An Approach to Portable Machine Physics Applications

M.Plesko*
Institute Jozef Stefan, Ljubljana, Slovenia
e-mail: mark.plesko@ijs.si
C.J. Bocchetta, F. Iazzourene, E. Karantzoulis, R. Nagaoka and L. Tosi
Sincrotrone Trieste, Trieste, Italy

ABSTRACT

Usually, shareable high level software (HLS) packages which perform machine physics calculations have to be run off-line on a specially prepared set of data. The programs that perform the necessary measurements contain explicit references to the underlying control system and are built with specific equipment in mind. Both reasons make it difficult to adopt existing HLS to another accelerator without substantial modifications.

An approach to overcome these limitations has been attempted at ELETTRA, a 2 GeV electron storage ring. The HLS applications use a) a data structure which uniquely describes the accelerator and is dynamically generated at run-time from a dedicated database and b) a set of utility routines which transparently access the control system and perform machine physics related calculations.

Both the data structure and the routines are based on common accelerator physics concepts and are not specific to ELETTRA. The user interface of the applications, which form a complete set of on-line commissioning and operations tools, is consistently realized with the X11/Motif toolkit and the code is written in the language C. Therefore, most of the applications, which total an effort of 10+ person-years, can easily be transferred to other accelerators as long as their control system uses UNIX workstations.

The data structure, the utility routines and a generalization of the concepts needed to obtain a completely portable set of HLS applications are presented in this paper.

I. INTRODUCTION

The historical approach in accelerator control was such, that the control system itself provided few complex actions, while accelerator physics related tasks were performed manually and then analyzed off-line by knowledgeable accelerator physicists. Some early approaches have been made in the direction of automated accelerator physics related tasks to solve individual problems like modeling [1], orbit correction [2] and measuring a machine physics parameter [3].

With the advent of distributed control systems, based on UNIX graphical workstations in the control room, it became easier to write specific application software. The graphical user interface available with the X11-windows system allows also non-physicists to use such applications. However, all those applications had been usually written for a given accelerator on an individual basis. They are highly interlinked with the specific code of its control system, contain accelerator specific computer code and do not profit from a common development environment. There is much discussion on sharing application software among accelerator centers [4] and it is acknowledged in the accelerator physics and control community that "... the implementation of application software might be rendered more efficient, opening ways to higher levels of automation and hence better physics" [5]. However, this has not been achieved so far.

We can see two reasons why the HLS applications could not have been extensively shared:

- There may be several control system independent HLS applications used for individual tasks, but they don't combine into a self-sustained class of independent *accelerator physics* software tools.
- Almost all applications contain explicit references to the underlying control system and are built with a specific control system and equipment in mind.

A concept to overcome these two limitations is shown in this article by introducing a unique data structure with data imported from a dedicated database and a set of utility routines which transparently access the control system. It had been successfully applied at the synchrotron radiation source ELETTRA [6], where for commissioning and operation a multitude of X11/Motif high level applications had been prepared and successfully used [7] on top of the control system [8]. They form a complete set as they cover all fields that are needed at ELETTRA: commissioning,

---

* Supported by the Slovenian Minstry for Science and Technology.

measurements, analysis, modeling, operations, machine monitoring, orbit correction and optics correction. Although the applications have been written by several authors, they all use the data structure and the routines as a basis. Both the HLS data structure and the routines are based on accelerator physics concepts. All those tools significantly reduced development time and ease future maintenance. As a special tool for the development of the HLS, a simulation program has been developed which can be linked to the applications instead of the control system in order to thoroughly test and debug the applications before real operation.

II. DESIGN CRITERIA

The HLS applications which were used during the commissioning and later for the operation of ELETTRA had been defined well in advance [9]. The applications had to cover all requests that had been raised by machine physicists and operations personnel. The user interface is based on X11/Motif, while the interface to the accelerator is realized through a separate machine physics-oriented software toolkit. The toolkit, which is composed of the data structure and the utility routines, has been developed for the following reasons:

- the HLS was authored by machine physicists who have little knowledge of the control system details;
- most of the HLS operates on machine physics variables which have to be readily available to the programmer;
- much of the constant data necessary for the HLS is not provided by the control system;
- the individual HLS programs should not contain any explicit reference to control variables;
- the actual control system parameters and procedure calls must be hidden from the HLS programmer in a transparent way;
- the HLS should be maximally flexible in case of hardware changes, even to the point of being easily portable to other accelerators.

During the design and development of the data structure and the utility routines, the following goals were considered as the most important ones, to:

- allow management of the constant data from one single source in order to quickly respond to hardware changes;
- keep control system related data separate from machine physics related data;
- design the structure as general as possible, ready for later requests and for use on other accelerators;
- provide all relevant machine physics parameters readily in the data structure or on a function call;
- create the data structure dynamically whenever the HLS program starts making it thus completely independent of the hardware platform;
- provide general modeling tools and the necessary data for each HLS application;
- channel all actions to/from the hardware through one routine which can be updated for any control system changes without any modification of the HLS;
- force HLS programmers to a standardized approach thus simplifying maintenance of the HLS.

