ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

MANAGE THE PHYSICS SETTINGS ON THE MODERN ACCELERATOR*

T. Zhang†, K. Fukushima, T. Maruta, P. Ostroumov, A. Plastun, Q. Zhao
Facility for Rare Isotope Beams, Michigan State University, East Lansing, USA

Abstract

The Facility for Rare Isotope Beams (FRIB) at Michigan State University is a unique modern user facility composed of a large-scale superconducting linac capable of accelerating heavy-ion beams from oxygen to uranium. An advanced EPICS-based control system is being used to operate this complex facility. High-level physics applications (HLAs) developed before and during the staged commissioning of the linac are one set of the critical tools that resulted in achieving the commissioning goals quickly, within several shifts. Many of these HLAs are expandable to other EPCIS controlled accelerators. Recently developed HLAs deal with the management of extensive data to achieve the repetitive high performance of ion beams in the entire linac measured by non-destructive diagnostics instruments, and open the possibilities to explore the extra values out of the data. This paper presents our recent significant development and utilization of these HLAs.

INTRODUCTION

The LINAC of the Facility for Rare Isotope Beams project at Michigan State University is a unique modern superconducting accelerator, it is designed to deliver all the stable isotope beams with multiple charge states to the kinetic energy higher than 200 MeV per nucleon, and power of 400 kW on the target [1]. Since the mid of 2017, staged commissioning has achieved a series of remarkable successes [2–4]. Now FRIB project is approaching the next milestone, that is to generate and separate the rare isotope beams at and after the production target system.

To expedite the beam commissioning, various kinds of high-level physics applications (HLAs) have been developed and deployed to FRIB controls network. The Python-based software framework called `phantasy` is designed and developed to drive the entire high-level communication between the accelerator and physicists [5]. Based on `phantasy`, the Python interactive scripting environment is implemented and utilized to prototype the physics tuning algorithms, and also used to control the machine in the advanced expert mode. Graphical user interface (GUI) applications have been developed with well-tested physics algorithms to make machine tuning simple and confident. General tools and GUI widgets based on Qt framework [6], together with physics-related widgets have been developed as another major part of `phantasy` framework to streamline the GUI application development.

* Work supported by the U.S. Department of Energy Office of Science under Cooperative Agreement DE-SC0000661, the State of Michigan and Michigan State University.
† zhangt@frib.msu.edu

To better organize the data generated in the controls network, software applications have been developed to ensure the quality of data, e.g. integrity, reliability, and availability, etc. The next coming sections will present the software development activity of FRIB high-level physics controls, the use cases of the development on the accelerator of FRIB, and the approach of how the physics data is managed.

THE EVOLUTION OF `PHANTASY`

Originally, `phantasy` was designed for the object orient high-level controls for EPICS-based accelerator system. The main goal was to abstract the entire machine to be controllable in the ecosystem of Python, by using the enormous third-party library to fulfill the machine tuning missions [7].

Several critical issues need to be addressed with neat solutions before accomplishing such high-level conrtols environment:

- Properly present the machine in the view of computing environment.
- Properly handle the value of physics quality in different scenarios, either from the view of device control or physics model.
- Properly interface with the physics model, which is usually separately developed to simulate the accelerator behavior.
- Properly manage the development resources, i.e. source code, support data files, testing, deployment, etc.

With `phantasy`, the physics model-independent machine representation could be generated in the view of object-oriented. The machine is represented as a series of devices, each device is an instance of a general abstracted high-level element class, which is composed of controllable and non-controllable attributes. The non-controllable attributes usually present the static properties of the device and the controllable ones are connected to the process variables which are served through EPICS IOCs. The device control is fulfilled via Python object getter and setter operations, through dotted-syntax. All the device information is maintained separately, such a way the core code of `phantasy` could also be working with other accelerator systems, the same strategy has been applied to the GUI application development.

The interpretation of the values for physics qualities is handled in another separated application called UNICORN [8], which is a web application that features REST API. The Python interface "python-unicorn" [9] is also developed to request either value in physics or engineering unit. The mapping rules between physics and engineering world could be defined based on the needs of the physics model, e.g. for

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

the dipole, the current in Ampere could be interpreted to Tesla as the magnetic field strength, or Tesla-Meter as the integral strength, thus two rules could be defined in UNICORN, use `.B` to read the device setting in Tesla, and `.B = <new-value>` to set with new magnetic field in Tesla, it is also possible to read it into Tesla-Meter via `.TM` if it is defined in the device configurations.

