AN ACCELERATOR CONTROL MIDDLE LAYER USING MATLAB*

G. Portmann LBNL, Berkeley, CA 94720 U.S.A.
J. Corbett, A. Terebilo SRRL/SLAC, Stanford, CA, 94309 U.S.A.

Abstract

Matlab is an interpretive programming language originally developed for convenient use with the LINPACK and EISPACK libraries. Matlab is appealing for accelerator physics because it is matrix-oriented, provides an active workspace for system variables, powerful graphics capabilities, built-in math libraries, and platform independence.

A number of accelerator software toolboxes have been written in Matlab - the Accelerator Toolbox (AT) for model-based machine simulations, LOCO for on-line model calibration, and Matlab Channel Access (MCA) to connect with EPICS. The function of the MATLAB 'Middle Layer' is to provide a scripting language for machine simulations and on-line control, including non-EPICS based control systems. The Middle Layer has simplified and streamlined development of high-level applications including configuration control, energy ramp, orbit correction, photon beam steering, ID compensation, beam-based alignment, tune correction and response matrix measurement.

The database-driven Middle Layer software is largely machine-independent and easy to port. Six accelerators presently use the software package with more scheduled to come on line soon.

INTRODUCTION

Matlab was originally used for machine control at the ALS in the early 1990's, shortly after commissioning. For algorithm development, the combination of a matrix-oriented programming language with built-in math libraries, an active workspace for system variables, and platform independence was quite desirable. In the beginning it was primarily used as a scripting language for accelerator physics studies but quickly found it's way into operations as well.

The fact that Matlab was reliable enough to run operational code allowed one to produce machine control and accelerator physics programs using the same language. It is now is used for routine machine operation (energy ramp, configuration save/restore, orbit correction and feedback, and ID compensation) as well as machine setup (beam-based alignment, tune and chromaticity correction, LOCO, photon beam steering, etc.) and as a scripting language for accelerator physics studies [1]. The method of connection between Matlab and the control system has changed over the years but the ALS presently uses EPICS with Simple Channel Access (SCAIII).

At SSRL, parallel Matlab developments in the late 1990's led to the Accelerator Toolbox (AT) to simulate machine performance [2], Matlab Channel Access Toolbox (MCA) for EPICS connections [3], an orbit control gui [4], and LOCO for accelerator model calibration [5]. In a collaborative effort between ALS and SSRL, many of the control functions developed at the ALS were ported to SSRL where the functionality was extended, the software was re-structured to incorporate MCA and made machine-independent. The entire software package (AT, MCA, Middle Layer) was successfully deployed and used extensively to commission SPEAR 3 [6,7]. As a result, the Middle Layer software is now easily ported to other machines [8].

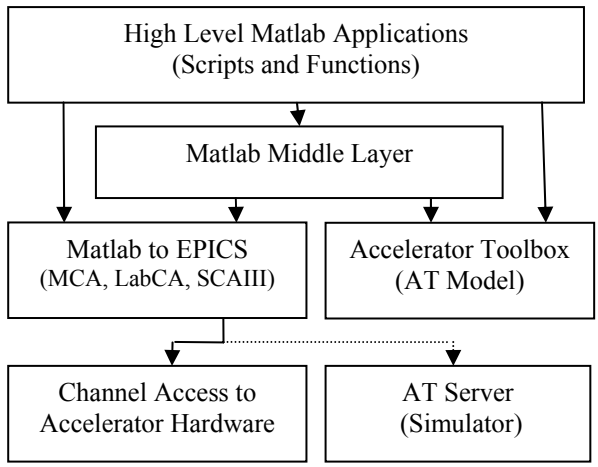


Figure 1: Software flow diagram.

SOFTWARE OVERVIEW

As shown in Fig. 1, the Middle Layer software provides a library of functions that either communicate with machine hardware for online applications or with AT for model-based applications. It is important to note that although the online get/set routines originally communicated via EPICS Channel Access, they can now communicate with a variety of control systems. There are only two core functions that need to be re-programmed to work with each new control system — *getpvonline* and *setpvonline*.