The data structure is composed of two parts: the data files and the run-time data structure (see figure 1). The data files contain the calibration data and other constant parameters of the accelerator elements. The run-time data structure keeps these constant data together with other data that are read in from the control system and converted to machine physics parameters, so that it is easily accessed by the HLS program.

The utility routines are divided into three layers (figure 1). The lowest layer is communicating with the control system and is not directly accessed by the HLS. The middle layer provides generic routines to access single components or groups of them and to store obtained data in the run-time data structure. The uppermost layer operates only on machine physics variables and provides modeling tools. The HLS application is written on top of these layers.

III. THE DATA STRUCTURE OF THE HLS

A. The Data Files

The *Machine file* provides the mechanism to save and restore the status of the machine - the transfer line and the storage ring in the case of ELETTRA. The Machine file is used to restore the equipment which is relevant for machine physics and machine operation. It contains the currents of all magnets (bending, quadrupole, sextupole and correctors), the insertion device gaps and the injection element settings. Both the set values and the actual read values are recorded.

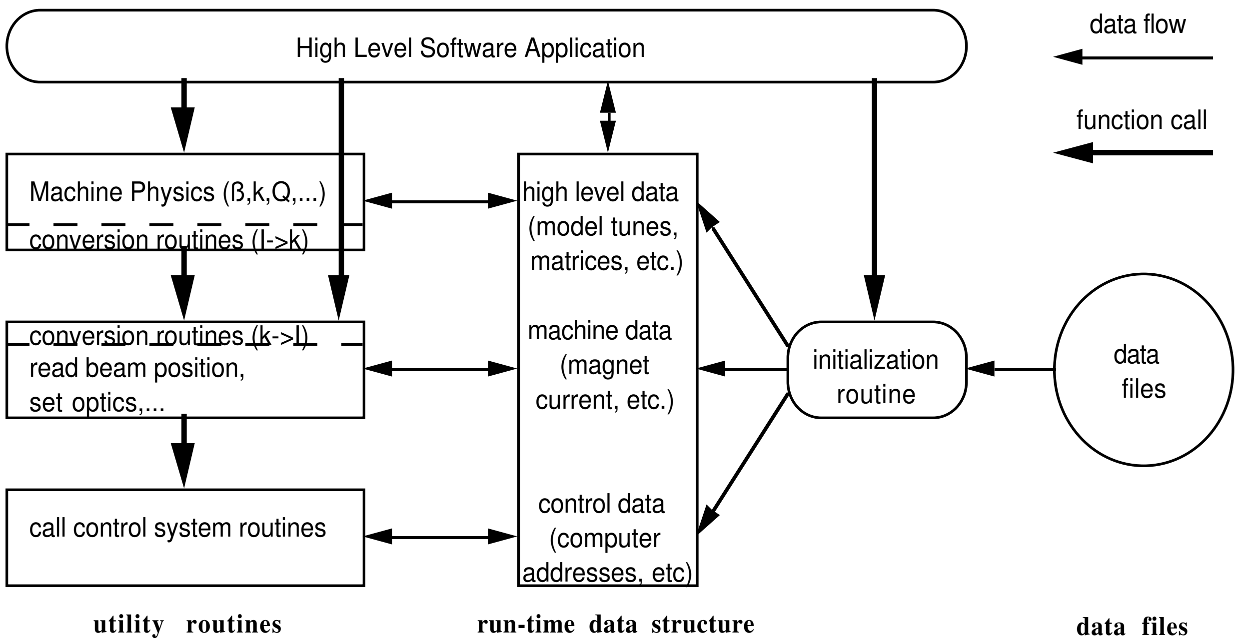

Figure 1: The components of the HLS data structure in relation to a HLS application.

The Machine file can be considered to be a snapshot of the machine. All HLS applications can access either the actual machine directly or read from the Machine file, which represents the machine at another point in time. Conversion routines convert the currents from the Machine file into machine physics variables and vice versa.

The other data files contain the constant parameters that describe the ELETTRA transfer line and storage ring from the machine physics point of view. The data in the files are read by an initialization routine, which must be called by the application program. The routine creates the run-time data structure providing access to all the data from within application programs. For a more complex accelerator plant, the files would have to be a part of a relational database. In the case of ELETTRA, the files are implemented as pure ASCII text files for simplicity. One record is represented by one line. Each record is given a unique name according to the control system naming convention. Links to other records in other files are realized by referencing these names.

