ACCELERATOR CONTROL MIDDLE LAYER*

J. Corbett, G. Portmann and A. Terebilo, SLAC, Stanford, CA 94309, USA

Abstract

This paper reviews an efficient implementation of the software 'middle layer' that resides between high-level accelerator control applications and the low-level accelerator control system. The middle layer software is written in MATLAB and includes links to the EPICS Channel Access Library. Functionally, the middle layer syntax closely parallels the Family/Index naming scheme used in many accelerator simulation codes and uses the same convention to communicate with both the online machine and the accelerator model. Hence, machine control, machine simulation and data analysis tools are integrated into a single, easy-to-use software package.

INTRODUCTION

As shown in Fig. 1, the *middle layer* provides a set of functions that communicate with machine hardware via the MATLAB Channel Access toolbox MCA [1]. At the heart of the middle layer is a data structure containing *Accelerator Objects* or Families of hardware elements with various attributes: element names, element indices, i/o channel names, unit conversions, etc. The naming scheme mimics the Family/Index convention commonly used in accelerator simulation codes. Hence, the language of simulation codes can be used to communicate directly with either online accelerator components or the model.

The middle layer family definitions are contained in a text file for easy editing. Typical families include dipoles, quadrupoles, sextupoles, correctors and BPMs. An additional *Accelerator Data* structure contains default directory specifications, file names and basic accelerator parameters. Execution of a simple MATLAB script loads both the *Accelerator Object* (AO) and the *Accelerator Data* (AD) blocks into memory - all routines in the middle layer toolbox have direct access to the AO and AD data.

Middle layer *functions* are used to communicate with accelerator hardware and access different family attributes. At present, hardware communications occurs via EPICS Channel Access. In this case, the middle layer provides channel names and keeps track of integer handles for each device thereby buffering the user from detailed Channel Access calls with complicated channel names. Other communication protocols are also possible. The middle layer also accommodates an accelerator model in the MATLAB Accelerator Toolbox (AT) [2,3] or can communicate with a MATLAB model server operating in an EPICS ioc [4]. The ability to switch between 'simulator' and 'online' modes is useful for program development and analysis.

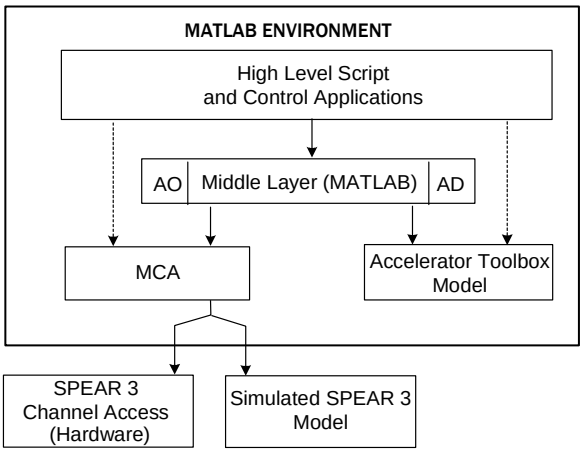

Figure 1: Middle Layer Software Flow Diagram

By design, the middle layer is machine independent – communication with different machines requires the user to reconfigure the *Accelerator Object* file, revise the *Accelerator Data* structure and update the model. Special functions may be required for machine-specific hardware.

USE OF MATLAB

One key feature of our approach to the middle layer is the use of MATLAB. MATLAB provides an active variable workspace, a built-in math library, powerful graphics capabilities and on-going development of new software features. Just as MATLAB can be augmented with commercial 'toolboxes', the Accelerator Toolbox (physics) [2,3], the MATLAB Channel Access Toolbox (EPICS interface) [4] and the Middle Layer Toolbox (controls & data organization) [5] facilitate accelerator simulation and control. All of these functions make use of the array processing capabilities inherent in MATLAB.

