AUTOMATED OPERATION OF THE METROLOGY LIGHT SOURCE STORAGE RING

Thomas Birke[@], Helmholtz Zentrum Berlin für Materialien und Energie[†], Berlin, Germany

Abstract

The Metrology Light Source (MLS) is in user operation since April 2008. This versatile facility has to work at energies ranging from 105 MeV up to 629 MeV, operating currents from a single electron up to 200 mA and different values for the momentum compaction factor according to the user demands that vary even on very short notice. A software system has been developed to control and coordinate the broad manifold of machine states.

Design goal of the software was to keep and transfer machine and control system within well-defined and consistent states. It should minimize errors due to inadvertences and avoid mistakes when following complex command sequences. The software is modelled as a finite state machine. Actions are configured and triggered by a few high level commands. This paper describes program functionalities and interfaces. Experiences with *automated operation* using this indispensable operator tool to reliably set up a very sensitive machine are reported.

MOTIVATION

The Physikalisch-Technische Bundesanstalt (PTB), a main customer of the BESSY II facility, is the owner of a low energy electron storage ring, the Metrology Light Source (MLS), located close to the BESSY II storage ring in Berlin. The MLS has been designed and built by HZB[†] according to the specifications of the PTB and is also operated by HZB staff. It offers user service since April 2008 and is now running in routine operation.

Table 1: Machine and Operating Parameters of the MLS

Circumference	48 m
Revolution Time	160 ns
Injection Energy	105 MeV
Operational Energy	105-629 MeV
Beam Current	1 pA-200 mA
Values for Momentum Compaction Factor α	$10^{-4} – 3x10^{-2}$
Insertion Device	Electromagnetic Undulator 23x180 mm

Table 1 shows that the MLS has a wide range of operating modes and parameter settings. Additional demands on operating the machine emerge from the use of an electromagnetic undulator.

A ramping procedure was developed, that keeps the electron beam stored not only when ramping the energy *up* but also when ramping *down*. This way the energy ramp acts at the same time as a degaussing cycle. But as it does not drive the storage ring magnets into full saturation, some remanent fields cannot be cleared and strongly influence the machine dynamics. As a consequence any error in setting a magnet power supply amplitude or polarity can strongly deteriorate the machine performance and leave the machine in a different state even after completing the time consuming special designed degaussing procedure.

Therefore it is crucial to avoid any operating error when establishing the desired user state in the MLS, which is best realized with completely predefined and automatically performed set up procedures.

Another motivation for a high degree of automation originates from the fact that MLS commissioning work and operation is a service provided by HZB to the PTB as a customer service. It should be as reliable and transparent as possible demanding user friendly interfaces and operation definitions.

Operating the MLS includes injecting beam up to a desired current, ramping the energy and adjusting the momentum compaction factor α. All these services require multiple actions to set up the machine for the mode requested by the users.

Since all signals that are required to determine the necessary steps are available as control system process variables (PVs), the decision was made to develop a software system performing the essential sequences of actions to get the machine into the desired states.

SOFTWARE SYSTEM

Finite State Machine

Based on experiences with smaller applications at BESSY II as well as at the MLS, the described software was developed as a hierarchical set of state machines.

Finite state machines (FSM) are a well proven software concept to model and control behaviour of complex systems. A finite state machine – in this case a transducer that converts input (events) into output (actions) – consists of a set of all possible states of the modelled system along with all possible transitions between these states. The transitions are unambiguously performed on conditions associated with input events. Any transition as well as entering or leaving a state may initiate output actions. States describe possible situations of the whole system while transitions define when (condition) and how (action) to transform the system into another state.

In a controls application, the input of a finite state machine usually consists of events resulting from

[@] Thomas.Birke@helmholtz-berlin.de
[†] By the merger with the former Hahn-Meitner-Institut (HMI), BESSY became part of the new Helmholtz Zentrum Berlin für Materialien und Energie (HZB)

incoming changes of PVs from the underlying control system (EPICS, Experimental Physics and Industrial Control System), timer-events (esp. timeouts) and of course actions initiated by the user (e.g. using the graphical user interface). The output typically is any sequence of statements/operations limited only by the software environment, but in particular writing new values to the control system (modify PVs) and give feedback to the user.

The very first version of the described program was a simple beam scrubbing automaton, Fig. 1. After the MLS had been commissioned to a point where beam-accumulation and ramping to the highest energy was possible, it was important, to keep the beam-current at this high energy above a certain limit during nights and weekends in order to improve beam-conditions.

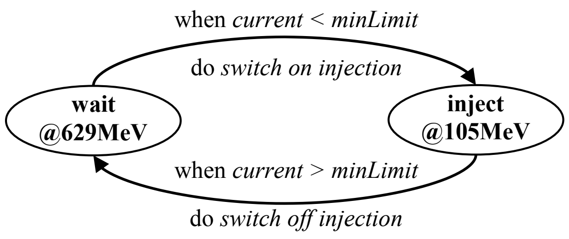

when *current* < *minLimit*
do *switch on injection*
wait @629MeV
inject @105MeV
when *current* > *minLimit*
do *switch off injection*

Figure 1: Simplified sketch of the first version.

