# THE RHIC SEQUENCER*

J. van Zeijts, T. D'Ottavio, B. Frak, and R. Michnoff,
Brookhaven National Laboratory, Upton, NY 11973, USA

*Abstract*

The Relativistic Heavy Ion Collider (RHIC) has a high level asynchronous time-line driven by a controlling program called the 'Sequencer'. Most high-level magnet and beam related issues are orchestrated by this system. The system also plays an important task in coordinated data acquisition and saving. We present the program, operator interface, operational impact and experience.

## 1 INTRODUCTION

In the year 2000 commissioning run for the RHIC, the machine was cycled through its modes using an ad-hoc program, called the 'Sequencer'. This program, written as a rapid prototype, did its job but was not easily user-modifiable, and lacked sufficient progress feedback to the user. Clearly there was a need to provide a more mature tool. In this article we report on this tool, implemented for the year 2001 run, used to control the sequence of events required to make the RHIC ramp, acquire data, switch data acquisition modes, and generally perform operations in a repeatable manner.

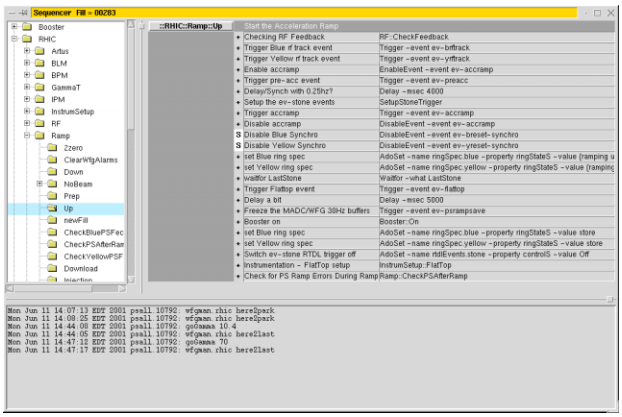


Figure 1: Sequencer Operational Interface.

## 2 REQUIREMENTS

The basic requirements are listed below:

- Provide a tool to allow reproducible, and reliable operations of the RHIC systems.
- Control the RHIC power-supplies modes as they cycle from 'Park' to 'Injection', and are ramped to the storage energy.
- Set up the various instrumentation systems to their appropriate modes along the machine cycle.
- Allow a simple mechanism for the user to skip a step in a sequence, and after an error, be able to restart the remainder of a sequence.
- Have a simple in input language with a placeholder for comments.

We will address these issues in the sections below.

## 3 SEQUENCES NAMING AND SYNTAX

Sequence names follow a naming convention using, where possible, the hierarchy: 'Machine System SubSystem Procedure', where Machine examples are: RHIC, AGS, Booster, Linac, or Tandem. System examples include the RHIC systems like: PowerSupplies, RF, Ramp, WCM (Wall Current Monitor), IPM (Ionization Profile Monitor), Permit, Abort, BLM (Beam Loss Monitors), and Artus (Tune Measurement). Subsystem are specific per system and can include: Schottky, PLL (Phase Lock Loop), or Blue, Yellow for the specific ring.

The Sequencer input language is loosely based on the LEP Sequencer language, each line starting with a key signifying normal run (R), skipping a line (S) , or break after this line (B). More keywords can be added if required. Following the keyword is a comment, and finally the code to be executed for that line. Build in primitives allow access to the multiple control systems, or to the many servers used to run the machine. Sub-sequences can be called by name and arguments passed in a standard way. An example of a sequence is given in the figure on the next page.

Apart from scripts written in the sequencer language, standard shell scripts, or any executable programs, are recognized and callable as a sub-sequence.

## 4 IMPLEMENTATION

The Sequencer is implemented in about 500 lines of Tcl/Tk code. This includes script parsing, GUI code, and run-time support. Each script hierarchy is living in its own name space, with multiple levels separated by '::', making full use of the Tcl naming resolution. Scripts can be modified on the spot, and modified code is generated dynamically.

Most sequences are executed in-line, giving speed comparable with compiled applications. Shell scripts, and executable programs are executed as sub-processes.