Speaking to the model interface, `phantasy` now supports modeling the accelerator with `FLAME` [10]. The internal virtual accelerator component is also developed with such a linear fast executing model. On top of the abstracted device control layer, any physics tuning algorithms could be prototyped quickly in any Python terminal environment, Jupyter [11] is used as the main one. `phantasy` is also designed with the ability to adapt with other physics models, to do that, the routine of model-dependent machine representation needs to be developed.

All the software development is managed with modern tools. `GIT` [12] is used to manage the source code, the whole physics application development is managed with the internal Bitbucket service [13]. More than 20 Debian packages are built to deploy the development into FRIB controls network via the automated CI/CD infrastructure [14]. Accumulated code changes will be released as new versions of software and deployed into production through the automated pipeline.

Recently, `phantasy` has been enhanced a lot on the GUI application development side. New GUI widgets have been developed for PyQt application development. To unify the UI style and make use of existing resources, command-line tools also have been developed and improved. One can initiate the app skeleton by "makeBasePyQtApp" command. It is the plan to release all the development through PyPI [15] (some of the packages have already been published there) and PPA [16] in the near future, along with the documentation sites, which require a lot of effort.

MANAGE THE DATA IN THE CONTROLS NETWORK

From the view of the high-level physics controls, there are different kinds of data distributed and generated in the controls network.

The first category is device information, e.g. the name, size, location, controls process variables, etc., all these information is already managed in `phantasy` framework, with above-mentioned dotted-syntax, all these attributes could be reached naturally in Python environment, as well as IOC application.

The second category is the data from other web services. For instance, the data from directory service ChannelFinder [17], the data from alarm services, the data from Archiver Appliance [18], the data from Olog [19], and the data from UNICORN service, etc.

The third category is the data from various software IOCs. When building the high-level physics controls system, software IOC applications are developed to fulfill the specific data processing tasks, the results are either integrated into `phantasy` or exposed as PVs.

The fourth category is the data from physics models, which could be lattice files, simulation results.

Last but not least one is the developed software itself.

On the other side, in the view of machine tuning, all these data should be well-organized to achieve the following goals:

- The developed machine tune should be accurately saved.
- The saved machine tune should be accurately restored.
- There must be some way to verify the restored machine tune.
- There must be some way to easily apply one machine tune to different charge states and ion species.

To address them, systematic software development is undergoing.

Physics Settings

Here the data of physics settings is defined as a series of device set values along the accelerator beamline. The app named "Settings Manager" has been developed to manage the physics settings. "Settings Manager" is one of the GUI apps that is built upon `phantasy` framework, it can work with different machine/segment, which is defined in "phantasy-machines" (a package that keeps all the configurations of machine/segment for `phantasy`).

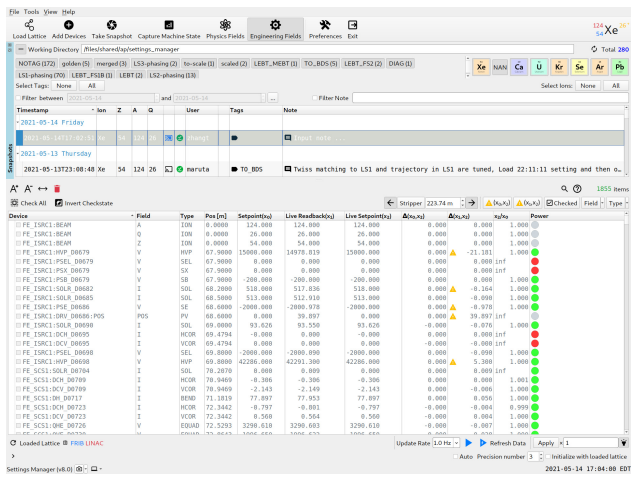

Figure 1: Main user interface of "Settings Manager".

Figure 1 shows the main user interface of "Settings Manager". Buttons in the toolbar indicate the features it provides, as well as the current working ion species shown at the rightmost. Right below the toolbar is the area for saved physics settings or snapshots. All the snapshots are sorted by the saved time by default, all come with the ion species info, saved user name, tags, and note. "Tags" is a string of words separated by a comma, to label the snapshot data, and "Note" is a string for arbitrary additional info about this snapshot.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

The expanded snapshot area is showing in Figure 2, it features tabular view of all the snapshots, supports sorting and filtering, and convenient buttons for filtering by ion names or tags, to quickly locate the snapshot entries.

