ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-THPRO107

APPLICATION PROGRAMS OF ELETTRA AND FERMI@ELETTRA

F. Iazzourene, C. Scafuri, Elettra Sincrotrone Trieste, Trieste, Italy

Abstract

At Elettra we have high level software application programs used on the ring and others, based on the TANGO control system, firstly developed and successfully used for the commissioning and operation of the Elettra booster and now for FERMI@Elettra transfer lines. The latter use a new High Level Framework and a new beam optics module. The paper describes the present status and some of the application programs.

INTRODUCTION

Elettra is a 2.5GeV 3rd generation light source in operation since October 1993. FERMI@Elettra, a 4th generation light source, is a linac based single pass free electron laser, presently in the commissioning phase. For the Elettra booster, in operation since 2007, a new High Level Framework (HLF) together with new beam optics utility routines and application programs have been developed. These are also been used for applications developed for the optics matching in FERMI@Elettra transfer lines.

ELETTRA RING HIGH LEVEL SOFTWARE

The original high level application programs are written in the computer language C. They use the graphical user interface based on the X11/Motif tool kit. Since 1993, many upgrades have been done and other applications have been developed.

Control System

The original Elettra control system is based on a three layer architecture [1,2]. It is still used in its original form for controlling most of the power supplies of the Elettra storage ring. The middle ware, that is the software interface to the control system, is provided by Remote Procedure Call (RPC) protocol, originally developed at CERN. The RPC provide, essentially, the methods to read and set control system variables, for example the current furnished by power supplies. The control system variables are specified by their uniquely defined names. Control system variables can be scalars or arrays.

Data Files and Utility Routines

The application programs use fixed data files and utility routines. There are four data files. A file which describes the ring lattice. A file which contains the access address of the ring components to the control system. A file which provides the range and speed variation of some variable parameters, like for example the power supplies intensities. And finally a calibration file which contains the coefficients obtained from a polynomial expansion of the measured magnetic field and gradient versus excitation and vice versa [3]. The utility routines allow transparent access to the control system, convert the data read from the control system to machine physics parameters and compute the optics.

Application Programs

The now mostly used application programs are OrbitFB[4], Bump[5] and Toca[6]. Bump and Toca are in use since 1994 and 1998, respectively. OrbitFB replaced SlowFB in 2002. There have been several upgrades. The migration of the control system platform from unix to linux[7]. The correction of the local orbit at the short straight section upstream the bending magnet photon source after a short planar undulator was added in the short straight section 12. In some sections , a combined horizontal/vertical corrector is added after the bending magnet photon source to allow the local correction of both position and slope at both the bending magnet photon source and the downstream undulator. Fig. 1 and 2 show Bump with the added corrector in section 6.

Figure 1: The selected bump and correctors at bending magnet photon source in section 6.

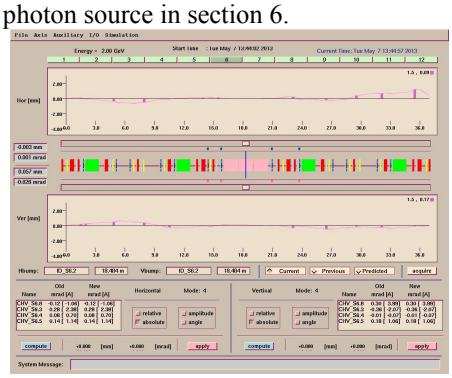

Figure 2: The selected bump and correctors at undulator photon source in section 6.

Bump creates a bump for a local orbit amplitude or/and a slop request at one point only. Furthermore, the user has to “compute”, then to “apply” and then “acquire” the new

ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-THPRO107

orbit. OrbitFB was and can be used as a slow local orbit feedback correction. OrbitFB allows bumps with up to 6 local constraints. In section 6 and 11, the code computes the bump which sets the position and slope at the user's values requests at the bending magnet photon source and the downstream undulator. In section 7, the bump includes also the position of the beam at the transverse feedback pick up position. In section 1, the bump sets the position and slope at the user's values requests at the bending magnet source and undulator and at the upstream short undulator of section 12. OrbitFB corrects all the selected sections together. Fig. 3 shows the panel with the read global and extracted local orbit.

Figure 3: OrbitFB panel with the global and local orbit.

OrbitFB stops the correction when the tolerances are or/and the number of iterations is reached. The control of the orbit and the subsequent correction if necessary repeats after a given time interval. The user can acquire the orbit only, button “Acquire only”, or start the loop of reading/correcting the orbit, button “Start loop”, or stop the loop, button “Stop”. The latter button is dramatically important if for some reason the process diverges. On top of the normal controls, for example check if the used correctors and bpms are functioning correctly, if the optic has a solution, etc, the application automatically stops the correction if the global rms orbit exceeds a chosen value. The code has a text window on which it prints the global and local orbit, the process steps, e.g. “I am acquiring”, “I am applying the correction”, the non satisfied constraints together with the corresponding tolerances.