This software has since then evolved into an important helper application. The state machine as it is currently used was not developed by design according to a full specification. It has undergone an evolutionary process influenced by experiences from machine commissioning as well as from daily use of the application itself. Only by using the application, it is possible to detect situations not yet handled by the state machine (often even by operating errors). The procedures described by the user to solve these problems are then implemented in the state machine and undergo a refinement phase based on the experiences using them. Numerous small development steps have been made, some of which were later removed in favour of alternative solutions or have simply proven obsolete during the commissioning process. A clear view of what actions are appropriate to setup a certain state often eventually arises from formally describing the solution in close cooperation of developers and users/scientists.

During this process, the application became an indispensable instrument performing all standard actions the operator has to take care of. It is an attempt to fill the gap between basic device control and the so-called *one-button-machine*.

By now, the main state machine (the *Operation Master*) consists of ~70 states and ~150 transitions. This state machine not only describes the command-sequences necessary to set up certain machine states, but also includes handling of otherwise unexpected conditions and implements solutions to maneuver out of these exception states into the desired or another safe state.

The whole system was running the MLS almost without any human intervention for about two weeks during holiday break 2008/2009 and performed well, Fig. 2. The only glitch was a microtron dropout that had to be cared for manually. This particular intervention is now part of the action-sequence to recover from microtron errors in the *Operation Master*, giving an example of how the system evolves by practical use.

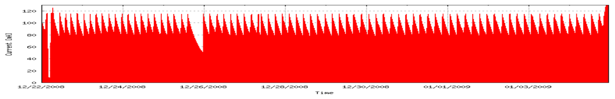


Figure 2: two-week run during holiday break 2008/2009.

Energy Ramp

Energy Ramp is the separate application to ramp the energy of the stored beam. Injection always runs at 105 MeV, and after injection has finished, the machine has to be ramped to the desired energy (up to 629 MeV). The driving parameter is a software parameter that corresponds to the energy. This parameter is smoothly driven to the desired end-value and is used as the input-value to an interpolated breakpoint table for the setting of each participating device. All devices are synchronously driven to the desired energy minimizing beam-loss.

These breakpoint tables are created with a semi-automated process. The machine is setup for a certain energy and all magnets are hand-optimized. When finished, a special command stores the current settings in the appropriate breakpoint tables.

Ramping usually is as simple as setting the target energy and hitting the "Go" button, but due to hysteresis in the magnets, ramping up and down has to be done using two different sets of breakpoint tables to not lose beam during ramp. These tables are only allowed to switch at the end-points of the energy ramp, where they basically map to the same values. Hence ramping to an arbitrary energy may require ramping to the endpoint in previous ramp-direction first before ramping to the desired energy (e.g. 105MeV ↗ 450MeV ↘ 300MeV won't work, must ramp 105MeV ↗ 450MeV ↗ *629MeV* ↘ 300MeV instead).

This constraint is not implemented in the *Energy Ramp* itself, but has to be followed by the caller. The *Operation Master* takes care of this, and switches tables properly.

The *Energy Ramp* is one example of a separate application that also uses a state machine as its controlling entity. All interaction with this application is handled via control system PVs.

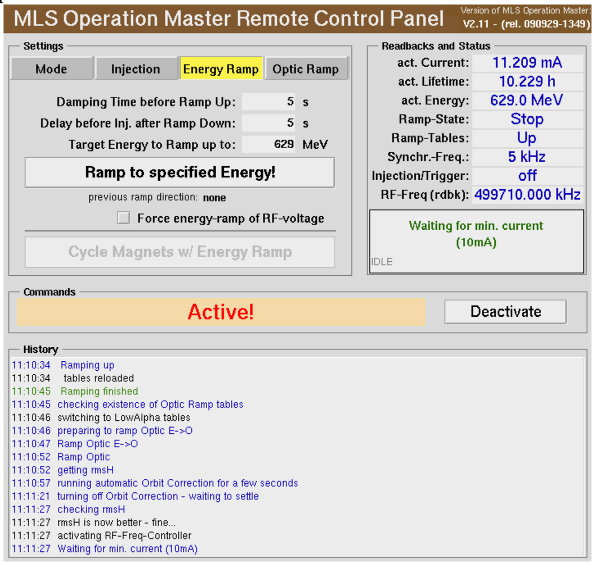
Optics Change

At defined energies another software module is used to change the optics of the machine (drive the momentum compaction factor α). The application itself is very similar to the energy ramp with the driving value corresponding to the synchrotron frequency.

After performing an Optics Change, the *Operation Master* configures and optionally activates the Orbit Correction Application for a while to optimize the beam position of the highly sensitive beam. Finally, a feedback system is started to keep the beam position stable by just manipulating the RF master clock.