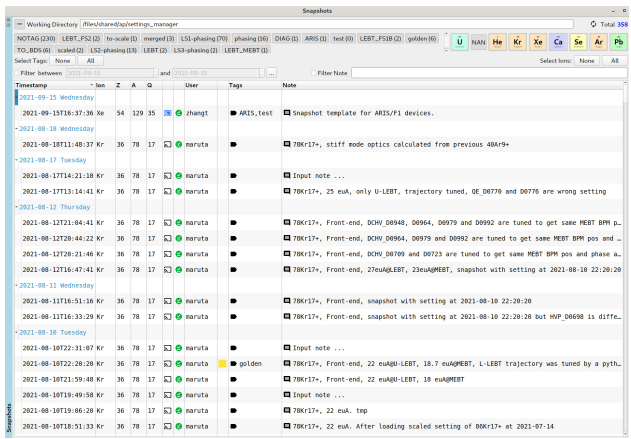

Figure 2: Snapshot management window of "Settings Manager".

The snapshots could be loaded and presented in the right bottom area by double-clicking or right-clicking context menu actions. In Figure 1, one can see the listed device settings, each row is for one "Field", each "Field" is a control knob, the device info could be reached via the context menu on the "Device" column. The saved device settings are shown in column "Setpoint(x_0)", while current settings are in column "Live Setpoint(x_2)", column "$\Delta(x_0, x_2)$" indicates how they are matched or not.

The procedure for applying saved settings to machine is, first check the fields that need to be applied (check by clicking, keyboard and context menu actions), then press "Apply" button to do process one by one, the details of device settings could be controlled via the "Preferences" menu. One feature to support multiple charge states settings is applying with a scaling factor, which is right after "Apply" button, the value will be automatically calculated, but also could be modified, default is 1.0. After applying all the settings, recheck column "$\Delta(x_0, x_2)$" to make sure the current settings are updated accurately, for the case of applying with a scaling factor, check column "x_2/x_0", which is expected to be the value of the scaling factor.

The data model of the snapshot supports exporting to different kinds of data files for portability, typical data formats are XLSX and HDF5, from which one can have the metadata and physics settings data, as well as the machine state data.

Machine state data is to address how to verify the machine performance under one specific physics settings. The definition of it is the readings from various diagnostics devices when taking the snapshot. The configuration file for machine state data capturing is a predefined file shown in Figure 3, could work with CLI tool `fetch_mach_state` (which is deployed with `phantasy-apps` package) as well. When taking a snapshot, the machine state data is first captured based on the configuration file, then capture current

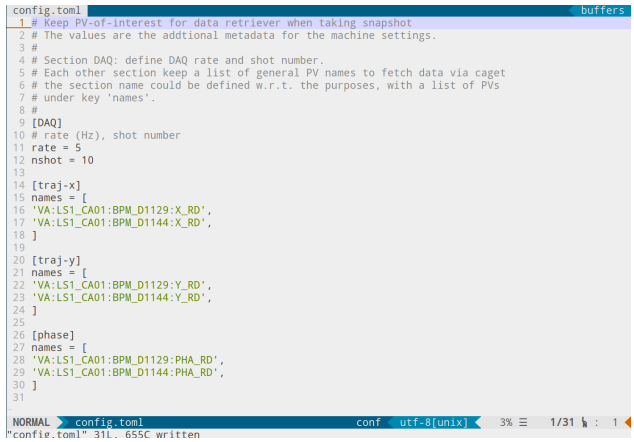

Figure 3: An example of the configuration file for machine state fetching.

physics settings, and save them together into one snapshot, finally shown as one entry in the snapshot window. "Settings Manager" supports managing snapshots in a database or a directory. The machine state data also could be captured at any time via the tool in the toolbar. The obvious benefit of saving physics settings with machine state is that all the data actually been accurately "labeled", which is very valuable for the future machine learning application.

Figure 4 visualizes the trajectory along the FRIB LINAC from the machine state data, one can also visualize other qualities, e.g. BPM phase, amplitude, to verify the physics settings.

It is also worth mentioning that the physics settings are also pre-generated from physics models before the beam commissioning and imported into "Settings Manager" for use, which significantly speeds the beam tuning efficiency.

ARCHIVED DATA

Another essential data source for the physics application software is the data archiver. FRIB is using the site-maintained Archiver Appliance application [18] to keep archived PV values. The archiving and data retention policies are defined in each IOC. The time-series data served by Archive Appliance could be visualized in CS-Studio data browser application.

To integrate the archived data into the Python ecosystem, a dedicated Python client package has been developed, called "pyarchappl" [20], which is not only designed for communicating with Archiver Appliance via REST API, but features with data-wrangling capabilities, to work with the archived data with popular data science tools.

One of the useful command-line tools distributed with `pyarchappl` is `pyarchappl-get`, which is a general command for data retrieval from Archiver Appliance. Figure 5 shows the user guide of the command. One can easily get timestamp aligned archived PV data in the form of CSV, XLSX and HDF5.

