SPEAR 3 COMMISSIONING SOFTWARE[§]

J. Corbett, G. Portmann, J. Safranek and A. Terebilo

Stanford Linear Accelerator Center/Stanford Synchrotron Radiation Laboratory
Stanford, CA 94309, USA

Abstract

The short SPEAR 3 startup time required pre-commissioned software for machine setup, beam measurements and data analysis. To accomplish this goal, we used Matlab with the Accelerator Toolbox (AT), the Channel Access Toolbox (MCA) and Middle Layer software to integrate code and streamline production. This paper outlines the software architecture, describes the Middle Layer component and provides examples from SPEAR 3 commissioning.

INTRODUCTION

The SPEAR 3 upgrade required a re-write of the control software with little opportunity to test with live beam [1]. For the physics part, we looked for a programming language with strong analytical and graphical capabilities, and easy-to-use syntax for non-computer scientists [2]. The search resulted in Matlab, which was in already in use at SLAC/SSRL for accelerator control, machine simulation and data processing, and at the ALS for direct machine control [3].

Specifically, Matlab provides a combination of an interpretive matrix-oriented programming language with powerful graphics capability, built-in math libraries and platform independence to give a high degree of flexibility.

As part of the SPEAR 3 project, a number of Matlab toolboxes were written for accelerator physics. The software suite includes the following:

1. AT – Accelerator Toolbox for simulation [4]
2. MCA and LabCA – Matlab-to-EPICS link [5]
3. Middle Layer – easy connection to AT, MCA plus machine measurement library [6]
4. LOCO (Linear Optics from Closed Orbits) [7]
5. AT Model Server (ATMS) [8]

The Accelerator Toolbox (AT) is a full 6-dimensional tracking code. Matlab Channel Access (MCA) provides a Matlab link to the EPICS channel-access library. The Middle Layer has three main functions: (1) to provide a method to group devices with a simple calling scheme for *get* and *set* operations, (2) to provide a library of common accelerator physics tasks and (3) to organize the data when running an accelerator. The ALS method of using Matlab for high-level software control provided the foundation for the Middle Layer [3].

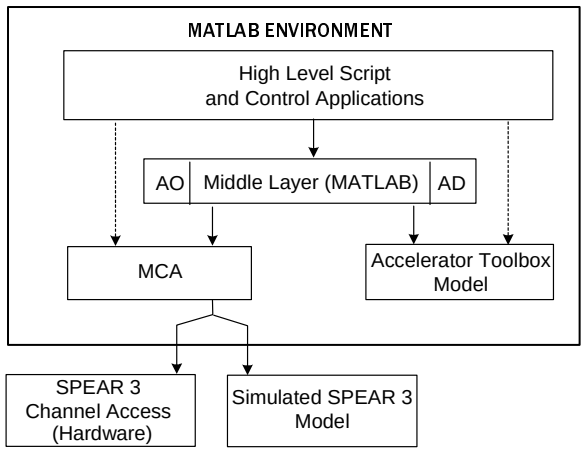


Figure 1: Software Architecture

An important part of the Matlab suite is LOCO (Linear Optics from Closed Orbits), a Matlab version of it's FORTRAN predecessor [9]. LOCO has been used world-wide to correct linear optics in storage rings and was a critical component in commissioning SPEAR 3.

Finally, the AT Model Server can be used to set up an EPICS ioc with Matlab running on a clock in the background to calculate orbits, tune, chromaticity, etc. Matlab software communicates with the EPICS ioc as usual but receives data calculated by the AT model.

Having all five software tools in a common programming language is both efficient and beneficial. Each tool can be used independently with the Middle Layer acting as a central hub. Figure 1 shows how top level applications access AT and MCA through the Middle Layer. The entire Middle Layer can be 'switched' between simulator and online mode, allowing the user to communicate with mode-independent syntax. For SPEAR 3, software scripts, functions and application programs were pre-tested in the simulation mode and switched seamlessly to ATMS well before hardware ioc's were available.

The architecture and interpretive code aspect of the software simplifies software development, machine control and data analysis. Furthermore, the complete package is largely machine-independent: connecting to a new accelerator requires a new set of magnet calibration factors, a machine-specific setup file and the AT model file.

[§]Work supported in part by the US Department of Energy Contract DE-AC03-76SF00515 and Office of Basic Energy Sciences, Division of Chemical Sciences.