At the application level, script-based control sequences and graphical interfaces utilize the middle layer to standardize and simplify programming. At the highest level, MATLAB and the associated toolboxes can be used to control the accelerator - at the Advanced Light Source MATLAB is used for energy ramp, configuration save/restore, global orbit correction, insertion device compensation and beam-based alignment [6]. Response matrix analysis routines are in turn used for accelerator calibration and lattice studies [7]. Several of the high-level ALS functions have been ported to SPEAR 3 and upgraded to the middle layer formalism. Well before SPEAR 3 start-up, the MATLAB tools were used for physics studies and simulated commissioning [8,9].

*work supported in part by Department of Energy Contract DE-AC02-76SF00515 and Office of Basic Energy Sciences, Division of Chemical Sciences.

Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309
To appear in the Proceedings of Particle Accelerator Conference (PAC 03), Portland, OR, 12-16 May 2003.

MIDDLE LAYER NOMENCLATURE

In the EPICS environment each hardware device has a unique set of identifiers or *Process Variables* (PV). Accelerator physicists, however, often think in terms of hardware families (dipoles, quadrupoles) and attributes of the family elements (length, strength, etc). In the middle layer, each family has a nominal set of structure fields (element names, element indices, channel names, etc). Specific hardware elements in a family are referred to by {Family, DeviceList} where DeviceList is an integer doublet {Sector, Index}. A further division of the family structure into *Monitor* and *Setpoint* sub-structures keeps element attributes well organized and fits neatly into the middle layer function architecture. The EPICS setpoint PV names, for instance, are found in Family.Setpoint.Channelnames.

Middle layer *function* names are characterized by a prefix to indicate action: get=[retrieve value]; set=[deposit value]; meas=[measure]; calc=[calculate]. *getsp* retrieves a setpoint, whereas *measchro* measures chromaticity. Step- and ramp- functions are wrappers for the 'set' routine. Wherever possible, functions are written in machine-independent format.

MODES OF OPERATION

Middle layer software can be run in several modes of operation. The *online* mode broadcasts get/set calls to EPICS Channel Access servers. The servers can be connected to live hardware modules or a model server [4]. The *simulation* mode directs get/set calls directly to the local AT model [2,3]. This mode is useful to develop and test control programs prior to deployment and for programs not intended for online use. In practice, get/set calls check if the mode is 'online', 'simulator', 'manual', or 'special.' The 'manual' mode prompts the user for manual data input (e.g. tunes) while the 'special' mode allows the user to define an in-line function to numerically process data.

MIDDLE LAYER FUNCTIONS

The middle layer function toolbox is well established and continues to expand. At present, it contains about 100 functions.

Get and Set Functions

These core functions communicate with Channel Access Servers or the MATLAB Accelerator Toolbox. The two main functions are *getpv* (*get* EPICS PV) and *setpv* (*set* EPICS PV). Both functions accept a variety of input formats via the Family/Index convention. Rather general calls are permitted and timing requests are possible. It is important to note that the MCA toolbox communicates with the *.val* field of an EPICS record. Nevertheless, each *Accelerator Object* family can contain many PV channel names for a given hardware device. A quadrupole magnet family, for instance, can have setpoint, monitor, voltage, and status PV channel names that refer to the *.val* field in the associated EPICS records.

Utility Functions

Utility functions allow easy conversion between fields in an *Accelerator Object* family. Examples include family2common (convert family name to element common names), common2dev (convert common names to numerical device indices) and common2channel (convert common names to PV channel names). *getfamilydata* is a particularly important utility function used to access information from an *Accelerator Object* family.

Shortcut Functions

Shortcut functions are designed to reduce number of parameters required in a function call. Examples include *getsp* and *setsp* which communicate with *setpoint* PV's, and *getx/gety* which return horizontal and vertical beam position values. Reference to the channel access handles is performed in the base routines *getpv/setp*.