The retrieved data set could be processed by reading the data file. Figure 6 is the heat map view for the x and y

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

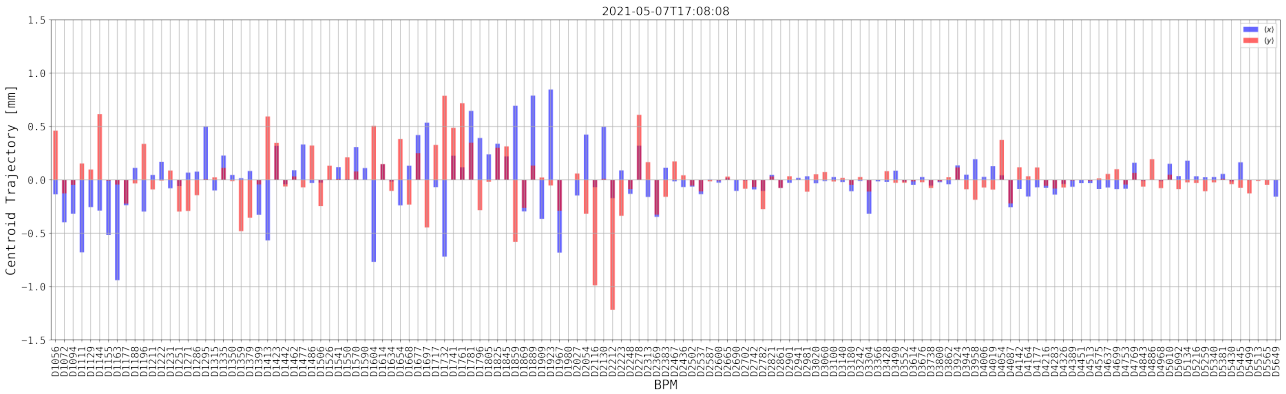

Figure 4: Typical example of trajectory visualization with BPM data

```
tong ~ > pyarchappl-get -h
usage: pyarchappl-get [-h] [--url URL] [--pv PV_LIST] [--pv-file PV_FILE]
                      [--from FROM_TIME] [--to TO_TIME] [--resample RESAMPLE]
                      [--verbose] [-o OUTPUT] [-f FMT]
                      [--format-args FMT_ARGS]

Retrieve data from Archiver Appliance and export as a file.

optional arguments:
  -h, --help            show this help message and exit
  --url URL             URL of Archiver Appliance, default is FRIB FTC
                        archiver (default: None)
  --pv PV_LIST          List of PVs for retrieval, each define with --pv
                        (default: None)
  --pv-file PV_FILE     A file for PVs, one PV per line (skip line starts with
                        #), append each to pv_list (default: None)
  --from FROM_TIME      A string of begin time in ISO8601 format (default:
                        None)
  --to TO_TIME          A string of end time in ISO8601 format (default: None)
  --resample RESAMPLE   The offset string/object representing target
                        conversion, e.g. '1S' for resample with 1 second
                        (default: None)
  --verbose, -v         Verbosity level of the log output, 0: no output,
                        1(-v): output progress, 2(-vv): output progress with
                        description (default: 0)
  -o OUTPUT, --output OUTPUT
                        File path for output data, print to stdout if not
                        defined (default: None)
  -f FMT, --output-format FMT
                        File format for output data, supported: csv, hdf,
                        excel, html, ... (default: csv)
  --format-args FMT_ARGS
                        Additional arguments passed to data export function in
                        the form of dict, e.g. '{"key":"data"}' (for hdf
                        format) (default: {})

Examples:
# Retrieve raw PV data in the defined time frame
$ pyarchappl-get -o data.csv -v \
  --pv LS1_CA01:BPM_D1129:XPOS_RD --pv LS1_CA01:BPM_D1129:YPOS_RD  \
  --from 2021-04-15T20:10:00.000Z --to 2021-04-15T21:25:00.000Z \

# Align the timestamps, resample at 1 second
$ pyarchappl-get -o data.csv -v \
  --pv LS1_CA01:BPM_D1129:XPOS_RD --pv LS1_CA01:BPM_D1129:YPOS_RD  \
  --from 2021-04-15T20:10:00.000Z --to 2021-04-15T21:25:00.000Z \
  --resample 1S
```

Figure 5: Help message of `pyarchappl-get`.

readings of the 75 BPMs along the accelerator through first LINAC segment (LS1) to the first fold segment (FS1), in the time range from 16:00 to 20:00 EDT on Oct. 13, 2020, total 72000 data entries. The blue boxed areas are the minimal absolute trajectory readings, which is within the range of ±0.5 mm after trajectory correction had been performed.

