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 a matrix manipulation language originally developed to be a convenient language for using the LINPACK and EISPACK libraries. What makes Matlab so appealing for accelerator physics is the combination of a matrix oriented programming language, an active workspace for system variables, powerful graphics capability, built-in math libraries, and platform independence. A number of software toolboxes for accelerators have been written in Matlab -- the Accelerator Toolbox (AT) for machine simulations, LOCO for accelerator calibration, Matlab Channel Access Toolbox (MCA) for EPICS connections, and the Middle Layer. This paper will describe the "middle layer" software toolbox that resides between the high-level control applications and the low-level accelerator control system. This software was a collaborative effort between ALS (LBNL) and SPEAR3 (SSRL) but easily ports to other machines. Five accelerators presently use this software. The high-level Middle Layer functionality includes energy ramp, configuration control (save/restore), global orbit correction, local photon beam steering, insertion device compensation, beam-based alignment, tune correction, response matrix measurement, and script-based programs for machine physics studies.

INTRODUCTION

Matlab started to be used at the ALS in the early 1990's shortly after commissioning. In the beginning it was primarily used as a scripting language for machine physics shifts but since the reliability was very good Matlab was used in operations as well. Present, Matlab is used for storage ring operations (energy ramp, configuration save/restore, global orbit correction, slow orbit feedback, and insertion device orbit and tune compensation) as well as machine setup (beam-based alignment, tune correction, chromaticity correction, response matrix measurement, LOCO, local photon beam steering, etc.) and as a scripting language for machine physics studies. The method to connection Matlab to the control system has changed over the years. Presently, the ALS uses EPICS and a version of simple channel access (SCAIII). For algorithm development the combination of a matrix oriented programming language, an active workspace for system variables, powerful graphics capability, built-in math libraries, and platform independence is quite desirable. The fact that Matlab is reliable enough to run operational code allows one to develop and deliver algorithms using the same language.

At SSRL, parallel Matlab developments in the late 90's led to the Accelerator Toolbox (AT) for machine simulations [8], Matlab Channel Access Toolbox (MCA) for EPICS connections [9], and LOCO for accelerator calibration, [10,11]. In a collaborative effort between ALS and SSRL, many of the control functions developed at the ALS were ported to SSRL, re-structured to incorporate MCA, and made machine independent. As a result, the methodology and structure of the control routines and functions can now be easily ported to other machines. The resulting "Middle Layer" software simplifies application program development, buffers the user from the details of EPICS, and hides cumbersome control system channel names. The naming scheme used by the Middle Layer has been adopted from accelerator tracking codes.

SOFTWARE OVERVIEW

As shown in Fig. 1 the Middle Layer software provides a library of functions that access either the machine hardware via EPICS (MCA, LabCA, or SCAIII) or the AT simulator. It can also connect to a remote AT simulator serving Channel Access, [12]. The ability to switch between online and simulate modes is helpful for analysis and debugging prior to deployment on am operating accelerator. One of the fundamental purposes of the Middle Layer is to simplify the hardware channel naming scheme used by the control system. Channel names are often quite obtuse so it is best not burden too many people with deciphering what names goes with what piece of hardware. The Middle Layer organizes channel names into groups (families), subgroups (fields), and devices. The Middle Layer tries to mimic naming schemes commonly used in particle tracking codes. Several naming options are provided for programming flexibility. Hence, the same language or terminology of accelerator tracking codes can be used to communicate with the online accelerator.

At the heart of the Middle Layer is a data structure containing the necessary information to setup the mapping from the Family/Device syntax to the control system hardware. The Matlab structure has been named the Accelerator Object (AO). The AO contains attributes for each Family (device list, channel names, etc.), hardware-to-physics conversion factors, range information, etc. The AO resides in a hidden memory location for application data associated with the Matlab command window. A parallel structure, called Accelerator Data (AD), contains directory locations, file names, and basic accelerator parameters. The Accelerator Data structure also resides in the application data location of the command window. Running the Matlab command *aoinit* will load both structures for access from the Middle Layer. The details of how to setup the AO and AD structures can be found in the Matlab Middle Layer Manual, [1].

