Enhancements of the Filling Pattern Controls at the Swiss Light

B. Kalantari, T. Korhonen
Paul Scherrer Institute, Villigen, Switzerland

ABSTRACT

Right after commissioning, the Swiss Light Source (SLS) operation was started with top-up being the most common operating mode. The control of the storage ring filling pattern was in the beginning limited but covered the basic needs. With time the demands from machine experts, experiments and beamlines have grown and to support these we have added applications to support new operating modes and a filling pattern feedback (FPF). As multiple subsystems (i.e. IOC's running EPICS) are involved, we have had conflicts between the applications. To resolve these conflicts we have redesigned the software algorithms for injection control in a way that guarantees the coordination between different subsystems that have to function synchronously. We designed the application so that the real-time actions are synchronized via timing system and do not depend on the network (EPICS channel access). In this paper we describe our experiences and motivations for upgrading and the methods as well as algorithms involved in our software redesign.

INTRODUCTION

As top-up operation [1] is becoming routine in many advanced third generation synchrotron light sources, applications related to it are also highly on demand [2]. One of the delicate and challenging issues in this context is the control of bunch pattern in top-up mode [3]. Although the principles of bunch pattern controls are very similar, the implementation can vary from one synchrotron light source to another. The implementation and quality of such kind of controls of course depend on control system, timing system, diagnostics and injection system, their flexibilities and coordination between the systems involved.

Bunch pattern controls can be implemented using open loop and feedback methods. In the open loop method no charge per bunch information is used. Depending on the flexibility of the timing system and controls, one or more specific sequences of the filling can be achieved. These are in principle different flavours of targeting one bunch (or a chain of bunches) sequentially one after the other. In the top-up mode with this method only the total accumulated charge in the storage ring determines when to start or stop injection process. Depending on the controls and timing system this method may still provide with required filling patterns in a limited extent due to lack of charge per bunch information.

In the feedback method in addition to all controls in the open-loop method, we have information about individual bunch currents. Using measured bunch by bunch current various filling strategies can be implemented. In principle in this method it is possible to achieve and maintain any arbitrary bunch pattern in the storage ring [3]. Individual bunch currents are measured using different methods, e.g. measuring the signal from beam position monitor (BPM) pick-ups, avalanche photo diodes (APD) or photon multipliers.

When all required systems are available and have the required quality to perform bunch pattern control using the feedback method, the remaining concern is how to design and implement the whole process so that it guarantees a reliable and safe operation. In the following we describe the problems which occur in practice and how we could handle them with a flexible design.

COORDINATION OF DISTRIBUTED COMPONENTS IN FPF

The SLS control system is based on the EPICS toolkit[4]. Almost every component is controlled via EPICS. The major control subsystems contributing in FPF can be summarized as the following:

- Bunch pattern acquisition system

- Linac timing
- Overall timing

At the SLS (controls of) each of these components are integrated in an IOC (Input Output Controller). IOC's are running EPICS and perform real-time controls under vxWorks using dedicated hardware and local CPU cards.

Bunch pattern acquisition consists of a fast digitizer which reads the bunch pattern provided by an APD [3]. The digitized filling pattern is then conditioned and processed to provide a list of the bucket numbers to be targeted at the next top-up cycle. This is done by comparing the actual bunch pattern and the reference bunch pattern which can be of any arbitrary shape like a rectangle or a sine wave or a comb etc.

Linac timing provides the synchronization and triggering of the linac in order to make a successful injection of particles in a specific bucket number in the booster synchrotron.

Overall (main) timing IOC is the central component that coordinates all the synchronous software/hardware actions by sending timing events for example to trigger pulsed magnets at the right time in order to extract accelerated particles from the booster and inject them into the specific RF bucket in the storage ring. Distribution of the timing signals (event) is done via a dedicated network which connects the overall timing IOC (event generator) to all the timing signal (event) receivers in other IOC's in a star shape.