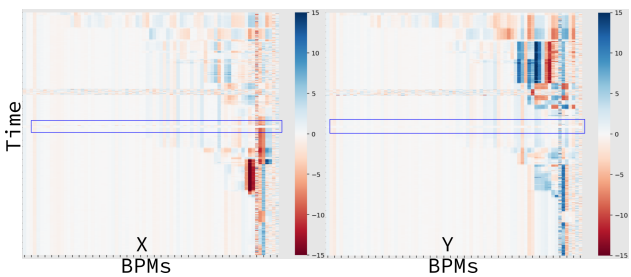

Figure 6: Heat map plot of time-series archived data for central trajectory in x and y.

Tools also have been developed to retrieve the physics settings at the time when the trajectory reached the minimal, then the retrieved data is processed to be a new snapshot for "Settings Manager". All these tools will be integrated into "Settings Manager" as new context menu actions.

CONCLUSION

The high-level physics controls software framework `phantasy` is continuously moving forward. The enhanced toolkits for PyQt GUI development greatly streamlined the HLA development. The solid software solution has been designed and implemented into the data management of FRIB LINAC commissioning. "Settings Manager" has been developed and used to accurately manage the physics tunes for the entire LINAC, as well as keep the machine state dataset to make the data of physics settings more valuable for other applications. "Settings Manager" expedites the LINAC tuning for multi-charge state operating mode. The Python client package "pyarchappl" for Archiver Appliance has been designed and developed for easy data retrieval and general data analysis with Python third-party packages. The data management solution is continuously consolidating to prepare for the concrete machine learning applications.

ACKNOWLEDGMENTS

The authors would like to thank A. Carriveau and T. Ashwarya for the useful discussions.

REFERENCES

[1] J. Wei *et al.*, "Advances of the FRIB project", Int. J. Mod. Phys. E 28, 1930003 (2019). doi:10.1142/S0218301319300030

[2] P. Ostroumov *et al.*, "Heavy ion beam acceleration in the first three cryomodules at the Facility for Rare Isotope Beams at Michigan State University", Phys. Rev. Accel. Beams. 22, 040101 (2019). doi:10.1103/PhysRevAccelBeams.22.040101

[3] P. Ostroumov *et al.*, "Beam commissioning in the first superconducting segment of the Facility for Rare Isotope Beams", Phys. Rev. Accel. Beams. 22, 080101 (2019). doi:10.1103/PhysRevAccelBeams.22.080101

[4] P. Ostroumov *et al.*, "First Simultaneous Acceleration of Multiple Charge States of Heavy Ion Beams in a Large-Scale Superconducting Linear Accelerator", Phys. Rev. Lett. 126,

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEBL04

114801 (2021).
doi:10.1103//PhysRevLett.126.114801

[5] T. Zhang *et al.*, "High-level Physics Controls Applications Development for FRIB", in *Proc. 17th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS 2019)*, New York, NY, USA, Oct. 2019, pp. 828–834. doi:10.18429/JACoW-ICALEPCS2019-TUCPR07

[6] Qt GUI framework, https://www.qt.io

[7] T. Zhang, "Physics high-level applications and toolkit for accelerator system", in *EPICS Collaboration Meeting*, Argonne National Laboratory, IL, USA, Jun. 2018. https://epics.anl.gov/meetings/2018-06/talks.html

[8] Web application for physics quantity interpretation, https://github.com/phantasy-project/unicorn-webapp

[9] Python interface to UNICORN, https://github.com/phantasy-project/unicorn

[10] Z. He *et al.*, "The fast linear accelerator modeling engine for FRIB online model service", Computer Physics Communications 234, 167 - 168 (2019). doi:10.1016/j.cpc.2018.07.013

[11] Jupyter, https://jupyter.org

[12] Git, https://git-scm.com

[13] Bitbucket, https://bitbucket.org

[14] M. Konrad *et al.*, "Automatic deployment in a control system environment", in *Proc. 17th Int. Conf. on Acc. and Large Exp. Physics Control Systems (ICALEPCS 2019)*, New York, NY, USA, Oct. 2019, pp. 126–131. doi:10.18429/JACoW-ICALEPCS2019-MOMPL006

[15] The Python Package Index, https://pypi.org

[16] Personal Package Archive, https://launchpad.net/ubuntu/+ppas

[17] ChannelFinder, http://channelfinder.github.io

[18] Archiver Appliance, https://slacmshankar.github.io/epicsarchiver_docs

[19] Online logbook service, https://github.com/Olog

[20] Python client to Archiver Appliance, https://github.com/archman/pyarchappl