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

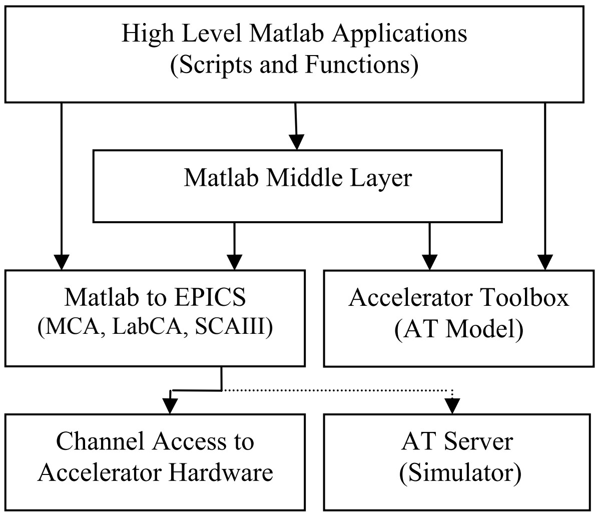


Fig. 1. Software Flow Diagram.

MIDDLE LAYER NOMENCLATURE

In the EPICS environment each hardware device is referenced via a 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, each family is represented by a structure with a nominal set of 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 structures into *Monitor* and *Setpoint* sub-structures keeps element attributes well organized and fits neatly into the middle layer function architecture. The EPICS setpoint channel 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. Wherever possible, functions are written in machine-independent format so that the machine dependence can reside completely in the AO and AD structures.

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 over ethernet. The servers can be connected to live hardware modules or a model server [12]. The *simulation* mode directs get/set calls directly to the local AT model. 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 (e.g. special unit conversion procedures).

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 Channel Access Servers or the MATLAB Accelerator Toolbox. The two main functions are *getpv* (*get* EPICS process variable) and *setpv* (*set* EPICS process variable). Both functions accept a variety of input formats via the Family/DevieList convention. Rather general calls are permitted and timing requests are possible.

Utility Functions

Utility functions allow easy conversion between fields in an *Accelerator Object* family. Examples include family2channel (convert family/Device to channel names). *getfamilydata* is a particularly important utility function used to access any information contained in the *Accelerator Object* families.

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*, and *getx/gety* which return horizontal and vertical beam position values. Shortcut functions are used widely in application development and for script development during machine physics studies.

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 (below) shows the conversion algorithm and associated parameters. Keeping track of the magnet hysteresis loops can be cumbersome and is usually machine dependent. For instance, at SPEAR3 polynomial current-to-field transfer functions based on magnet measurements are used. These details get programmed into the amp2k and k2amp conversion functions. The software user decides what unit to operate the Matlab session in – *switch2hw* or *switch2physics*. The units can also be controlled by an optional override keyword on the input line in functions calls such as *getpv*.

Simulator Functions

These functions communicate directly with the AT model to return simulated physics parameters. For example, *getbeta* calculates the beta function of the mode. Many Middle Layer functions allow for a model override. For instance, m*easbpmresp('model')* and *meastuneresp('model')* will compute the BPM and Tune response matrices of the model where as *getbpmresp* and *gettuneresp* will return the default saved matrices. These functions are quite useful during commissioning or when the working point of the accelerator is changed. All Matlab function in the AT toolbox are also available.

Special Functions

Some devices do not conform neatly with the Family/Index formalism so special functions are created to access the data. For example, the tune could be a typical Accelerator Object family using channel names or it could be a special function getting data via a totally different mechanism (e.g., a GPIB connection to a signal analyzer).

DATA MANAGEMENT

Data management for measurement and control can be a challenging task. As describe above, the *Accelerator Objects* framework organizes element names and attributes in a local database. The AO can be loaded from a text file or created from a master database of the accelerator control system. The *Accelerator Data* structure contains the file organization scheme, machine- and Middle Layer specific data. Examples include calculated physics parameters (e.g., momentum compaction factor) and directory locations to store measured data. Another important aspect of the Middle Layer is that it provides a framework for saving operational files, like lattice saves, golden orbits, BPM gain errors, response matrices, etc.

APPLICATION PROGRAMS

A primary reason for middle layer software is to simplify script construction for machine studies and high-level application programming for machine control. 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 and greatly simplifies communication with the 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

EPICS OR NOT

The Middle Layer typically works with EPICS, however, this is certainly not a requirement. There are only two core functions that need to be reprogrammed to work with a different control system protocol — getpvonline and setpvonline. This was done without much effort at Brookhaven for the X-ray and VUV rings.