Unit Conversion Functions

Unit conversions play an important role in modeling the on-line machine. For this purpose, the middle layer supplies two functions *HW2Physics* (hardware-to-physics) and *Physics2HW* (physics-to-hardware). Both functions accept the Family/Index naming convention but refer to the *Accelerator Object* database to retrieve the numerical conversion algorithm and associated parameters. For SPEAR 3, polynomial current-to-field transfer functions are used for each magnet family. Each individual magnet has an additional numerical scaling factor for detailed modeling applications.

Simulator Functions

These functions *only* communicate with the AT model to return simulated physics parameters. Examples include *getbeta*=(calculate beta functions), *getchro*=(calculate chromaticity) and *getdisp*=(calculate dispersion). MATLAB functions in the AT toolbox can also be used directly to augment the set.

Special Functions

Some devices do not conform neatly with the Family/Index formalism so special functions are created to access the data. An example is *getid/setid* for insertion device gap control. Alternatively, since it is easy to create *Accelerator Object* families, special families can be added for a specific task. The storage ring tunes, for instance, can be represented by a family structure containing fields for the common names, channel access handles and golden tunes. For ramping applications an *Accelerator Object* with every magnet involved in the ramp can be created.

DATA MANAGEMENT

Robust data management for accelerator control and accelerator measurements can be a challenging task. The *Accelerator Objects* framework organizes element names and attributes in a local database. In principle, much of the AO data can be loaded from a master site-wide database if it exists. But this is not always the case - SPEAR 3 studies in MATLAB commenced years before a database was available. The *Accelerator Data* structure contains machine- and middle layer specific data that resides outside of the site-wide database. Examples include calculated physics parameters and directory locations to store measured data.

APPLICATION PROGRAMS

A primary reason for middle layer software is to simplify script construction and high-level application programming. Scripts rely heavily on middle layer software to perform correlated perturb/measure studies. Application programs can be dominated by user-interface software but again benefit from the middle layer for machine control and data handling. In both cases the middle layer buffers the user from detailed Channel Access calls. The middle layer also provides high-level functions for common accelerator physics tasks. Examples include:

(1) *measrespmat* - measure response matrix
(2) *getrespmat* - read response data from files
(3) *measdisp* - measure the dispersion function

SUMMARY

The MATLAB middle layer provides convenient and easy-to-use 'glue' for experimentalists and application programmers to access online hardware, model programs and data analysis tools. In conjunction with the inherent flexibility of MATLAB, the middle layer allows rapid software development and testing. Due to the machine-independent nature of the software, it is readily adapted to other accelerators, particularly storage rings and LINACS which feature magnet families and repetitive cells. In principle, the middle layer can be adapted to other applications in experimental physics and industrial control.

ACKNOWLEDGEMENTS

The authors would like to thank D. Robin, H. Nishimura and C. Steier at the ALS for a fruitful on-going collaboration - many of the original concepts were developed under their watch. We are also grateful to M. Cornacchia, J. Safranek and the SPEAR 3 project for support in this area.

REFERENCES

[1] A. Terebilo, "Channel Access Toolbox for MATLAB," Proc. of 8th ICALEPCS, San Jose, USA, Nov. 2001.
[2] A. Terebilo "Accelerator Modeling with MATLAB Accelerator Toolbox," PAC'01, May 2002, pg. 3203.
[3] A. Terebilo, "Accelerator Toolbox for MATLAB," SLAC-PUB-8732 and www-ssrl.slac.stanford.edu/at/.
[4] A. Terebilo, "Simulated Commissioning of SPEAR 3," these proceedings.
[5] G. Portmann, et al, "Middle Layer Software for Accelerator Control", SSRL Internal, Dec. 2002.
[6] G. Portmann, "ALS Storage Ring Setup and Control Using MATLAB," LBL LSAP Note #248, June 1998.
[7] J. Safranek, et al, "Linear Optic Correction Algorithm in MATLAB," these proceedings.
[8] J. Corbett, et al, "Orbit Control Using MATLAB," PAC'01, Chicago, May 2002, pg. 813,
[9] A. Terebilo, "Global Beam-Based Alignment Method," these proceedings.