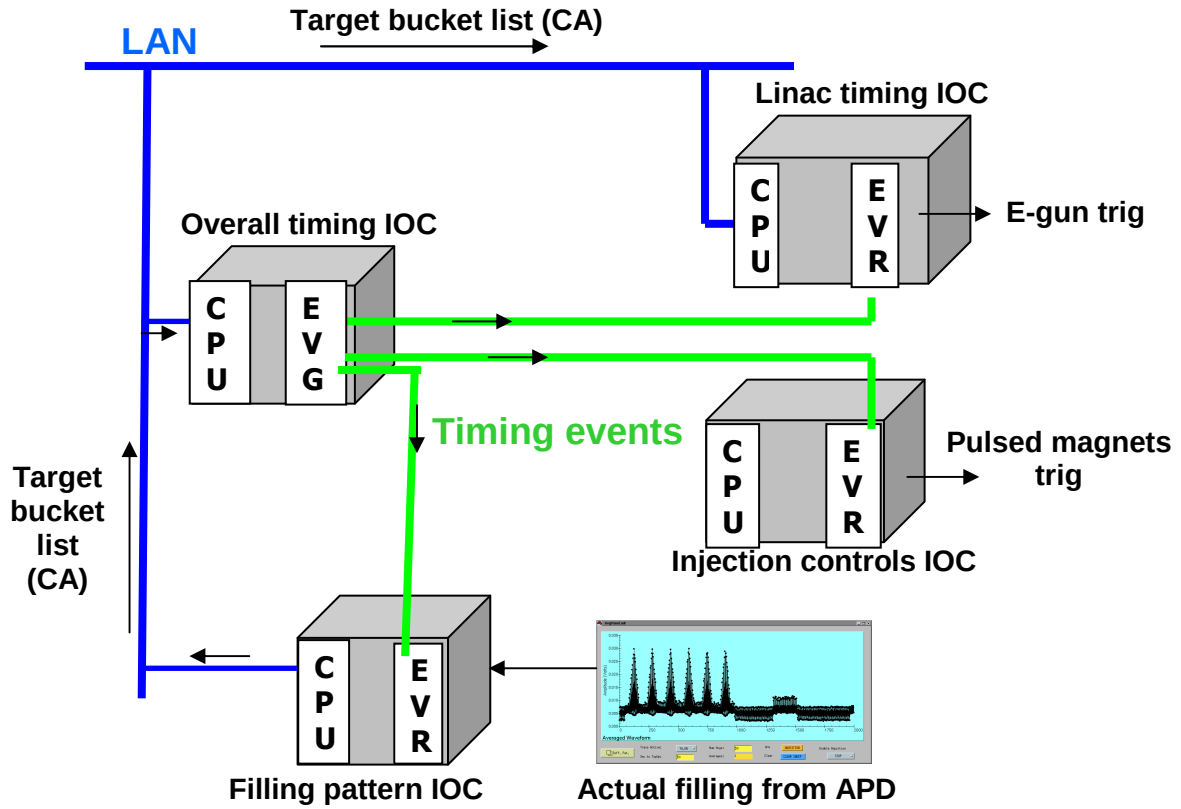

Figure: relation of the subsystems involved in filling pattern controls

Originally in the filling pattern controls application, we could just determine start and end of bunch train and their spacing and number of shut per bunch in a very simple application. A bucket counter

could be incremented by programmable steps from the start till the end of the bunch train. Whenever it reached the end it rolled over to the starting bucket number again and so on. This worked out quite fine at the time we had no FPF involved.

The filling pattern feedback system then separately was developed with a dedicated IOC in charge of the controls of that system. The task of this IOC is to get the bunch pattern and according to some algorithms and having bunch current information to specify the next buckets to be injected to in the coming top-up cycle.

The problem comes up where the bucket numbers to be targeted are determined in the bunch pattern acquisition (filling pattern) IOC and the overall timing as well as linac timing need to know each bucket number in order to perform synchronised action to achieve injection in that bucket at each injection repetition rate (in SLS case 3 Hz). The only way to communicate the bucket numbers is the network and more precisely CA protocol in EPICS (see Fig. 1). The non-real-time nature of the network cannot guarantee sending the information shot by shot (e.g. 3 Hz). It happens occasionally that these transactions do not reach the destination at the same time due to condition of the network and its instantaneous traffic load, etc. As result of such delays sometimes the target bucket number in overall timing IOC is not the same as that of linac. Therefore the required coordination is lost and the particles are injected to a wrong RF bucket in the storage ring. That produces in turn an uneven filling pattern which may even affect the orbit stability in some extent.

