A FLIGHT SIMULATOR FOR ATF2 - A MECHANISM FOR INTERNATIONAL COLLABORATION IN THE WRITING AND DEPLOYMENT OF ONLINE BEAM DYNAMICS ALGORITHMS*†

Glen White, LAL, Univ Paris-Sud, CNRS/IN2P3, Orsay, France & SLAC, Menlo Park, California, Stephen Molloy, Andrei Seryi, SLAC, Menlo Park, California, Daniel Schulte, Rogelio Tomas, CERN, Geneva, Shigeru Kuroda, KEK, Ibaraki, Philip Bambade, Yves Renier, LAL, Univ Paris-Sud, CNRS/IN2P3, Orsay, France

Abstract

The goals of ATF2 are to test a novel compact final focus optics design with local chromaticity correction intended for use in future linear colliders. The newly designed extraction line and final focus system will be used to produce a 37nm vertical waist from an extracted beam from the ATF ring of ~30nm vertical normalised emittance, and to stabilise it at the IP-waist to the ~2nm level. Static and dynamic tolerances on all accelerator components are very tight; the achievement of the ATF2 goals is reliant on the application of multiple high-level beam dynamics control algorithms to align and tune the electron beam in the extraction line and final focus system. Much algorithmic development work has been done in Japan and by colleagues in collaborating nations in North America and Europe. We describe here development work towards realising a 'flight simulator' environment for the shared development and implementation of beam dynamics code. This software exists as a 'middle-layer' between the lower-level control systems (EPICS and V-SYSTEM) and the multiple higher-level beam dynamics modeling tools in use by the three regions (SAD, Lucretia, PLACET, MAD...).

OVERVIEW OF THE FLIGHT SIMULATOR PROJECT

The main goals of the flight simulator project can be summarised as follows:

* Provide a simple to use, beam dynamics oriented, portable control access framework for ATF2 tuning tasks.

* There should be simple and reversible transition from beam dynamics simulation to accelerator-ready code.

* Provide the ability for international collaborators to develop beam tuning tools without the need for expert-level knowledge of control systems.

* Allow the flight simulator to operate in simulation mode at an external location in the same way as the production system deployed at ATF2.

Another key difference between the interface offered by the flight simulator to that of the standard control system is that it provides a mechanism to automate multiple beam tuning tasks.

Through a common interface, it is left to the user to decide which accelerator code best suits his or her needs. The flight simulator software thus forms a management package which allows beam dynamics code written in a variety of environments to interact.

The main concentration of effort for this project is to address the tuning tasks of the ATF2 extraction line and final focus system. However, the extension of the project to include beam physics tasks in the Damping Ring is also envisioned for the future.

COMPONENTS OF THE FLIGHT SIMULATOR

The core functionality of the flight simulator software is written as a library of Matlab functions which extends the functionality of the Lucretia accelerator toolbox [1]. Lucretia contains many functions useful for the simulation of single-pass electron beamlines and has been used extensively in the modelling of ILC. It was written to provide the user with an interface very much like one would use at an operational accelerator installation. As such it lends itself well to being used in the fashion described here.

The flight simulator can be described as a 'middle-layer' between the lower-level control system interface and the beam dynamics simulations and algorithmic development performed by physicists. By providing a portable system that behaves in the same way in 'simulation mode' as the production version at the accelerator, it is possible to develop code in an environment that the user is used to and can be run on the real machine with little or no alteration. It is also possible to develop a final version of the beam dynamics code without direct access to the production control system which has security benefits.

The pre-existing Lucretia data structures are used to hold the real-time data. The design model values are kept in the BEAMLINE array, and real-time modifications in the PS (power supply), GIRDER (movers) and INSTR (BPMs and other monitors) arrays.

A schematic representation of the flight simulator and its installation into the ATF control system environment is shown in figure 1. At the lowest level of the system are the Input-Output Controllers (IOCs) that read and write data to the beamline hardware systems. For many of the new hardware systems these are EPICS-based IOCs running on a mixture of VME, CAMAC and PC-based architectures. The re-used and existing IOC systems from ATF are based on the V-SYSTEM control architecture. An additional 'soft' EPICS IOC has been written to generate EPICS Process Variables (PV's) that synchronise to the control variables in V-SYSTEM. The flight simulator

*work supported in part by Department of Energy Contract DE-AC02-76SF00515
†work supported in part by the Commission of the European Communities , contract number RIDS-011899

implements its hardware read-write functions through an EPICS Channel-Access (CA) client interface.

