LNF-92/016

MARCO – models of accelerators and rings to commission and operate

L. Catani, G. Di Pirro, C. Milardi, A. Stecchi, L. Trasatti, M.J. Lee

Nucl. Instr. & Meth. In Phys. Res- A313 (1992) 273-276

Nuclear Instruments and Methods in Physics Research A313 (1992) 273–276
North-Holland

NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A

MARCO – models of accelerators and rings to commission and operate

L. Catani, G. Di Pirro, C. Milardi, A. Stecchi and L. Trasatti
INFN, Laboratori Nazionali di Frascati, P.O. Box 13, 00044 Frascati (Rm), Italy

M.J. Lee
Stanford Linear Accelerator Center, P.O. Box 4349, Stanford, CA 94309, USA

Received 30 August 1991

MARCO is a knowledge-based user interface for commissioning and operating modern accelerators and storage rings. Its purpose is to provide a model-referenced graphical interface system between the users and the machine. It allows access to modeling and simulation codes that are used in the design of the machine. It can be used to predict the effects of a change of parameters on the beam, or to compare the predicted with the measured effects. The design and prototype development of MARCO using HyperCard will be described in this paper.

1. Design codes

Two types of programs are used in machine design:

1) Lattice modeling – to define the location and strength of the beamline elements in order to obtain the desired lattice function values (e.g., the transport matrix and the Twiss parameter values). Examples of the lattice calculation programs are COMFORT [1], LEDA [2], etc.

2) Error simulation – to find the location or the strength of the beam monitors and correctors to change the beam errors (e.g. the beam trajectory and shape). Examples of the correction programs are RESOLVE [3], etc.

MARCO is an attempt to build a common graphical and interactive user interface to these programs, hiding the complexity of the many existing input–output data formats. The integration of such a tool in the control environment of an accelerator will greatly enhance the possibility of understanding and predicting machine behaviour for the operators, during both commissioning and normal operation.

1.1. Control applications

MARCO will be an interface to both types of programs: lattice modeling and error simulation. Using these programs, it is possible to operate the machine more intelligently. For instance, lattice modeling programs can be used to see the effects on the Twiss function values before making a change in the beamline elements; they also can be used to compute the strength of the beamline elements required for making a specific change on the Twiss functions. Both applications are needed in a procedure to "Setup the beam line". Also, trajectory error simulation programs can be used to predict the effects on beam trajectory before making a change on a corrector; they also can be used to compute the strength of the correctors required for making a specific change in the trajectory. Both applications are needed in a procedure to "correct the beam error".

1.2. Commissioning applications

The goal in commissioning is to find and correct the errors in the beamline. The error verification procedure involves two steps: 1) Identify the good regions by finding the largest regions where the prediction from the simulation codes agrees with the measurement; 2) Identify the errors by searching for the most likely candidate in the bad region (a bad region usually lies between two adjacent good regions).

For example, it will be possible to use MARCO to find field errors due to misalignment or miscalibration of the beam-line elements by analyzing the measured trajectory of a test beam. The measurement usually involves changing the beam trajectory by kicking the beam with some trajectory correctors and measuring the beam trajectory at the beam position monitors

(BPMs). In practice, 1) Focusing errors are found by analyzing a multiple set of beam oscillation data (an oscillation is defined as the difference between two sets of beam trajectory data); 2) Bending errors are found by analyzing a set of trajectory data (not their differences). Both applications are needed in a procedure to "verify the beamline". This procedure has been used successfully to find focusing errors and bending errors in SLC and to find the alignment errors in PEP and SPEAR at SLAC.

2. The system requirements

Recently, a user interface, GENI [4], has been developed for modeling and simulation programs such as COMFORT and RESOLVE. These modeling and simulation programs are normally used to control and commission any machine.

To use these programs, the user has only to read the beamline dataset which contains the element definitions and beam line definitions. The name, type, strength, etc. of every element are described in the element definition data, and the position along the beam line of the elements is contained in the beam line definition data. After reading in the beam line dataset, it is possible to use GENI to set up the beam line configuration and to analyze measured beam trajectories.