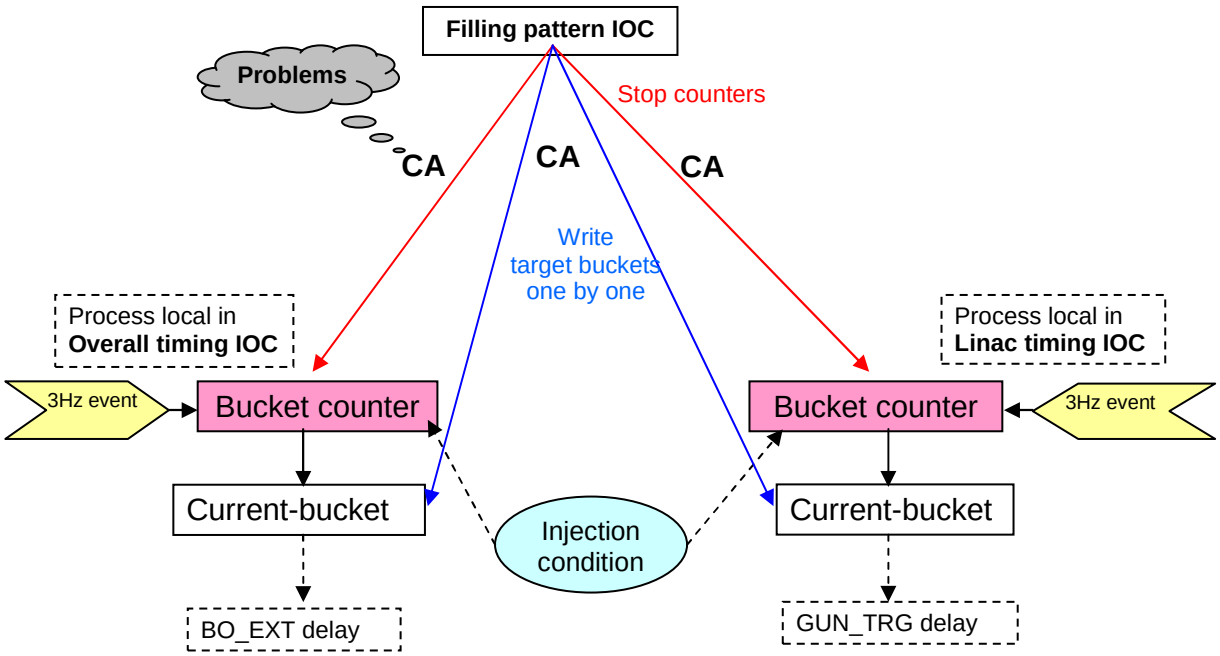

Figure 1: communication of the IOC's in filling pattern controls

The other problem that we used to suffer from was handling of the camshaft mode which had a dedicated software application in the open-loop bunch pattern controls. The method to inject to single isolated (camshaft) bucket was to stop (freeze) sequential bucket (Fig. 1) counter which fills the bunch train and change the bucket numbers to be filled to the camshaft bucket number. Of course the bucket number had to be sent to linac and overall timing IOC via CA and the previous problem shows up again. The result was in this case even worse because sometimes releasing the sequential bucket counter from the frozen state could not occur in time (synchronous to other conditions) and it could result to a very high peak current in a single bucket.

To address these problems we tried to define a single method which could handle all modes of the bunch pattern controls and guarantee the real time behaviour in all cases avoiding the nondeterministic behaviour of the (Ethernet) network.

In the new approach (Fig. 2) instead of the bucket number counters (old method) the application maintains an index counter of an array holding the target bucket numbers in both timing IOC's. The local applications need only to increment or reset the index counters synchronously. The perfect synchronization is achieved then by controlling the index counters using the global timing events. For example at the SLS the 3 Hz event of the injection repetition rate is used to increment this counter for each successive injection shot. Each IOC at a time and synchronously should know the current target bucket number (CUR_BKT) variable in order to adjust the required timing. The negative numbers indicate no injection should occur (Fig. 2).