The flight simulator software itself exists as two installations. The "trusted" installation has read/write access to EPICS and runs applications that have been well tested and as such are integrated into the control system proper. The other installation is the "test" installation. This typically runs on a users personal computer. A server-client relationship thus exists between the trusted and test installations. The user writes applications in the test environment, and to gain access to the control system PV's that it needs to run, it must request access rights from the server which then handles read/write requests.

"Trusted" Installation Functionality

The core functions of the flight simulator server software are listed here:

Server-client communications: A tcp socket implemented in java natively in the Matlab environment is used for communications between the "trusted" server and "test" client. The server constantly listens out on a tcp socket and assigns new ports to inbound requests, thus allowing the possibility of multiple simultaneous connections.

Access control to client installations: By default the "test" client has no write access to any control variable.

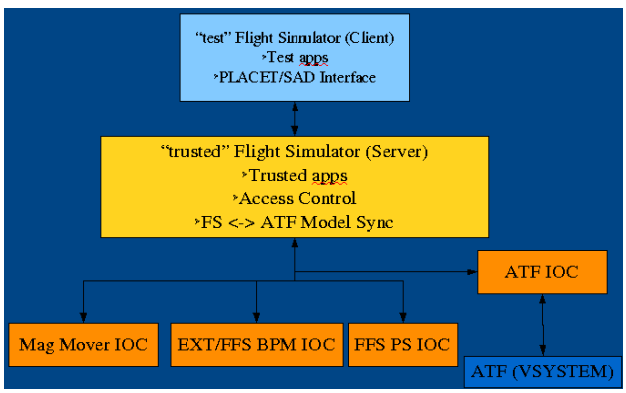

Figure 1: Implementation of the Flight Simulator at ATF

Using one of the built-in flight simulator functions, it must request access for all the PV write-access it needs to run a given application. Each new request from a client appears on a window on the computer running the server. An operator must then grant permission for the PV access. This way, full control is maintained over which systems are being used at any time avoiding problems with multiple simultaneous control requests and providing an additional layer of security to the control network.

Server-based data services: Where tested applications exist, they are available through the flight simulator interface to be called from the server. An example is BPM averaging- it is possible to call the averaging function from the client and the server will run the application and return the averaged readings, with optional quality cuts.

Sync. Control system with Lucretia model: The server updates it's internal Lucretia model of ATF2 at the machine rep. rate of 1.56 Hz. It has internal look-up tables to translate the raw control system values (magnet currents etc) into the values used by Lucretia.

The server connects to the EPICS databases through labCA [2]- an EPICS CA client written for Matlab.

Maintain updated AML and Lucretia models: The two model definition standards supported by the flight simulator are the native Lucretia model data (binary matlab .mat files) and AML. Both files are periodically written to disk, with the intention of making these available on a public server so a user can obtain the current state of the machine at any time from any location. It is also planned to implement an archive function, such that the running state of the machine at any given time can be obtained which should prove very useful for post-mortem diagnostics.

Provide PS setting and magnet move functionality through native Lucretia functions (PSTrim, MoverTrim): The way of making changes to the accelerator (namely, power supply changes and magnet mover settings) is through the usual native Lucretia functions, with additional complexity necessary to deal with access from the server or client hidden from the user.

"Test" Installation Functionality

The client flight simulator, installed on a user's laptop for example, is effectively the same software environment as the server. It contains a local version of the ATF2 lattice and settings in it's own Lucretia model. On startup, it initiates a link to the server through a tcp socket connection. It initially loads the default lattice, but then can synchronise it's Lucretia database with that of the server's automatically or at the users request.

There is a common directory structure between the 2 installations; test applications and certified trusted applications exist in separate folders. Both are available to the client, but only the trusted applications are available to the server which runs on the control system proper.

The client then has the full functionality of the Lucretia toolbox which can now be operated on with a live version of the ATF2 machine state. However, access to write to PS (power supply) and GIRDER (magnet mover) elements is restricted initially and access must be specifically granted by an operator through the server interface.

To enable the use of other accelerator codes within this same framework, the client also runs an additional tcp socket server. This is implemented in an identical way to the other flight simulator server socket. This client-side server processes ascii commands from a source on the localhost. In this way, another accelerator code has access to the full functionality of the flight simulator as long as it can implement a tcp socket client.

The flight simulator supports the AML deck standard [3] through the use of Lucretia2AML- code written using UAP (Universal Accelerator Parser) to synchronise the Lucretia model with an AML file. This serves as the entry-point for an external accelerator code package. Further updates and access/read/write requests are performed through the flight simulator client socket server process. Through the use of a shared data structure on the flight simulator client program, it is possible for sharing of data and application processing between Lucretia and other accelerator codes.