All the needed requests and settings for the use of the code are in an input file. The user can change the data in the file or change them from the menu “ChangeDefault”. In the latter case the new values are no more available when the user exits from the application. For example, the user can change the selected constraints points from the “Correction Points” widget as shown in fig.4.

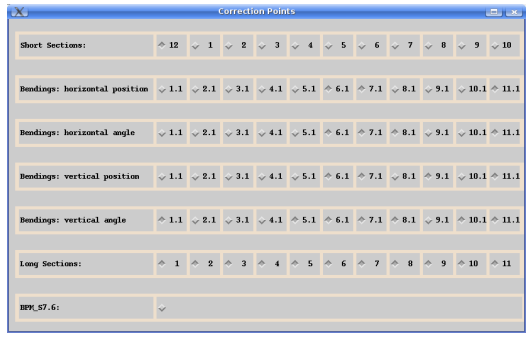

Figure 4: Correction points selection.

The user can also undo the corrections through the widget “Undo Correction”.

ELETTRA BOOSTER AND FERMI@ELETTRA HIGH LEVEL SOFTWARE

With the new control system based on Tango, a high level framework together with utility routines have been firstly developed for the new Elettra injector[8,9] and then extended to FERMI@Elettra transfer lines. The utility routines and the high level application programs are written in the computer language C++. The application programs use the graphical user interface based on the Qt tool.

Control System

The booster control system [10] is an extension of the Elettra control system; they share the same ethernet network, operator consoles and central servers. The booster control system architecture is based on a two layers design. The bottom layer is based on VME disk less systems, known as Equipment Controllers (EC), equipped with Motorola MVME5100 PowerPC boards running GNU/Linux. The bottom layer performs all the Input/Output to the booster equipment by means of different type of electronic cards hosted on the VME systems. Each EC is in charge of a set of functionally related equipment, such as power supplies, RF equipment, vacuum. The top layer is based on GNU/Linux workstations and servers. The booster control system middle ware is based on Tango [11]. Tango devices on the ECs perform all the local control of physical devices managed by that EC. Each physical device has been modelled and a Tango device server providing the functionalities required for that device has been developed and deployed in the ECs. A number of Tango device servers were already available from the Tango community, ready to be used or requiring minor changes. The structure of the FERMI@Elettra control system [12] is almost identical to the one of the booster. The differences do not affect this high level software.

ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-THPRO107

High Level Framework and Utility Routines

The High Level Framework (HLF) is a set of software libraries providing robust tools for the development of machine physics programs. The HLF design is fully object-oriented and follows a modular approach. Each module is dedicated to a well defined task needed by model based accelerator control programs: management of machine configuration and calibration, access to the real machine and unit conversions and optics calculations. HLF is written in C++. Management of machine configuration and calibration is based on a relational database that stores all the needed information: lattice description, magnet and other devices calibration tables, mapping between accelerator element models and control system devices. A number of utility classes are available for building models of the accelerator elements from the data stored in the database. Access to the real machine and units conversion is performed by a set of classes modelling accelerator elements. The most important objects are magnet devices (bending, quadrupole, etc...). The purpose of these objects is to provide the actual physical quantities relevant to optics calculations (e.g. gradient, bending angle, etc...) by reading the engineering quantities (e.g. magnet current) by means of the control system and transforming them via the appropriate calibration tables. Accelerator elements are assembled into lattices to model parts of or the whole accelerator.

Beam Optics Utility Routines

These routines perform mainly four tasks. One accesses the control system to get the lattice, read the physical objects, for example the excitation in a power supply, the position of the beam in a beam position monitor. The second controls if these data are valid. The third computes the optics. The last computes the matching of the optics and the correction of the orbit. For both matching and orbit correction, the code uses SVD method. For the orbit correction, Micado is also available. The mathematics used are from it++ package.

Application Programs

Many application programs have been developed for the new Elettra injector[9]. BtsTrajectory has been upgrated. Fig. 6 shows the horizontal trajectory before (red), predicted (green) and measured after correction (blue), using a measured response matrix.

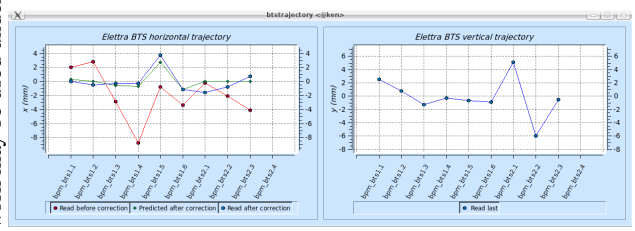

Figure 6: Horizontal trajectory before correction (red), predicted after correction (green), read after correction (blue).