PERFORMANCE

The question of speed often gets asked when discussing Matlab or any interpreted language. In general, Matlab is not usually the bottleneck. Usually waiting for the hardware to reach a setpoint is the time consuming task. For instance, waiting for a power supplies to ramp to a setpoint. For this reason, a series of "WaitFlags" are included in the Middle Layer communication routines for careful time sequencing of an experiment. Acquiring data is usually quite fast. At the ALS, or instance, one can acquire 120 BPMs at about 100 Hz with Matlab.

SUMMARY

The various Matlab toolboxes written for accelerator physics applications (Middle Layer, AT, MCA, LOCO, etc) are well integrated and have proven to be quite useful for machine studies and control at several operating machines. The relatively user-friendly software and almost 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 machine development studies with minimal training. To date, the software has been installed on five machines (ALS, CLS, SPEAR3, NSLS VUV and X-ray rings) and has received large interest from other laboratories. The Australian light source, DIAMOND, Soleil, and ALBA all have plans to use the Middle Layer software at some level. In principle the Middle Layer can be applied to linear accelerators and transport lines as well.

The Middle Layer does take some upfront effort and expertise to install on a new accelerator. Typically, one of the authors will work with a laboratory during the installation phase. The software can be made functional for most applications within a few days and fully operational in a few weeks. Developing a fully calibrated online model (magnet hysteresis, calibration factors, BPM errors, etc) is the most time consuming part of the software 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 inexpensive and fast way to acquire high level control and simulation software that has also been thoroughly tested on other machines. Also, software development is not only expensive from a labor point of view, it is very expensive to test and commission new software from a beam time perspective.
- Having one software package that is debugged at many laboratories improves reliability. Thousands of dedicated accelerator hours have been spent testing, improving, debugging, and exercising the Middle Layer/MCA/AT software packages.
- As with the EPICS collaboration, software expansion, suggestions, and new ideas come from a bigger pool of people.

- The number of physicists and engineers trained on the Middle Layer is growing rapidly. That way visiting scientists can work immediately on the new accelerator with very little hand-holding. This was very useful for commissioning SPEAR3.
- Since it's easy to switch between different accelerators in a simulated mode, it's easy to test algorithms on different accelerators. This is often an informative procedure when developing new algorithms. Writing software in a machine independent way is somewhat more time consuming but the final product tends to be better written and more robust. Orbit correction, beam based alignment of quadrupoles, and LOCO are examples of algorithms where effort was spend to make them machine independent.

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. And we would like to thank them in advanced for any future software developments that organically emerge from the collaborations. We are especially thankful for the free flow of ideas between all the physicists and engineers that we have worked with.

REFERENCES

[1] G. Portmann, J. Corbett, A. Terebilo, "Middle Layer Software Manual for Accelerator Physics," LBNL Internal Report, LSAP-302, 2005.

[2] J. Corbett, A. Terebilo, G. Portmann, "Accelerator Control Middle Layer," PAC 2003.

[3] J. Corbett, G. Portmann, A. Terebilo, J. Safranek, "SPEAR 3 Commissioning Software" Proc. of EPAC 2004.

[4] G. Portmann, J. Corbett and A. Terebilo, "Middle Layer Software for Accelerator Control." SSRL Memo, March, 2004.

[5] G. Portmann, "Slow orbit feedback at the ALS using MATLAB", PAC 1999.

[6] J. Safranek, et. al., "Spear 3 Commissioning," Proc. of APAC 2004.

[7] G. Portmann, "ALS Storage Ring Setup and Control Using Matlab." LBNL LSAP Note #248, 1998.

[8] A. Terebilo, "Channel Access Toolbox for Matlab," ICALEPCS 2001, San Jose, CA.