The filling pattern IOC on the other hand, would care just about the content of the arrays holding the target bucket numbers which in turn depend on mode of accelerator operation, desired filling pattern, FPF status (active/idle) etc. It writes the array of the target bucket numbers to both timing IOC's through network via CA. However, this is much less frequent than injection repetition rate. If the FPF is active (feedback method) the arrays are updated each top-up cycle which is something between 3 to 5 minutes depending on the beam lifetime and top-up current dead band, if FPF is idle (open-loop method) then target bucket numbers (arrays) will be changed only when operator asks for a new pattern different from previous one e.g. changing from a simple bunch train to camshaft mode.

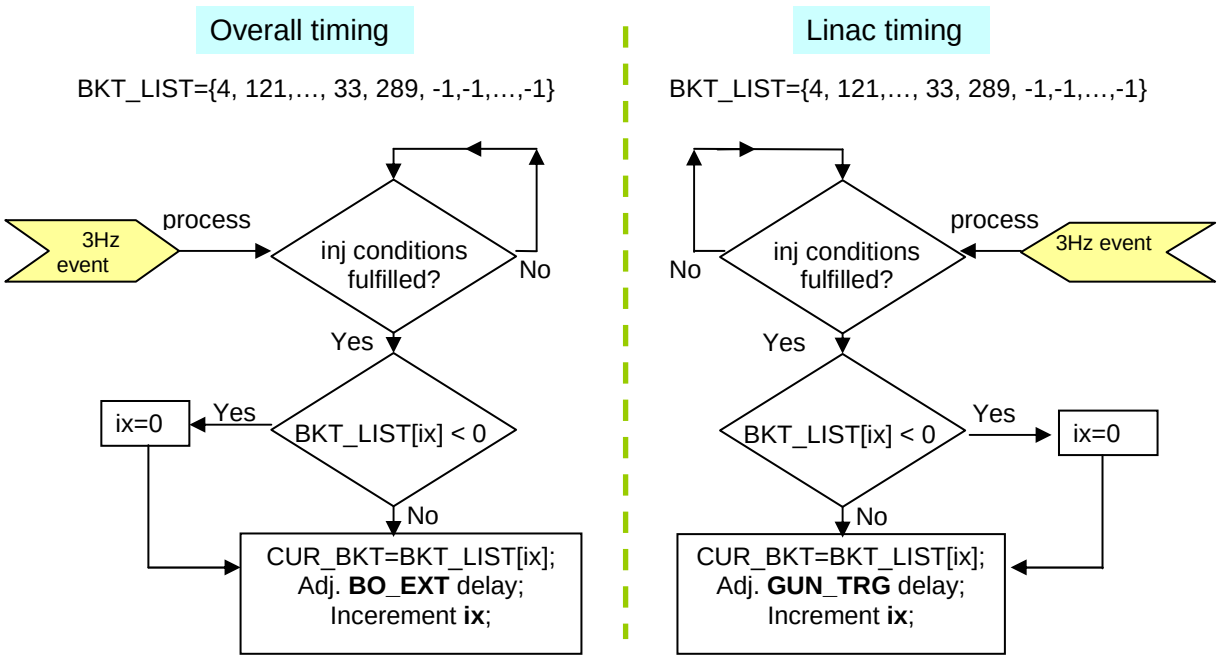

Figure 2: Injection synchronization in Linac and overall timing IOC

FILLING PATTERN FEEDBACK (FPF) ALGORITHM

Generally in the filling pattern controls there is always a reference pattern and the application tries to establish that pattern in the storage ring. The prefect case is achieved when the FPF is active, then the actual charge distribution in the storage ring can be a perfect match to the reference pattern and the amount of charge in each single bucket can be controlled very well.

The FPF algorithm is basically implemented as the following steps:

- Acquire the actual filling pattern in the storage ring
- Compute the difference of the scaled actual and reference patterns
- Specify the next target buckets to minimize the difference of the actual and reference pattern

Reference Pattern Generation

Part of the application in the filling pattern IOC is in charge of the reference pattern generation (see Fig. 2). That has three modes or pattern options: *Default*, *Camshaft* and *Define*.

