistor (TR) matched to the characteristics impedance of the generator, transmission line and magnet and absorbing the pulse energy supplied by the generator.

On this basis, the simplest kicker system comprises at least one PFN charged by one RCPS and discharged through the TL within one terminated magnet by one high voltage switch as shown in Figure 1.

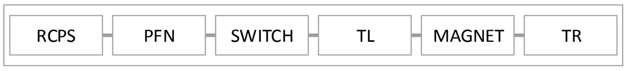

Figure 1: Simplest kicker system architecture

Although peak current, system impedance, pulse shape, pulse duration and repetition rate are different for every system, the different hardware components have been standardised as far as possible, and, depending of the kicker functions (beam injection, extraction or excitation), the number of RCPS, PFN, high voltage switches and magnet are combined in more or less complex architecture in order to provide the required functionalities. Different types of combination used in the SPS and the LHC are summarised in Table 1.

Table 1: Example of Kicker Architecture

System	RCPS per System	PFN per RCPS	Switch per PFN	Magnet per PFN
SPS Injection	4	2	3	2
SPS East Extraction	1	5	2	1
SPS West Extraction	1	1	1	4
SPS Tune	4	1	3	1
LHC Injection	2	2	2	1

Up to now the timing control of these different hardware combinations was based on dedicated real-time software strongly linked not only to the kicker hardware configurations themselves but also to the beam process where the system was used. With time, this approach has resulted in a high number of software packages to be maintained and to a high dependency of each software package with machine operation conditions.

Additionally, as this approach has been used for more than 40 years, a high diversity of electronic modules is used to generate the delays needed to trigger the different high voltage switches. As the access to these different timing delays is strongly embedded inside the real-time software, maintenance is now becoming more and more difficult due to the obsolescence of the delay modules and the difficulties to replace them without having to do a full re-engineering of the actual software.

In order to solve these two problems, a generic kicker timing software constructed on the basis of the existing set of standardised hardware components has been

ISBN 978-3-95450-148-9

developed and successfully deployed across a first set of kicker systems.

KICKER TIMING SOFTWARE (KITS)

The control of a kicker is divided into two functional subdomains:

- The *Slow Control* has the role to control the operational state (On, Off, Standby, Faulty) high-voltage pulsed generators and to protect the system against any internal failure modes.
- The *Fast Control* has the role to control the dynamic performance of kicker in accordance with the operational parameters driven by the control room and in perfect synchronization with the timing and the beam of the machine. Through the control modules of the equipment, it generates and distributes the references and the necessary stimulus for the RCPS and PFN to produce the kicks.

The KiTS, a generic software solution, has been developed for homogenisation of the fast control of all the kicker systems across the whole accelerator complex at CERN. Then it is highly bound to the RCPS and PFN hardware components, the control modules and the timing.

Architecture

The operation of the KiTS has to be real time because it should, depending on the settings sent by the control room, provide the required stimulus and references by following timing constraints that must be rigorously respected.

At CERN the FESA Framework [1] has been developed (Front-End Software Architecture) in order to develop object oriented classes for the control of accelerators equipment. FESA provides real time features linked to interrupts coming either from the accelerator central timing system or from the low level equipment hardware (timing or external events synchronization) and standard communication interfaces with external applications. The KiTS has been developed on the basis of this framework and thus benefits from all the features intrinsic to FESA listed above.

As highlighted in the introduction, kicker systems at CERN are based on a reduce set of hardware power components (RCPS, PFN, switch and magnet). Usually all kicker systems do not have the same number of components and their association can differ significantly from one equipment to the others.

In order to rationalise the kicker logical architecture, the association of RCPS with PFN can be called a *Generator*. On this basis, kicker systems can be represented as different configurations of generator (see Fig. 2) each composed of a set of RCPS and PFN.

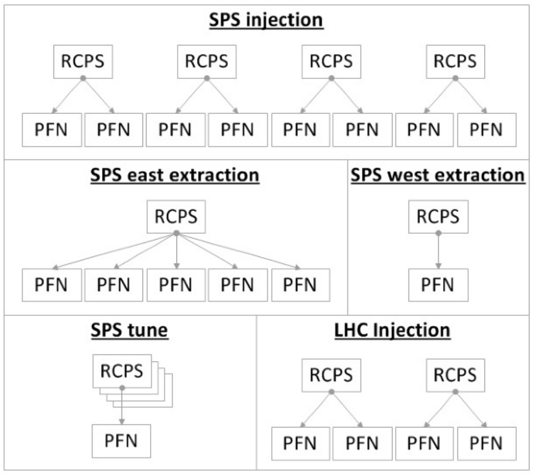

Figure 2: Simplified software model of different type kicker generator architecture

This hardware representation of a kicker system can then be easily modelled within software through an object oriented approach. The KiTS system offers a configurable object-oriented model consisting of three independent FESA classes (see Fig. 3).