The files are called the *Calibration File*, the *Parameter File*, the *Access File* and the *Structure File*. The Calibration File contains data obtained from the calibration of elements, mainly magnetic elements (bend, quad, sextupole, ID, corrector, etc.) but may contain calibration data for any type of equipment. The Parameter File contains constant parameters that describe non-changing properties of elements, such as length, lower and upper limits of variables, etc. The Access File contains the control system addresses and references to relevant records in the Parameter and Calibration File. Finally, the Structure File represents the logical structure of the transfer line or the storage ring, with their elements listed according to their position in the structure. For each element, there is one record containing the name and the type of the element and references to the relevant Access and Calibration File records.

B. The Run-Time Data Structure

The high level software (HLS) run-time data structure is a global data structure implemented in C, which contains all important parameters that describe the ELETTRA transfer line and storage ring from the machine physics point of view. Its hierarchical structure provides a logical and efficient access to all the data from application programs. It consists of a multitude of data records declared as *structs* and references to and among them declared as *pointers*. For a complex accelerator one can not keep all relevant data in the run-time memory of each application. In such a case the static data would have to be stored in a run-time database in shared memory, available to all applications.

For each physical element in the structure (magnet, drift section, monitor, etc.) there is a corresponding general_record, which contains general data about this element, as well as a corresponding *specific* record, which contains data depending on the type of the element. The specific records are drift_record for drifts, magnet_record for bends, quads and sextupoles, etc. The data which are the same for several elements, such as the current of a power supply which drives several magnets, are stored in the family_record. Other records link elements that are acted on by a feedback system. The records contain both the fixed data that describes an element as well as the variable data that has been acquired through the control system.

To each element of finite length in the structure there is a corresponding 3x3 horizontal and vertical transfer matrix and a twiss_record, which contains the Twiss functions in both planes and their derivatives at the position of the element. The list of records includes also the access_record containing control system related data from the access file, the parameter_record which is read from the parameter file and others.

Programs access the records only via pointers. One array of pointers is pointing to each element's general_record, ordered according to the positions of the elements in the structure file. Another access to elements is possible through the *specific* pointers, which point to *specific* records, declared for each type. Records themselves contain various pointers, allowing access of data through several levels of the data hierarchy. Several records may have pointers to a record with common data. Pointer links are bi-directional where this is useful for the programmers.

IV. The Utility Routines of the HLS

Actions which are common to many programs have been gathered in the set of utility routines which are provided to the programmer as a software package containing basic tools. They can be divided into two groups: machine physics calculations and actions on the machine. The routines which perform actions on the machine do not communicate with the control system directly in order to keep the explicit references to the control system at a minimum. Instead they register the individual actions via the run-time data structure and call a special routine which handles all control system specific tasks.

Thus there are three layers of utility routines (see figure 1), two of which are accessible by the application programmer: the *control system layer*, the *machine action layer* and the *machine physics layer*.

The control system layer has been added only to allow transparent access to the control system via one well defined point. The advantage is that individual programmers do not even have to take into account the explicit implementation of the control system. The control system had been developed in parallel and all changes were easily added to the routines of the control system layer without changing any line of HLS code. As a side product, it was easy to replace the control system layer with a machine simulation program in order to test each application before releasing it for use on the real machine.

The machine action layer contains the most routines. It includes routines which read/set one single power supply current, switch on magnets in a pre-defined way, routines which act on all power supplies of a given type, routines which close or open the scraper and the insertion device (ID) gaps, read the current and calculate the lifetime, read the current state of the machine into the run-time data structure, save the run-time data-structure to a machine file, etc. The input and output of these routines do not include any explicit reference to the actual control system.

A set of conversion routines converts the machine parameters like magnet currents, insertion device gaps, etc. to machine physics parameters like quadrupole strengths, ID strengths and vice versa. The machine physics parameters are being acted upon by machine physics routines, most notably the routine update_twiss, which calculates the transfer matrices and the linear model of the machine in one step. For a HLS program to use the model of the actual state, all it takes is a call to the routine input_all which sequentially reads all magnets, converts the currents into strengths and finally calls update_twiss.

V. Towards Complete Portability

The high level software was used very successfully during the commissioning and operation of ELETTRA (see [7] and references therein), showing that it is based on solid concepts. It is also planned to port major parts of the HLS to the new light source BESSY II.

However, the approach of ELETTRA is still limited in the following ways:

- the HLS is based on a synchrotron radiation source and the data structure and utility routines lack accelerator physics concepts that are needed at other types of accelerators
- although the code itself is independent of actual control system parameters, it still bases part of its data structure on the explicit design of the ELETTRA control system;
- therefore a considerable amount of work has to be done when transferring the HLS to an accelerator with a different control system philosophy;
- the specifics of the equipment are - albeit hidden from the application designer - hardcoded in the utility routines;

- as a result, any hardware modification results in the need to rewrite the functions and to relink all existing applications.