MIDDLE LAYER SOFTWARE

The Middle Layer provides a flexible way to access accelerator variables. Function calls are typically formatted with machine-independent Family/Devicelist syntax to identify specific hardware elements. Each parameter type is organized into groups (Families), subgroups (Fields), and devices (Elements). The combination (QF, 'Setpoint', [3,1]), for instance, refers to the setpoint for the first QF magnet in sector 3. The same combination can be used to *get* or *set* variables either in the simulator or online.

The heart of the Middle Layer is a data structure containing Accelerator Objects (AO). The AO contains attributes for each Family (element indices, channel names, limits, etc). The AO families also contain hardware-to-physics unit conversion functions and associated conversion factors. The AO structure is loaded into Matlab memory for fast and easy access.

A parallel structure, Accelerator Data (AD), contains directory locations, file names and basic accelerator parameters. The AD also resides in Matlab memory.

Since Matlab is an interpretive language, the Middle Layer can be used directly from the command line, in short scripts or can be integrated into high-level application programs. All three techniques were used to commission SPEAR 3 and have been applied at the ALS, CLS and NSLS.

Get/Set Functions

The Get/Set functions communicate with the accelerator simulator or hardware (iocs). The two core functions, *getpv/setpv*, simplify MCA calls and to make details of the control system transparent. Both functions accept a variety of input formats including timing information, and keyword-driven options.

```
getpv('Family', 'Field', 'DeviceList', 'Time', 'Keywords')
setpv('Family', 'Field', 'Value', 'DeviceList', 'Keywords')
```

At the ALS, for example, one can turn on, reset, set ramp rate to 10 amp/sec, and ramp the QF-family setpoint to 80 amp with the following sequence:

```
setpv('QF', 'On', 1)
setpv('QF', 'Reset', 1)
setpv('QF', 'RampRate', 10)
setpv('QF', 'Setpoint', 80)
```

Alias functions to *getpv* and *setpv* are also often used:

```
getam –  get analog monitor values
getsp/setsp –  get/set setpoint values
stepsp –  step setpoint values
```

The user can also access PV names that do not reside in the AO database. Getpv('PVName') is a valid call.

Modes of Operation

The middle layer permits several modes of operation for each family. The *simulator/online/manual* mode allows the user to communicate with the local AT model, external hardware, or manually input changes from the keyboard. A mode override is possible on any function by putting the keyword '*simulator*', '*physics*', or '*manual*' in the function call. This feature is often used to get response matrices from the model while working online.

Another useful Middle Layer feature is to establish different *operational modes*. By specifying directory paths and filenames in the AD, the Middle Layer can be set to operate in a different accelerator configuration, for instance, to separate developmental machine studies from the day-to-day operational configuration.

Physical Units and Unit Conversion

Accelerator modeling and accelerator control often use different units with complicated unit conversion functions. The control system typically operates in *hardware* units (e.g. bit count, amp) whereas the accelerator model uses *physics* units (e.g. k-value, radian). To update the model with the present *online* values, magnet currents are converted to k-values with the *hw2physics* function. Such conversions are often used for model calibration or to compute the electron beam optics.

Conversely, model-based calculations are downloaded to the machine using *physics2hw*. For SPEAR 3, the magnet conversions are polynomial functions but in general arbitrary functions with arbitrary coefficients are possible. Calibration errors detected by LOCO can also be taken into account in these functions.

The Middle Layer can operate in either *physics* or *hardware* units. And one can override the units on any function by putting the keyword '*physics*' or '*hardware*' in the function call. *getpv* and *setpv* perform unit conversions automatically depending on mode of operation.

Data Management

There is a large amount of data that needs to be maintained to automate and optimize the running of an accelerator. The AO and AD data structures (already discussed) contain data which is accessed frequently but does not change very often – like EPICS channel names. Data that occasionally changes and does not get accessed often is stored in a special physics data file. Typical data stored in this file are golden/offset orbits, BPM gains, power supply gains/offsets, and coupling coefficients. Data which is specific to a particular accelerator state is maintained in separate files. Lattice saves, response matrices, and insertion device compensation tables fall into this category.

Many Middle Layer functions also generate data. The data is automatically organized in a user-specified directory tree and stored in a data structures which define how and when the data was measured.