One of the central features of the Middle Layer is to simplify the channel naming scheme used by the host control system. Channel names are often quite obtuse so it is best not burden the programmer with hardware nomenclature. The same is true for model-based applications – locating and keeping track of numerical element indices can be a cumbersome process. The Middle Layer organizes element names into groups (Families), subgroups (Fields), and uses a device numerology that mimics naming schemes commonly used in accelerator simulation codes. Element QF [3,1], for example, refers to the first QF magnet on third magnet

* This work was supported U.S. Department of Energy under Contract No. DE-AC03-76SF00098 and DE-AC03-76SF00515.

raft. By using the Middle Layer, the same terminology applies for both online and model-based applications.

At the heart of the Middle Layer is a the 'Accelerator Object' data structure (AO). The AO maps the Family/Device syntax to the control system hardware and contains attributes for each Family (device list, channel names, hardware-to-physics conversion factors, range information, etc.) The AO resides in hidden memory associated with the Matlab command window. By issuing a simple Matlab command, the AO structures are initialized and a software path is established to access the Middle Layer functions and application programs.

A parallel structure, called Accelerator Data (AD), contains directory locations, key file names and basic accelerator parameters. Response matrix data, for example, is automatically stored and retrieved from the correct directories. The AD also resides in hidden memory. A description of how to build the AO and AD structures and how the file and function architecture are arranged can be found in the Middle Layer manual [1].

MIDDLE LAYER NOMENCLATURE

In a typical hardware control environment each device is referenced via a specific channel name. Accelerator physicists, however, often think in terms of hardware families (dipoles, quadrupoles, BPMs, etc) and attributes of the family elements (length, strength, gain, etc). In the Middle Layer software, each family is represented in the AO by a structure with well-defined fields. A further division of each Family structure into *Monitor* and *Setpoint* sub-structures keeps element attributes well organized and fits neatly into the architecture of the Middle Layer software functions.

MODES OF OPERATION

The Middle layer software can be run in several different modes. The ability to switch between *online* and *simulate* modes is helpful for model-based analysis and to debug software prior to deployment on the operating accelerator. In the *online* mode, get/set calls are broadcast over Ethernet to remote computers (EPICS IOC's, for example). The remote computers can either connect with live hardware modules or function as a remote model server [9]. The *simulation* mode directs get/set calls directly to the local AT model. This mode is useful to develop and test control programs and for programs not intended for online use. In practice, get/set calls check to see 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 (e.g. special unit conversion procedures). Similarly, the Middle Layer can switch between *hardware* and *physics* units depending on requirements of the particular software.

MIDDLE LAYER FUNCTIONS

The Middle Layer function toolbox is well established and continues to expand. At present, it contains over 100 functions.

Get and Set Functions

These core functions communicate with control system or the MATLAB Accelerator Toolbox. The two main functions are *getpv* and *setpv*. Both functions accept a variety of input formats via the Family/DeviceList convention. Rather general calls are permitted and timing requests are possible.

Utility Functions

Utility functions allow easy conversion between different fields in the Accelerator Object families. Examples include family2channel (convert Family/Device to channel names) and GetList (retrieve list of devices in a Family). *getfamilydata* is an important utility function used to access any information contained in the AO.

Shortcut Functions

Shortcut functions are designed to reduce the number of parameters required in a function call. Examples include *getsp* and *setsp* which communicate with the device *setpoint*, and *getx/gety* which return horizontal and vertical beam position values. Shortcut functions are used widely during machine physics studies and in application program development.

Unit Conversion Functions

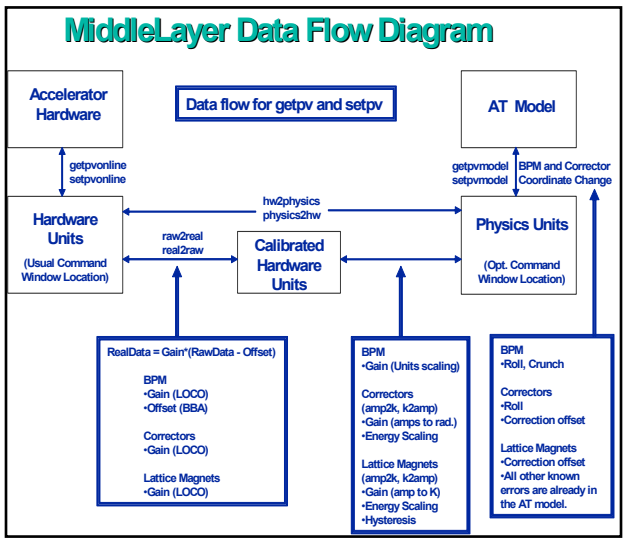
Unit conversions play an important role in modeling the on-line machine. For this purpose, the Middle Layer supplies two functions *hw2physics* (hardware-to-physics) and *physics2hw* (physics-to-hardware). The data flow diagram shown below outlines the conversion algorithm and associated parameters.