In the first case (Default) it generates a simple bunch train from a start point until an end point which can be set by the user. The Camshaft mode takes more parameters in addition to the Default case which are single isolated bucket number and its amplitude. In fact it generates a bunch train plus an isolated bucket anywhere in the range of the total bucket numbers (harmonic number of the storage ring) with any desired amplitude. In the Define mode, the reference pattern is read from a text file provided by user or experts. The text file is a column of numbers which represents the desired relative charge distribution. Each row corresponds to a bucket number. In our case (SLS) this would be a single column in 480 (harmonic number) rows. A zero value in this column shows no injection in that bucket number.

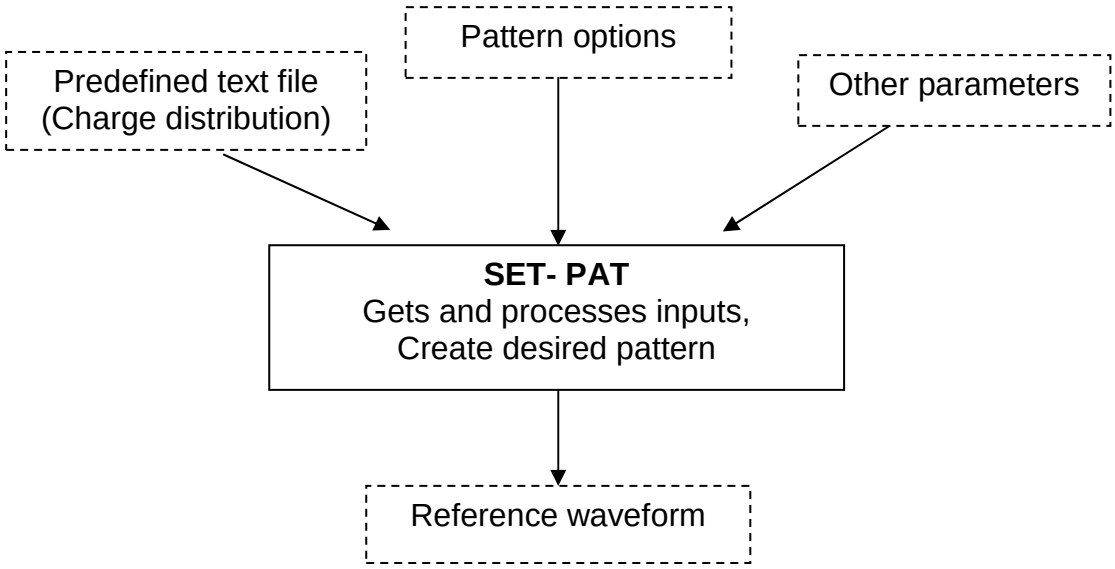

Figure 3: Reference Pattern Generation

FPF algorithm

When the reference pattern is determined and the user starts FPF, the algorithm shown in Fig. 4 is executed after finishing each top-up cycle. By a top-up cycle we mean a series of injections which results in filling up the storage ring with dead-band current. For example SLS is operated now at 350 mA top-up plus 1 mA dead-band. A top-up cycle then means process of consequent injections (in different buckets specified by FPF) to fill up the ring when it goes less than 350 mA until it reaches 351 mA.

Whenever a top-up cycle finishes overall timing IOC distributes a global timing event. The event receiver in the filling pattern IOC receives that event (as all event receivers do) whereupon the FPF algorithm (Fig. 4) is executed. The application first checks if FPF is in active state which means to check if FPF has been requested by the operator and if the beam current is above a certain limit. Then it starts bunch pattern acquisition by enabling a fast digitizer to read the pattern for many turns and average them. Then the acquired pattern is transferred to the IOC, scaled and subtracted from the scaled reference pattern. This difference waveform is then sorted from the smallest element to the biggest to determine how far the charge in each bucket is with respect to the reference pattern. This determines injection priority of each bucket for the next top-up cycle. The FPF application keeps each

time a list of the previous bucket numbers which were targeted and at each turn of execution verifies the result by comparing that with the difference waveform. If it finds out that most of the targeted injections were not successful it tries to find the offset and stops. A calibration procedure has been foreseen to correct the offset. When the FPF stops for what ever reasons the filling pattern controls switches to the open-loop mode but still to try to follow the reference pattern in a very limited extent of course.