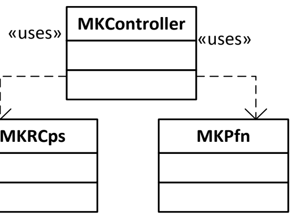

Figure 3: KiTS simplified FESA class model

The *MKRcps* FESA class handles the features of the RCPS hardware components. There are as many instances of the *MKRcps* class as the number of active RCPS elements required per kick in the system. This class manages the kick strength references sent to the power supplies and the timing delays related to the trigger the resonant charging process to load the PFN just before pulsing. It also controls the load balancing of the kick strength distribution between the different RCPS hardware components available for a kick.

The *MKPfn* FESA class handles the features of PFN hardware components (i.e. high voltage switches connected to it). There are as many instances of the *MKPfn* class as the number of active PFN elements required per kick in the system. This class manages the kick synchronisation with beam, the fine internal synchronisation when more than one magnet is used per kick, the kick length and some internal delays when pre-trigger signals are required. It generates the delays that trigger the switches present on the PFN in order to discharge it:

- A first set of delays that re-phase all PFNs relative to a START event synchronised with the beam;
- A second set of delays per PFN for staggering the PFNs with respect to each other.

The *MKController* FESA class is the controller of the KiTS system. It has only one instance (singleton) per system (kicker). It gathers and analyzes information from the control room and from the kicker to provide parameters and operating data to *MKRcps* and *MKPfn* classes. Furthermore, it can generate the required timing when it cannot be obtained by timing or external sources as an example when a kicker system has to be pulsed in local mode.

Each instance of each class has a specific set of configuration parameters in order to define the operational configuration. Thanks to this FESA model, the KiTS can be adapted to most of kicker generator configuration existing at CERN.

These 3 classes are also based on a second abstraction framework which introduces two standardisation layers (see Fig. 4):

- A business logic layer that homogenises the business common features of control applications in order to make the access to the hardware layer uniform, and to allow to share the business logic commands between the FESA actions.
- A hardware logical layer that homogenises the hardware common features of control applications, and makes uniform the access to the hardware factory.

Thus, we obtain a system of classes where all the classes have a homogeneous architecture where the common features are generalized into Frameworks and where the specific functions of the different logic layers are well separated.

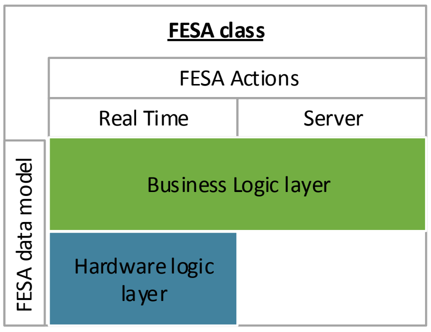

Figure 4: KiTS software layers

Hardware Abstraction

To interact with the active components of the kicker, the real-time tasks uses different types of hardware modules (see Tab. 2).

These hardware modules are not the same on the different kicker systems, then the KiTS must be able to use all these different types of hardware modules to control the different kickers. To do so, it uses a software library based on the factory design pattern which creates and allows to use abstract software objects such as ADC, DAC, DELAY, TDC, SAMPLER, etc. The factory hardware of each abstract type contains the implementation to drive several models of hardware module.

Table 2: List of Modules in the Hardware Factory

Category	Reference	Manufacturer
DAC	VMOD12A2 VMOD12A4	Janztec AG Janztec AG
AG	VMOD12E16	Janztec AG
Pulse delay	CTR V850 / V851 FMC-FD	CERN HIGLAND Technology CERN
Time-to-digital converter	TSM FMC-TDC	CERN CERN
Sampler	VD80 FMC-ADC	INCAA Computer CERN
Digital I/O	CTR VMOD-TTL	CERN Janztec AG

For example, the DELAY hardware factory contains the implementation of the following modules V850, V851, CTR, FMC-FD as shown in Fig. 5.

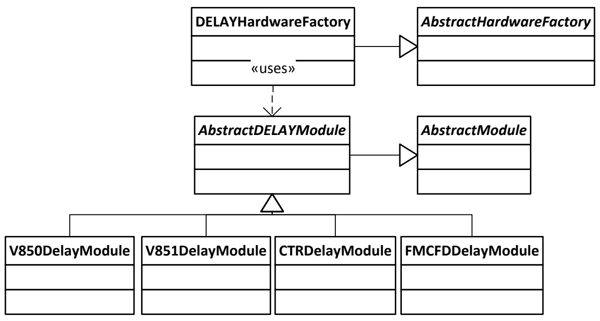

Figure 5: The Delay module factory simplified class model

The list of hardware modules of a factory can be change without any impact on the client applications code as a generic API has be defined for each category of module.