For FERMI@Elettra transfer lines, two application programs have been developed, OM_FEL1 for the transfer line FEL1 and OM_FEL2 for the transfer line FEL2[13]. The code provides two modes. The mode "normal" matches the optics of the transfer line upstream the first undulator of the radiator to the radiator optics which should be closed FODO optics. The mode "expert" allows, in one go, the above matching or Twiss functions matching in any position of the transfer line and the dispersion matching in the achromats. The input optics are read from the Fermi server and the user can select which one to use for the computations. The requests and necessary data for the code are in an input file. They can be changed inside the file or interactively from the menu "ChangeDefault". Fig. 7 shows OM_FEL2 panel.

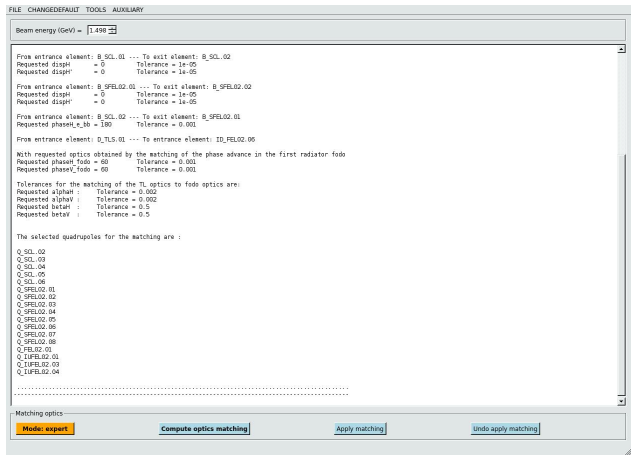

Figure 7: OM_FEL2 panel.

Fig. 8 shows the optics for FEL2 after the matching and application to the field, with the undulators closed to circular polarization.

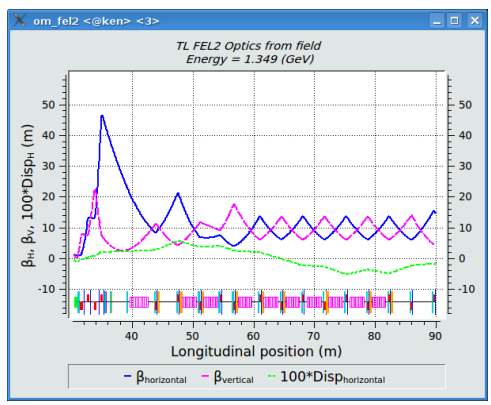

Figure 8: Optics after matching.

CONCLUSION

The original and the new high level software are working correctly. The new high level software, which was designed having in mind the booster, has shown to work very well with FERMI@Elettra and proved to be very useful during the commissioning and tuning of the new light source.

ISBN: 978-3-95450-132-8 doi:10.18429/JACoW-IPAC2014-THPRO107

REFERENCES

[1] D. Bulfone et al. "The Elettra Field Highway System", ICALEPCS'91, Tsukuba, November 11-15 1991.

[2] D. Bulfone, P. Michelini "The ELETTRA Power Supply Control System", EPAC'92, Berlin, March 24-28 1992; http://www.JACoW.org

[3] F. Iazzourene, "Calibration Coefficients of Elettra Magnets", internal report, September 1993, Trieste Italy.

[4] F. Iazzourene, "OrbitFB User's Guide", internal report, Trieste, Italy.

[5] M. Plesko et al., "The High Level Software of Elettra", EPAC'94, London, June 1994, p. 1776 (1994); http://www.JACoW.org

[6] F. Iazzourene, "TOCA: a Highly User Friendly Application Program for the Tune, Orbit, Dispersion and Chromaticity Correction", PAC'99, NY, March 1999, p. 2707; http://www.JACoW.org

[7] F. Iazzourene and C. Scafuri, "High Level Software Migration to Linux", internal report, Trieste, Italy. March 2006.

[8] C. Scafuri & F. Iazzourene, "Elettra New Full Injector High Level Software", EPAC'2006, Edinburgh, June 2006, p. 3353 (2006); http://www.JACoW.org

[9] F. Iazzourene and C. Scafuri, "Application Programs for the Elettra Booster Commissioning and Operation", EPAC'2008, Genoa, June 2008, p. 1535 (2008); http://www.JACoW.org

[10] L. Battistello, D. Bulfone et al. "The Control System of the Elettra Booster Injector", ICALEPCS'2005, Geneva, Switzerland.

[11] A. Götz, et al. "TANGO v8 - Another Turbo Charged Major Release", ICALEPCS'2013, San Francisco, CA, USA.

[12] M. Lonza et. al. "Status Report of the FERMI@Elettra Control System", ICALEPCS'2011, Grenoble, France.

[13] F. Iazzourene, "OM_FEL user's guide", internal report, June 2014, Trieste, Italy.