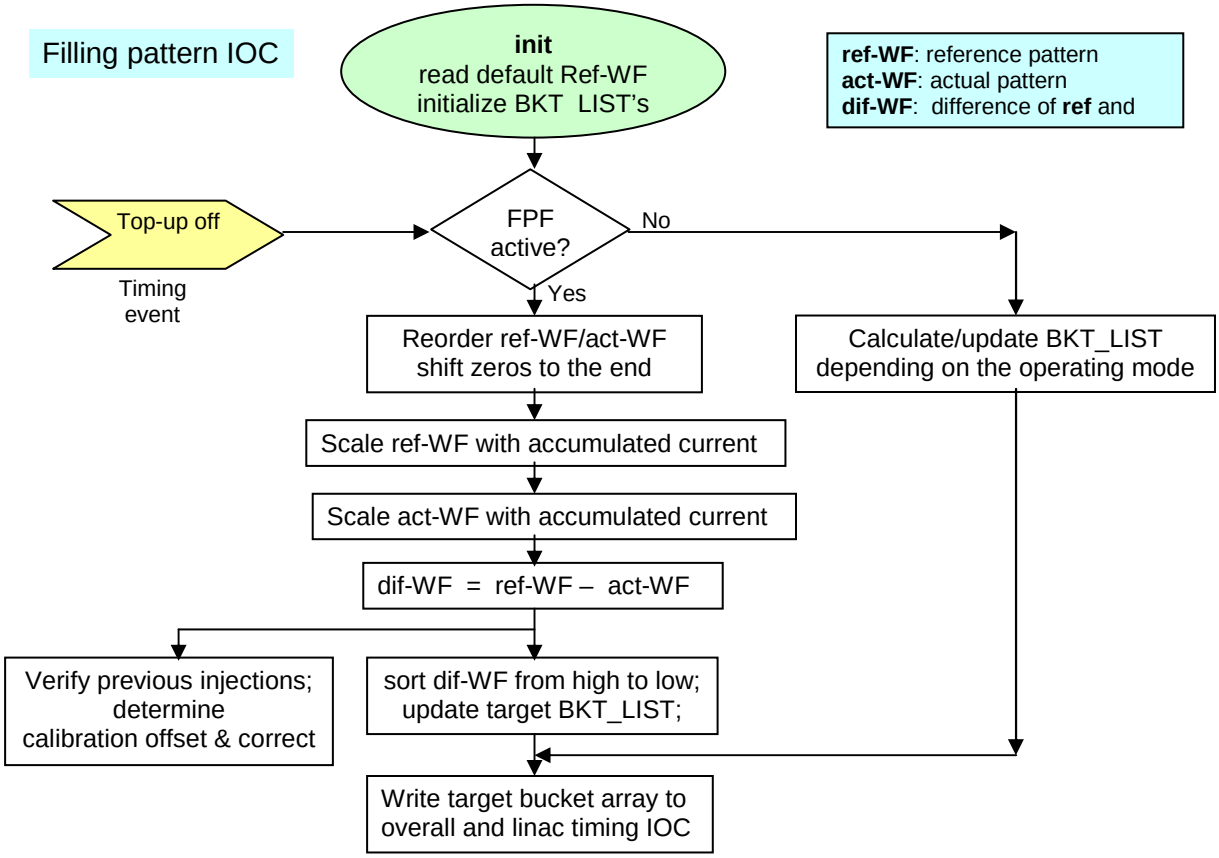

Figure 4: FPF algorithm flowchart

CONCLUSION

By changing the filling pattern software structure we have achieved reliable and robust controls over the filling pattern mechanism at the SLS. One of the main point was to understand the real-time requirements and try to separate them from no-real-time ones. The real-time actions are controlled then via the global event system (timing system) and the rest of the controls are handled via the network and CA.

REFFERENCES

[1] M. Munoz and A. Luedeke, "Top-up Operation at the Swiss Light Source", EPAC02, Paris, France, June 2002.

[2] B. Kalantari and T. Korhonen, "Enhancements of Top-Up Operation at the SLS", EPAC04, Lucerne, Switzerland, July 2004.

[3] B. Kalantari, T. Korhonen and V. Schlott, "Bunch Pattern Control in Top-up mode at the SLS", EPAC04, Lucerne, Switzerland, July 2004.

[4] http://www.aps.anl.gov/epics/