We would like MARCO to expand the range of GENI. It should offer a standard interface to all of the modeling and simulation programs that are commonly used for lattice design and error study. Since GENI was implemented on a MicroVAX workstation, its availability will be limited. MARCO is required to operate on a Macintosh II in order to be available to many more users. In particular, the use of HyperCard [5] on a MAC can offer a simple way to maintain and upgrade MARCO.

2.1. The HyperCard solution

We have designed an interface system for MARCO that can meet these requirements. Fig. 1 shows a block diagram for the layout of the interface system for MARCO.

For beam-line setup, the lattice modeling code interfaces to the input beam line description data and to the output data package. Two copies of the input data are provided: an original copy containing the element strength values of the design solution, and a second copy containing the strength values of a new solution. The user can run the modeling code to find the strength of the elements (FITTING) that produce the desired values of the TWISS functions at some specific points (FIT POINT). The output data include a table of the

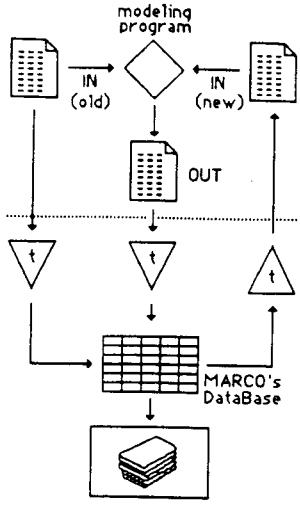

Fig. 1. A block diagram showing the layout of an interfacing system to MARCO for beam-line setup. A dotted line separates the modeling program from the translators (t) and MARCO.

Twiss functions at every element and the strength of the elements.

MARCO uses three translator codes: one for translating the input data from the data format of the modeling code, the second for translating the MARCO's output data to the format of the modeling code, and the last one to translate the output of the modeling code to the MARCO interface standard. After the input–output data have been translated, the user can make changes or select a display interactively using buttons, menus, graphs and windows.

At Frascati, most existing codes are working on a VAX computer. It is not necessary to implement them to work on the MAC. It is possible using HyperCard to run MARCO on the Mac and to run the modeling code on the VAX simultaneously. Alternatively, for a code that has been implemented to run on the MAC, both MARCO and the modeling code can run on the MAC concurrently. The use of HyperCard offers us a simple way to connect new programs to MARCO with minimum effort.

3. First implementation

The first two programs to be connected to MARCO are the lattice design codes LEDA and COMFORT. LEDA was originally written to run on a VAX using Top Drawer as a graphical interface. All of the I/O functions provided by Top Drawer are replaced by HyperCard.

Using MARCO it is possible to run Leda in two modes:

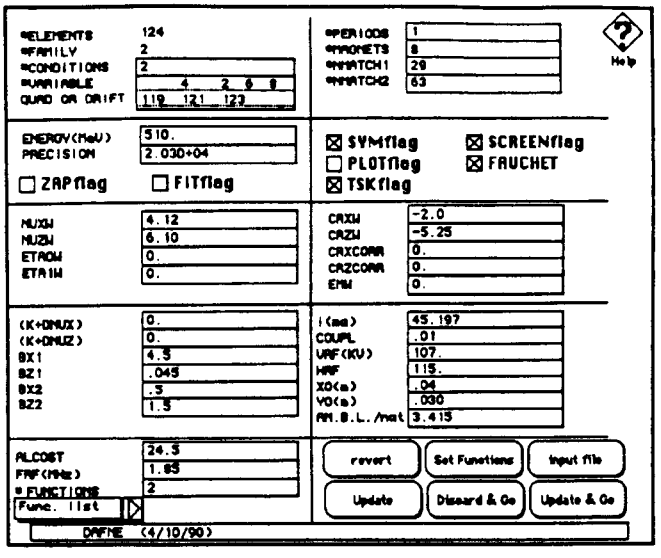

Fig. 2. General parameter setting window for LEDA.

– Importing the code to the Macintosh and recompiling it (using MPW 3.1 [6] and the Language System FORTRAN [7]).

– Using an RS232 interface to the VAX (the input and output files are transferred between Macintosh and VAX). In this mode, LEDA runs directly on the VAX cpu. In the future the use of the Apple Communication Toolbox will allow faster and more flexible communications with external computers.