Simulator Functions

Using the argument 'model', Middle Layer functions can access the AT model directly. Examples include *measTuneResp('model')* and *getBPMResp('model').* All Matlab functions in the AT toolbox are also available for accelerator simulation.

Special Functions

Special functions can communicate directly with the AT model to return simulated physics parameters. For example, *getbeta* returns the beta functions from the model. In addition, some devices such as storage ring tune do not conform neatly with the Family/Index formalism so *special* functions are created to access the data.



APPLICATION PROGRAMS

The Middle Layer software simplifies high-level application programming for machine control and script development for accelerator physics. Matlab 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 data manipulation, file handling and machine control. In both cases the Middle Layer greatly simplifies communication with accelerator hardware.

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

A centralized accelerator control gui *PlotFamily* launches many high-level Middle Layer functions and can graphically displays live data, simulated data or data contained in the AO.

SUMMARY

The Matlab toolboxes written for accelerator applications (AT, MCA, LOCO, Middle Layer) are well integrated and have proven quite useful for machine studies and control. The relatively user-friendly software and machine-independent programming language have fostered a number of collaborations. Most scientists find the syntax quite intuitive making it possible for visitors to participate in accelerator physics studies with minimal training. To date, the software has been installed on six machines (ALS, CLS, NSLS VUV, NSLS X-ray, PLS, SPEAR3) and has received interest from other laboratories including ASP, ALBA, CAMD, DIAMOND, SOLEIL, and SSRF.

The Middle Layer requires a small, upfront investment to build the AO, AD and AT-model files in order to install on a new accelerator. The complete software can be made functional for most applications within a few days and be fully operational in a few weeks, including exercising routines in the *simulate* and *online* modes. Developing a fully calibrated online model with accurate hardware-to-physics conversion parameters is the most time-consuming part of the setup.

Having multiple laboratories use the same high level software has proven to be quite useful.

- Not every laboratory has to spend the resources to write the same algorithms. For new laboratories it's a very fast and inexpensive way to acquire high level control and simulation software.
- Having one software package that is debugged at many laboratories improves reliability. Thousands of dedicated accelerator hours have been spent testing, debugging and improving the AT/MCA/Middle Layer software packages.
- As with any collaboration, software expansion, suggestions, and new ideas come from a bigger pool of participants.
- The number of physicists and engineers trained on the Middle Layer is growing rapidly. Visiting scientists can work immediately on the new accelerator. This was very useful for SPEAR 3 commissioning.
- Since it's easy to switch between different accelerators in the simulation mode, it's easy to test algorithms on different accelerators.

ACKNOWLEDGEMENTS

The authors would like to thank the ALS/LBNL and SPEAR/SLAC management for encouraging and supporting a productive collaboration on the Middle Layer. We would also like to thank the staff members at the laboratories that use the Middle Layer for the helpful suggestions that everyone has made along the way.

REFERENCES

[1] G. Portmann, J. Corbett, A. Terebilo, "Middle Layer Software Manual for Accelerator Physics," LBNL Internal Report, LSAP-302, 2005.
[2] A. Terebilo, "Accelerator Modeling with MATLAB Accelerator Toolbox," PAC 2001.
[3] A. Terebilo, "Channel Access Toolbox for MATLAB," ICALEPCS 2001.
[4] J. Corbett et al., "Orbit Control Using MATLAB," PAC 2001.
[5] J. Safranek, G. Portmann, A. Terebilo and C. Steier, "MATLAB Based LOCO," EPAC 2002.
[6] J. Corbett, G. Portmann, A. Terebilo, J. Safranek, "SPEAR 3 Commissioning Software," EPAC 2004.
[7] J. Safranek et al, "SPEAR 3 Commissioning," APAC 2004.
[8] J. Corbett, A. Terebilo, G. Portmann, "Accelerator Control Middle Layer," PAC 2003.
[9] A. Terebilo, "Simulated Commissioning of SPEAR 3," PAC 2003.