These abstract software objects allow the KiTS to control the hardware modules of any type without knowing the exact implementation of the hardware modules used.

Therefore, this abstraction layer allows the KiTS to be fully independent of the models of the hardware modules used for the control of the kickers.

Timing

Kicker systems work with different timing sources according to the accelerator to which they are used (LHC, SPS, PS…) and with different families of timing events depending of the beam process for which they are used (injection, extraction, tuning). In some cases, some specific events can also be generated by the real-time control system itself (SPS tune kicker of SPS dump kicker) a defined in dedicated configuration files.

To adapt to the different type of kicker timings, the KiTS is based on an event model architecture scheduled by four virtual events. This virtual event model serves as generic timing structure for the KiTS to drive a kicker. These four virtual events are associated by configuration to hardware timing events from the accelerators central timing.

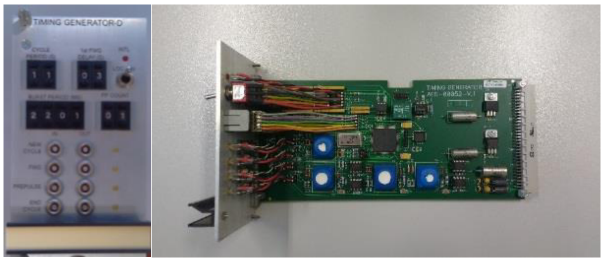

Figure 6: TG-D Module

Before reaching the KiTS, the timing hardware events are controlled by a Timing Generator module (see Fig. 6) providing the following functionalities:

- Control of the correct scheduling of the four events;
- Generation of the four events when the kicker is operated in local;
- Inhibit of all events in case of interlock condition coming from the slow control.

The TG-D module also plays the role of local timing generator which allows pulsing the kicker independently of the central timing.

From the four primary events, it is possible to generate as many secondary events as needed to control a kicker (see Tab. 3). To be noted that a full beam process is always enclosed between a *Start Cycle* and an *End Cycle* event and that between these two events, as many sub-processes as desired can be played.

Table 3: List of Primary and Secondary Events

Primary events	**Secondary events**
Start Cycle (NC)	
Forewarning (FW)	Forewarning prepared Before RCPS Before prepulse After prepulse
Prepulse	
End Cycle (EC)	

These primary and secondary events are used to trigger the KiTS FESA class real time actions that will call the KiTS FESA class business logic commands (see Tab. 4).

This model of virtual events allows the KiTS to be easily adapted to specific timing requests of different kicker systems in operational mode or test mode.

Table 4: List of KiTS FESA Classes Commands

Class	**Commands called by RT actions**
MKController	• Reset Equipment • Get User Context • Start / End Timing Cycle • New / End Cycle • Forewarning, • Prepulse, After Prepulse • WatchDog
MKRcps	• Reset Equipment • New / End Cycle • Write / Read / Reset Vref • Start / Stop Sampling • Write Trigger • WatchDog
MKPfn	• Reset Equipment • New / End Cycle • Write Fine Tuning • Write Prepulse • Write / Read Fast Timing • Single PFN Voltage Acq • Start / Stop Sampling • Check / Notify PFN Voltage • WatchDog

CONCLUSION

Thanks to its object oriented architecture, its virtual timing model and its hardware abstraction library, the KiTS has been successfully deployed on different types of kicker system at CERN.

During LS1, the end of support of FESA2.10 has required the migration of the KiTS real time kernel to FESA3. This update has involved an extensive re-engineering of the real-time code in order to remain fully compliant with the new FESA framework. After the successful migration and series of validation tests, the KiTS has been easily deployed to all kicker systems already equipped with it proving the benefits to use a generic approach.

For the kicker systems not yet equipped, the use of the KiTS instead of developing a dedicated software solution has proven to be a more optimised approach reducing significantly development and validation resources, provided that all the required functionalities are already integrated.

The KiTS will be facing more challenges that will continue to profoundly assess its flexibility, modularity and adaptability. Its planned deployment on the renovated PS Booster distribution and consolidated SPS Beam Dumping kicker systems are good examples.

REFERENCES

[1] M. Arruat et al., “Front End Software Architecture”, WOPA04, ICALEPCS’11, Knoxville, TN, USA, 2011.