These two modes of operation reflect different needs which may arise for different programs: While running a program on the machine for which it was originally intended is usually the easiest way to proceed, it may sometimes be convenient to be able to work in a standalone environment by running it on the Macinotsh.

As far as COMFORT is concerned, only the first method has been implemented, due to the high level of complication of the program.

To run LEDA or COMFORT three separate windows have been developed using HyperCard 2.

A) General parameter setting window (see fig. 2).

When this window is selected, a menu becomes available to load an input data file and to translate it to window B.

B) Element index and histogram window (see fig. 3). The elements are selected by the upper scroll bar which permits rapid scrolling through the entire machine. Once an element is selected by clicking on the corresponding icon, it is possible to change its parameter values, or to assign/delete a parameter from the list of elements to be used in fitting (for LEDA, fitting is the process of calculating the values of the variable element parameters to obtain the desired Twiss function values at some specific points along the beam line). These points are called the fit points. The output is displayed in graphic form. Multiple plots can be shown in the same window, including the output from a previous run of the program.

On window B, the user can click the buttons to: reset the machine to the original configuration; save the changes entered or discard them; go back to the parameter window; run the program on the Macintosh (only for LEDA); run the program on the VAX. If the last option is selected, the user is presented with window C.

C) VAX communication window, which allows through a simple terminal emulator to log onto the

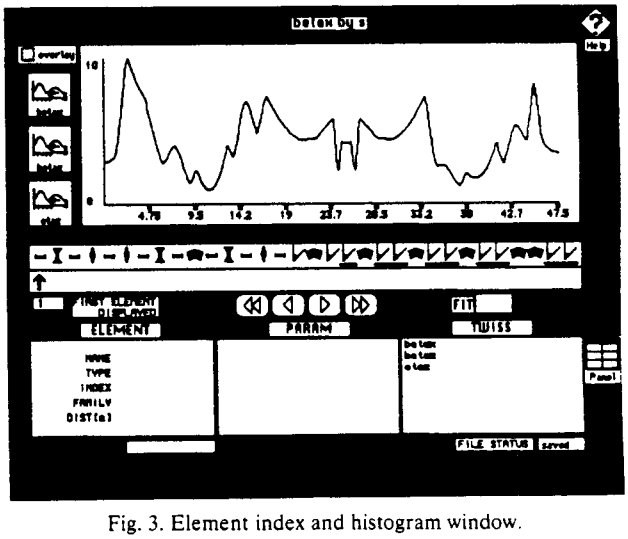

Fig. 3. Element index and histogram window.

VAX and to change directories. Moreover, while the program is being run, messages showing the status of the execution may be displayed on the screen.

4. Summary

The experience with HyperCard 2 has proven extremely positive. We have demonstrated that a user interface for a model-based control system can be implemented in a highly modular way that can ease debugging and improve maintainability, as can be expected from the use of an object oriented language (HyperTalk).

We have found that development time using HyperCard can be reduced to 1/5 or 1/10 of the development time using UIS graphics (VAX workstation). In particular, the number of lines of code has been reduced from 5000 (GENI) to less than 2000 (MARCO). We believe that the reduction in the number of lines will also lead to a reduction in the effort to maintain/upgrade MARCO.

The success obtained in building a common interface to programs as different as LEDA and COMFORT encourages us to continue to implement other modeling and simulation programs into MARCO. Eventually, MARCO will be the user interface for the model-based control system for the new PHI-factory DAΦNE under construction at the LNF.

Acknowledgements

We would like to thank the accelerator group of the LNF for continuing discussions and encouragement. We are particularly grateful to G. Vignola, C. Biscari and Steven Kleban (SLAC) for their help in the implementation of LEDA with MARCO.

References

[1] M.D. Woodley, M.J. Lee, J. Jäger and A.S. King, SLAC-PUB-3086 (March, 1983).
[2] G. Vignola, private communication.
[3] M. Lee, private communication.
[4] S. Kleban, M. Lee and Y. Zambre, Nucl. Instr. and Meth. A293 (1990) 475.
[5] HyperCard v2.0v2, Apple Computer Inc, © 1987–90.
[6] Macintosh Programmer Workshop v3.1, Apple Computer Inc, © 1985–89.
[7] Language System FORTRAN, Language System Corporation, © 1988.