* Work supported by U.S. DOE under contract No. DE-AC02-98CH10886.

```
sequence "Ramping Preparation" Prep () {
        R "Set slow factor" AdoSet -name wfgman.rhic -property slowFactorS -value 2
        R "Download Ramp to WFG's" Ramp::Download
        R "Init RF Radial Steering" RF::InitRadial
        R "Start WFG's & MADC's 240 second buffers" Trigger -event ev-psrampsave-off
        R "Booster Off" Booster::Off
}
sequence "Start the Acceleration Ramp" Up () {
        R "Instrumentation - AccRamp setup" InstrumSetup::AccRamp
        R "Checking RF Feedback" RF::CheckFeedback
        R "Trigger Blue rf track event" Trigger -event ev-brftrack
        R "Trigger Yellow rf track event" Trigger -event ev-yrftrack
        R "Enable accramp" EnableEvent -event ev-accramp
        R "Trigger pre-acc event" Trigger -event ev-preacc
        R "Delay/Synch with 0.25hz?" Delay -msec 4000
        R "Setup the ev-stone events" SetupStoneTrigger
        R "Trigger accramp" Trigger -event ev-accramp
        R "Disable accramp" DisableEvent -event ev-accramp
        S "Disable Blue Synchro" DisableEvent -event ev-breset-synchro
        S "Disable Yellow Synchro" DisableEvent -event ev-yreset-synchro
        R "set Blue ring spec" AdoSet -name ringSpec.blue -property ringStateS -value "ramping up"
        R "set Yellow ring spec" AdoSet -name ringSpec.yellow -property ringStateS -value "ramping up"
        R "waitfor LastStone" Waitfor -what LastStone
        R "Trigger Flattop event" Trigger -event ev-flattop
        R "Delay a bit" Delay -msec 5000
        R "Freeze the MADC/WFG 30Hz buffers" Trigger -event ev-psrampsave
        R "Booster on" Booster::On
        R "set Blue ring spec" AdoSet -name ringSpec.blue -property ringStateS -value store
        R "set Yellow ring spec" AdoSet -name ringSpec.yellow -property ringStateS -value store
        R "Switch ev-stone trigger off" AdoSet -name rtdlEvents.stone -property controlS -value Off
        R "Instrumentation - FlatTop setup" InstrumSetup::FlatTop
        R "Check for PS Ramp Errors During Ramp" Ramp::CheckPSAfterRamp
}
sequence "Instrum AccRamp Setup" AccRamp () {
        R "SnapshotEv - PeriodicStartOnAccRamp" SnapshotEvent::PeriodicStartOnAccRamp
        S "SnapshotEv - 4Sec" SnapshotEvent::4Sec
        S "SnapshotEv - Stepstone" SnapshotEvent::Stepstone
        S "SnapshotEv - Transition" SnapshotEvent::Transition
        R "Disable WCM presnapshot script" DisableEvent -event scriptTrigger.16
        R "Disable WCM snapshot script" DisableEvent -event scriptTrigger.17
        R "WCM - 5 min log" WCM::Logging -state on -seconds 300
        S "WCM - FillPatt" WCM::FillPatt
        R "WCM - Transition" WCM::Transition
        R "Artus - SnapshotStoreAll" Artus::SnapshotStoreAll
        S "Artus - StepstoneStoreAll" Artus::StepstoneStoreAll
        S "Artus - 4SecStoreAll" Artus::4SecStoreAll
        R "IPM - HVPS Auto Disable" IPM::HVPSAutoDisable
        R "IPM - HVPS On" IPM::HVPSOn
        R "IPM - SnapshotStoreAll" IPM::SnapshotStoreAll
        S "IPM - StepstoneStoreAll" IPM::StepstoneStoreAll
        S "IPM - TransitionStoreAll" IPM::TransitionStoreAll
        S "IPM - 4SecStoreAll" IPM::4SecStoreAll
        R "BPMavgOrb - SnapshotStoreAll" BPM::aveOrbit::SnapshotStoreAll
        S "BPMavgOrb - StepstoneStoreAll" BPM::aveOrbit::StepstoneStoreAll
        R "BLM - SnapshotStoreAll" BLM::SnapshotStoreAll
        S "BLM - TransitionStoreAll" BLM::TransitionStoreAll
        S "BLM - ResetSum" AdoSet -name blmRing.all -property resetSumA -value 0
        R "BLM - ResetSum1 at Transition" BLM::ResetSum1AtTransition
}
```

Figure 2: Sequencer Input Language

**Control System Access:** Access to the multiple control systems used for Booster, AGS, and RHIC is through CDEV services [1, 2], providing a consistent interface.