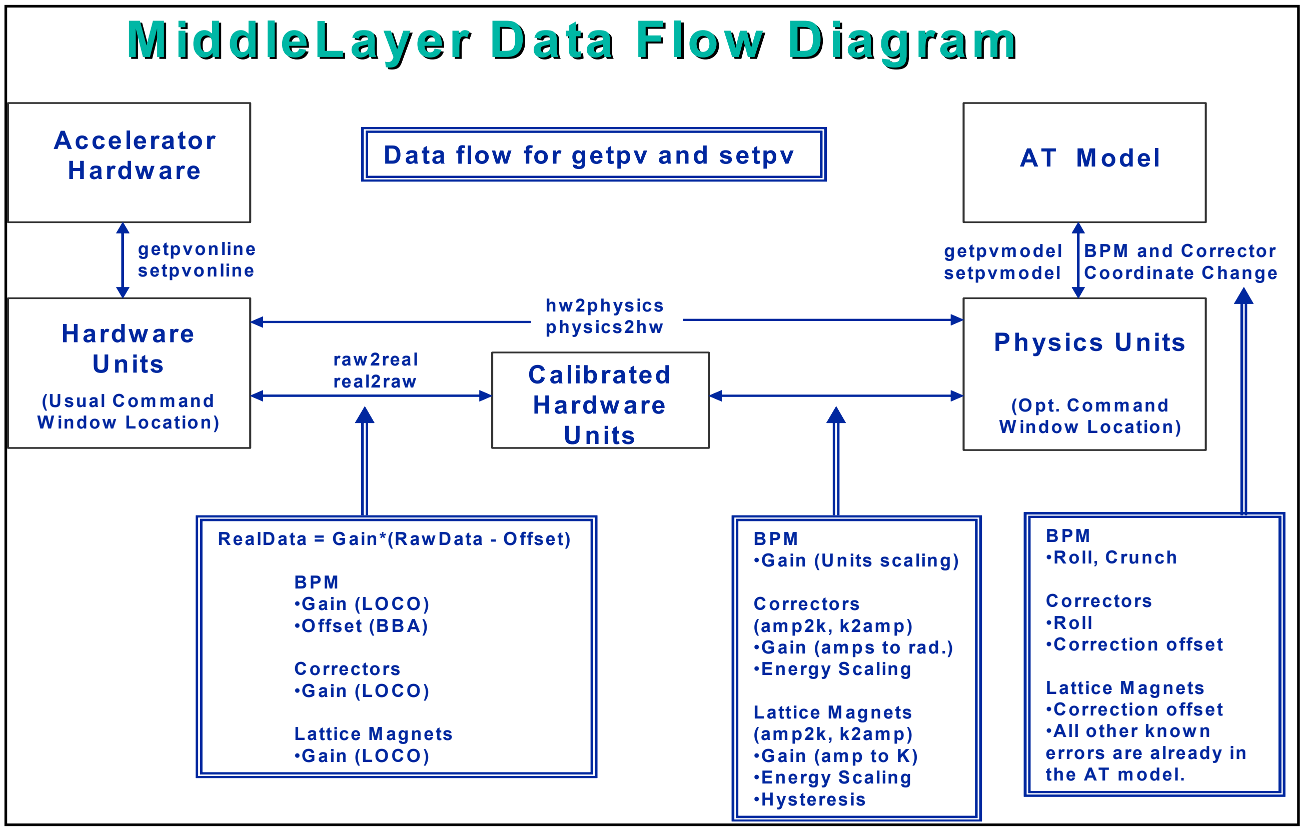
[9] A. Terebilo "Accelerator Modeling with MATLAB Accelerator Toolbox," PAC'01, May 2002, pg. 3203.

[10] J. Safranek, et al., "Linear Optic Correction Algorithm in MATLAB," PAC 2003.

[11] J. Safranek, G. Portmann, A. Terebilo and C. Steier, "Matlab Based LOCO." Proc. of EPAC 2002, p. 1184.

[12] A. Terebilo, "Simulated Commissioning of SPEAR 3," PAC 2003.

[13] J. Corbett, et al., "Orbit Control Using MATLAB," PAC'01, Chicago, May 2002, pg. 813.


MiddleLayer Data Flow Diagram
Accelerator Hardware
Data flow for getpv and setpv
AT Model
getpvonline
setpvonline
getpvmodel
setpvmodel
BPM and Corrector Coordinate Change
hw2physics
physics2hw
Hardware Units
(Usual Command Window Location)
raw2real
real2raw
Calibrated Hardware Units
Physics Units
(Opt. Command Window Location)
RealData = Gain*(RawData - Offset)
BPM
•Gain (LOCO)
•Offset (BBA)
Correctors
•Gain (LOCO)
Lattice Magnets
•Gain (LOCO)
BPM
•Gain (Units scaling)
Correctors
(amp2k, k2amp)
•Gain (amps to rad.)
•Energy Scaling
Lattice Magnets
(amp2k, k2amp)
•Gain (amp to K)
•Energy Scaling
•Hysteresis
BPM
•Roll, Crunch
Correctors
•Roll
•Correction offset
Lattice Magnets
•Correction offset
•All other known errors are already in the AT model.