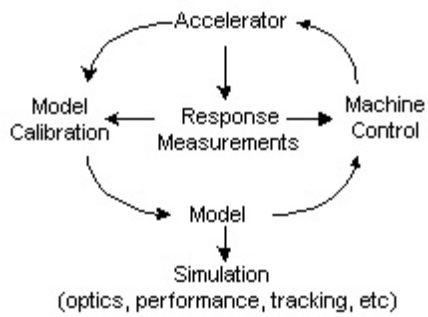
HIGH-LEVEL SOFTWARE

High-level software and applications can use Middle Layer functions to access to the accelerator hardware and model. The resulting applications are largely accelerator-independent and can be shared between laboratories. Examples of shared high-level applications are configuration save/restore, beam-based alignment, global orbit feedback and control [10,11], variable monitoring GUIs, aperture scans, tune and chromaticity control, energy ramping, insertion device compensation and dynamic aperture scans.

Many high level programs use a mix of pure Matlab and Middle Layer functions. LOCO, for example, can accept input data generated with Middle Layer measurements, processes the data in Matlab based on user-defined setup files, and apply the result to the machine with Middle Layer functions. Even more advanced data acquisition and analysis routines are progressively integrating into the Matlab softare [12].

ACCELERATOR MODELING

Control systems at SLAC have a history of incorporating the accelerator model into the on-line software. The basic philosophy is simple: convert power supply currents to magnet k-values to produce the machine model, and convert k-values to supply currents to control beam optics. In recent years, response matrix measurements have been used to *calibrate* the model to the machine and vise-versa (LOCO). The MATLAB software suite enhances the coupling between machine and model, allows easy extraction of data for post-processing and increases programming flexibility in both domains.



SUMMARY

The accelerator physics software suite developed for SPEAR 3 proved successful in simulation and in the control room. The user-friendly software and machine-independent library 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 minimum training on the host system. To date, the software has been installed on five machines (ALS, CLS, SPEAR, NSLS VUV and X-ray rings) and has received large interest from other laboratories. The Australian light source, for instance, is planning to further expand the applications library.

For a new machine, the system can be made functional within a few days and fully operational in a few weeks. Developing a fully calibrated online model (magnet calibration factors, BPM errors, etc) is the most time consuming part of the software setup.

ACKNOWLEDGEMENTS

The authors would like to thank the ALS accelerator physics staff for a productive collaboration on LOCO and the Middle Layer. This collaboration helped SPEAR 3 commissioning exceed expectations. Thanks to M. Yoon for many contributions and the CLS collaboration which led to several important software developments.

REFERENCES

[1] R. Hettel, et al, "SPEAR 3 Upgrade Project: The Final Year." Proc. of PAC, 2003, p. 235.
[2] J. Corbett, et al, Satellite meeting on Accelerator Physics Software for SPEAR 3, ICAP, 1998.
[3] G. Portmann, "ALS Storage Ring Setup and Control Using Matlab." LBL LSAP Note #248, 1998.
[4] A. Terebilo, "Accelerator Modeling with Matlab Accelerator Toolbox", Proc. of PAC, 2003, p. 3203.
[5] A. Terebilo, "Channel Access Toolbox for Matlab ", ICALEPCS 2001, San Jose, CA
[6] G. Portmann, J. Corbett and A. Terebilo, "Middle Layer Software for Accelerator Control." SSRL Memo, March, 2004.
[7] J. Safranek, G. Portmann, A. Terebilo and C. Steier, "Matlab Based LOCO." Proc. of EPAC 2002, p. 1184.
[8] A. Terebilo, et al, "Simulated Commissioning of SPEAR 3 Storage Ring", Proc. of PAC, 2003, p. 2372.
[9] J. Safranek, "Experimental determination of storage ring optics using orbit response measurements", Nuc. Instr. Meth., 388, 27 (1997).
[10] J. Corbett and A. Terebilo, "Interactive Orbit Control Program in Matlab." Proc. of PAC, 2001, p. 813.
[11] G. Portmann, "Slow Orbit Feedback at the ALS Using Matlab," Proc. of PAC, 2003, p. 2373.
[12] C. Steier, D. Robiin, L. Nadolski, W. Decking, Y. Wu and J. Laskar, "Measuring and Optimizing the Momemtum Aperture in a Particle Accelerator." Phys Rev. E., 65 056506 (May 2002).