Operation Master

Both applications are controlled by the *Operation Master* so the operator just defines and sets the main parameters (target current, energy and synchrotron frequency …) and then issues the initial command to start transition. The *Operation Master* takes care of all necessary steps to get from the current state to the desired parameters which may involve any subset of injection sequence, energy-ramping and optics-change. It is the central controlling instance keeping track of several decentralized non-linear processes. The *Operation Master* itself controls/monitors 45 devices and serves 28 PVs to control its own behaviour. The controlled 45 PVs include the Energy Ramp and Optic Ramp which both in turn operate on about 50 devices.


MLS Operation Master Remote Control Panel
Version of MLS Operation Master:
V2.11 - (rel. 090929-1349)
Settings
Mode
Injection
Energy Ramp
Optic Ramp
Ramp to specified Energy!
Cycle Magnets w/ Energy Ramp
Readbacks and Status
Commands
Active!
Deactivate
History

Damping Time before Ramp Up:	5	s
Delay before Inj. after Ramp Down:	5	s
Target Energy to Ramp up to:	629	MeV

previous ramp direction: none

☐ Force energy-ramp of RF-voltage

act. Current:	11.209 mA
act. Lifetime:	10.229 h
act. Energy:	629.0 MeV
Ramp-State:	Stop
Ramp-Tables:	Up
Synchr.-Freq.:	5 kHz
Injection/Trigger:	off
RF-Freq (rdbk):	499710.000 kHz

Waiting for min. current (10mA)

IDLE

```
11:10:34  Ramping up
11:10:34   tables reloaded
11:10:45  Ramping finished
11:10:45 checking existence of Optic Ramp tables
11:10:46 switching to LowAlpha tables
11:10:46 preparing to ramp Optic E->O
11:10:47 Ramp Optic E->O
11:10:52 Ramp Optic
11:10:52 getting rmsH
11:10:57 running automatic Orbit Correction for a few seconds
11:11:21 turning off Orbit Correction - waiting to settle
11:11:27 checking rmsH
11:11:27 rmsH is now better - fine...
11:11:27 activating RF-Freq-Controller
11:11:27 Waiting for min. current (10mA)
```

Figure 3: *Operation Master* at work.

IMPLEMENTATION

The current implementation is an application written in Tcl/Tk. Tcl/Tk is a proper choice for rapid prototyping and development of an exemplar application including a graphical user interface. But as the system settled and stabilized some drawbacks of a monolithic application became immanent. To avoid conflicts, the system has to ensure that only one instance is actively running, and hence the application itself and the current status of the *Operation Master* used to be visible on a single screen only. The latter shortcoming has been solved by separating the GUI from the application itself. The *Operation Master* is now using control system PVs for any kind of interaction as well with the machine itself as with other applications and of course also the user. A standard control system panel is used as a GUI to monitor and control the *Operation Master*, Fig. 3. Additionally, all other EPICS tools can be used to monitor activities. As an example, the alarm-handler is used to alert/notify operator and others on unexpected events and the archiver is used to log activity of the *Operation Master* for diagnostic and development purpose. This way, the application integrates very well into the existing control system infrastructure.

State Machine

The state machine consists of a set of *states* and all *transitions* between these states as well as all *possible actions* to be performed during transition or when entering a state.

The Tcl implementation is completely data-driven. The state machine is specified using a data-structure of nested lists and hashes/dictionaries. It is very simple in structure, but nevertheless powerful and easily extensible.

State Engine

The state engine is the actual processor that not only parses the state machine data-structure, but also runs the state machine. On external events, it checks for all possible transitions of the current state and also manages timeouts. The state engine is just about 10% of the source code and has not been modified for more than 18 months now.

FUTURE DEVELOPMENT

Tcl might not seem a perfect choice for implementing a complex system like the *Operation Master*. Especially the early versions contained operations like file-system manipulation (re-linking symbolic links) and process control of external programs. These were the main reasons not to use the standard EPICS sequencer – a very powerful tool to implement finite state machines in an extended C syntax.

As the state machine still undergoes further development, an easy to maintain formal description of states and transitions is mandatory. The GUI has already been factored out and also the file-system operations as well as process control and inter-process-communication have been replaced by more generic implementations.

Since all shortcomings have been removed from the *Operation Master*, the current Tcl-implementation is neither better nor worse than any other implementation.

CONCLUSION

The *Operation Master* minimizes errors due to inadvertences and avoids mistakes by taking the load of precisely following complex command sequences off the operator. It also implements standard mechanisms to recover from failure situations as long as no human interaction is necessary.

Experiences and the convincing success of the system are very encouraging to use the same system to develop new core control system components for other currently running as well as upcoming projects at BESSY/Helmholtz-Zentrum Berlin.

REFERENCES

[1] R. Klein et al., "Operation of the Metrology Light Source as a primary radiation source standard", Phys. Rev. ST Accel. Beams 11, 110701-1 (2008)s