Security

Access security for the ATF2 hardware controls database is enforced in three ways:

All read/write access is channelled through the "trusted" flight simulator server installation on the local ATF2 control subnet. A client program that wishes to connect to the server must also be on the same subnet.

The EPICS IOC's are implemented using the standard EPICS access security protocols. These are set to only allow write commands (dbputs) from the user/host that the flight simulator is installed on.

The ATF2 network infrastructure- the flight simulator software and EPICS IOC's run on the ATF2 local subnet, which is firewalled from the main KEK intranet, itself firewalled from the internet.

SIMULATION MODE FUNCTIONALITY

To enable the offsite preparation and testing of applications, the flight simulator was written to be able to run in an environment away from the ATF2 control system (the personal computer of a user at their home institute for example). This simulation environment is kept as similar to the production environment as possible. The client software operates exactly as it does in production, as does the server software. However the default way of running the flight simulator in simulation mode is to run all required software on the same computer installation. Also, the server gains additional functionality in simulation mode. As it no longer is connected to the ATF2 control system, it simulates the action of the ATF2 beamline by utilising the Lucretia tracking software. The graphical interface for the server thus has additional inputs to define how the tracking of the beam is handled, allows for the setup of different simulated error parameters etc. To further minimise the difference between the production and simulation environments, when the server software is started, the EPICS IOC's are also started in the same environment. These are the same IOCs that run at ATF2, but a SIM_MODE record is activated which causes them to not try and access hardware but instead use a simulation layer written in the driver code which causes the IOC's to behave as if they were connected to hardware. This allows the flight simulator server to behave in a very similar fashion as in production mode where it still directs read/write requests through EPICS CA. An additional benefit to this is that new hardware can also be developed offsite, an EPICS database written to communicate with the new hardware, and tested in the simulated version of the control system environment. This is done with the knowledge that it will behave in the same way in the production version at KEK. This can greatly reduce the time needed to spend onsite at KEK testing new hardware and controls interfaces.

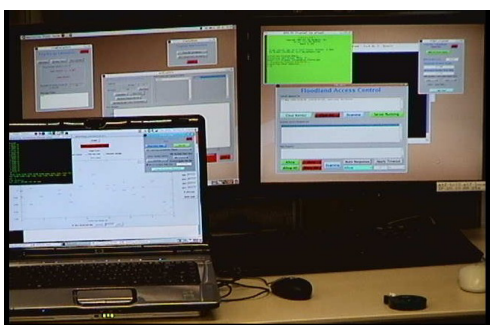

Figure 2: A test of the flight simulator at ATF- client running on user laptop and server running on ATF control system.

TESTING THE FLIGHT SIMULATOR SOFTWARE AT ATF

In May 2008, a test of the flight simulator software and an example client application was performed (integrated with the ATF control system).

The 'soft' EPICS IOC that handles communication with the ATF V-SYSTEM controls was installed on a pc on the ATF controls subnet. The "trusted" flight simulator installation (server) was compiled on a control room pc. Thus, access to the ATF magnet power supplies and BPM readout system was possible through the flight simulator and an EPICS IOC as envisioned for ATF2.

A test application was developed using the PLACET [4] accelerator code which was run through the flight simulator client server interface as described above. The test application was to experimentally determine transfer matrix elements between corrector magnets and BPMs in the ATF extraction line and use these to perform automated orbit steering and provide a bump calculation and application tool. This application was tested offsite at the users home institute using the simulation mode, then used at the production installation at ATF. The test was successful- figure 2 shows a picture of the test in progress with the client application running on the users laptop and the server code running on the ATF control system.

PLANS FOR ATF2

With the core flight simulator software written and tested, the next stage is to start to migrate existing beam tuning algorithms into this software infrastructure and to use them to commission and tune ATF2. The existing plans centre around the extraction line and final focus system- e.g. Orbit feedback systems; magnet mover based BBA and steering; IP tuning using orthogonal sextupole-mover knobs; extraction line coupling and dispersion correction etc. The project is being documented on the SLAC ATF2 wiki pages [5].

REFERENCES

[1] P. Tenenbaum, "Lucretia: A Matlab-Based Toolbox for the Modelling and Simulation of Single-Pass Electron Beam Transport Systems", PAC-2005-FPAT086, Sept 2005.

[2] Till Straumann, http://www.slac.stanford.edu/comp/unix/package/epics/extensions/labca/manual/

[3] D. Sagan et. al., "The Accelerator Markup Language and the Universal Accelerator Parser", EPAC-2006-WEPCH250, Aug 2006.

[4] D. Schulte et. al. "Recent improvements of PLACET", CERN-AB-2006-048, Jun 2006.

[5]https://confluence.slac.stanford.edu/display/ATF/ATF2