**System Server Access:** Many servers are used in the running of RHIC [3, 4]. Servers are used for the managements of power supplies, administer ramps, etc. Access to these servers is through the CDEV API.

**Logging and Error Reporting:** When running a sequence, progress and errors are reported by a running indicator in the table column. In addition sequence and subsequence enter actions are listed in the cmlog logging system [5]. On failure of executing a line, the sequence will stop, and the GUI will indicate where the error occurred. The error is also reported to the logging system. At this point the operator has to make a decision to skip the problematic line, or wait and fix the problem. Since reproducibility will suffer if lines are skipped, the preferred scenario is to solve the underlying problem before proceeding.

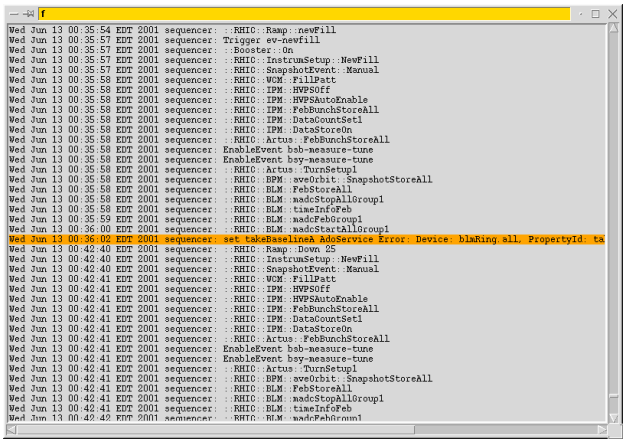

Figure 3: Sequencer Logging and Reporting Interface.

**Extensions:** In order to keep the main program responsive, time consuming tasks are handled by several subsystems. This includes for example checking the state of all power supplies, or checking all WFG's (Wave Form Generators) for errors. The 'SequencerServer' is a CDEV server process that is callable from the Sequencer, and reports its progress through call backs.

## 5 SUMMARY

The system is in continuous use in the RHIC control room and has been instrumental in providing reproducibility in the operation of the machine. Critical systems are setup in a standard way, and standard data is stored for each ramp. The simplicity of the sequence hierarchy and the straightforward language make it easy to maintain and improve.

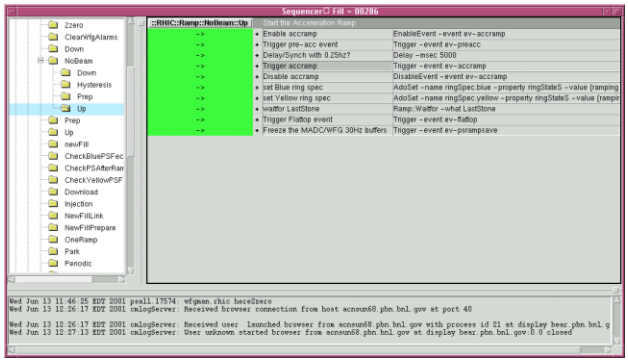

Figure 4: Successfully completed Ramp::NoBeam::Up sequence.

## 6 ACKNOWLEDGMENTS

The authors are thankful to the operations, controls, and accelerator physics staff for giving feedback on the program design and implementation. We also acknowledge the influence of the Fermilab and LEP sequencers on our tool.

## 7 REFERENCES

[1] J. Chen, et. al., "CDEV: An Object -Oriented Class Library for Developing Device Control Application", proceedings of ICALEPCS 1995, Chicago.

[2] T. D'Ottavio, "Experience layering CDEV on top of the AGS and RHIC Control systems", proceedings of SOSH98, Villigen.

[3] W. Akers, "An Object-Oriented Framework for Client/Server Applications", proceedings of ICALEPCS 1997, Beijing.

[4] J. van Zeijts, "CDEV Generic Servers for RHIC Commissioning and Operations", proceedings of ICALEPCS 1999, Trieste.

[5] J. Chen et al., "CMLOG: A Common Message Logging System", proceedings of ICALEPCS 1997, Beijing.