Future developments should overcome these limitations and result in a truly portable set of HLS applications without any modifications of the application code as long as the control system of the other accelerator supports process communications, X-windows and C. All accelerator-specific information will be contained in textual form in the HLS database, which will be read by the HLS at run-time. Only one routine will contain all explicit calls to the control system.

There are two different ways to implement such a package at the lowest layer:

1) As already indicated in the previous paragraph, all accelerator specific data is provided in a database. In order to use these data, the heart of the lowest layer will be an interpreter, which will digest textual data and component descriptions and issue the relevant commands to the control system. Therefore any change in the accelerator structure, control system or in the handling of individual components can be adapted by simply editing a text file or database entry, without the need to recompile any HLS program. The advantage of this system is that even during commissioning, when certain control tasks have to be redefined (e.g. the speed of setting magnets), one can adapt without the need for recompilation. The disadvantage of this system is that the user must learn a relatively complex interpreter language.
2) A complete set of tasks that are necessary for all applications is defined, like switch_on_magnet, set_one_magnet, set_all_magnets, read_all_beam_position_monitors, etc. The routines which perform these tasks are then implemented for each control system separately. The control system details are hidden in the implementation and any ported application is just linked to the low level library. The advantage is the flexibility of the C language, which allows for complicated specific control tasks to be implemented in the utility routines.

There will, of course, always remain needs for specific, accelerator dependent applications. Therefore one should not only concentrate on the set of applications, but also on the tools that should be given to local application programmers. By using the tools in their own applications, the programmers will also be able to understand and possibly tune the applications they have obtained through sharing. Thus it is necessary to define a generic set of utility routines and data structures and make it a **standard** for machine implementations. Because in the end, it is only standards that allow an efficient sharing, be it simple screws, source codes, or complete control systems.

Such a scheme would be impossible if it was not that accelerator physics concepts are the same at all accelerators. It is obvious that this can not be done on the level of the control system, although even there are strong trends for a common approach: the control system EPICS is becoming widely used [10]. The HLS presented in this paper can serve as a good starting point and is an explicit proof that such a project is feasible.

An interesting possibility is to implement such a HLS under EPICS. The system EPICS provides exactly the type of standard which we need. Once the HLS is attached to EPICS, the sharing of it with the laboratories that already use EPICS will be become very simple.

VI. CONCLUSIONS

The design of the HLS at ELETTRA was made such that its structure is not specific to ELETTRA. Therefore it should be possible to use the applications at other accelerators. This flexibility has been achieved by introducing a separate machine physics oriented data structure and a set of utility routines.

The data structure contains all data which describe a transfer line or a storage ring from the accelerator physics point of view and in addition contains all necessary references to the control system in order to access single and multiple controlled points. The utility routines, which use the control system in a transparent way, access single components or groups of them, store obtained data in the run-time data structure, convert physical parameters into machine physics variables and provide modeling tools.

However, in order to reach the stage of common application sharing, a new set of standard routines and data structures for common accelerator operations that are used by high level applications has to be defined and rigorously implemented.

VII. REFERENCES

[1] M.D. Woodley, L. Sanchez-Chopitea, H. Shoaee, *Application of Online Modeling to the Operation of SLC*, Proc. IEEE Particle Accelerator Conference 1987, Washington 1987, p.772.

[2] W. Herr et al., *A New Closed Orbit Correction Procedure for the CERN SPS*, Proc. IEEE Particle Accelerator Conference 1989, Chicago 1989.

[3] M.C. Ross, N. Phinney, H. Shoaee, J.C. Sheppard, *Automated Emittance Measurements in the SLC*, Proc. IEEE Particle Accelerator Conference 1987, Washington 1987, p.775.
[4] J. Poole, *Requirements and Trends in the Control of Accelerators*, Proc. European Particle Accelerator Conference 1994, London 1994, p. 332.
[5] A. Daneels, *Current Trends in Accelerator Controls: The Issue of Application Software, Particle Accelerators*, **29** (1990), 173-182.
[6] ELETTRA - *Conceptual Design Report*, Sincrotrone Trieste, April 1989.
A.Wrulich, *Status of ELETTRA*, Proc. European Particle Accelerator Conference 1994, London 1994.
[7] M.Plesko et al., *Commissioning and Operation Applications of ELETTRA*, this conference.
[8] M. Mignacco, *The ELETTRA Control System*, Nucl. Instr. Meth., **A293** (1990).
D. Bulfone, *Status and Prospects of the ELETTRA Control System*, Proc. ICALEPCS 93, Berlin 93.
[9] M.Plesko, *The High Level Software Routines*, Sincrotrone Trieste. ST/M-TN-91/11. 1991.
[10] M. Knott et al., *EPICS: A Control System Codevelopment Success Story*, Proc. ICALEPCS 93, Berlin 1993.