Recent Developments for the MX Beamline Control Toolkit

W. M. Lavender
Biological, Chemical, and Physical Sciences Department,
Illinois Institute of Technology, Chicago, Illinois 60616

(Dated: October 15, 2004)

This report discusses recent progress with MX, a data acquisition and control system with over 350 device drivers that has been under development for 9 years. MX is designed to be easily retargeted to alternative hardware or operating systems and has long had the ability to call out as a client to external control systems. We now report progress in the other direction of making MX usable as a server component for other control systems. The goal here is to make it easy to reuse MX's large driver package in other control systems. We have striven to make it easy to write device drivers for MX, so that beamline staff and graduate students can easily do the work. In addition, the new server support will make drivers written this way immediately usable by multiple control systems. The initial effort is to create EPICS and Spec servers that use the MX driver library to control the hardware. We also report progress in other efforts, such as performance work to make MX as fast and efficient as possible, both internally and in its connections to external control systems. Support has been added for CCD camera control and robotic sample changers for crystallography. There is also new support for MODBUS and USB interfaces, for beam position monitors, and new enhancements to support for XIA multichannel analyzers. Other improvements include work specific to Delta Tau PMAC motor controllers that allows a PMAC coordinate system to be used as an MX pseudomotor. On the user interface side, work is underway for writing user interfaces for a variety of synchrotron radiation experiments. These include a new tabbed format crystallography GUI being developed by SER-CAT (Univ. of Georgia) and new GUIs for microfocussing, XAFS, and lithography experiments. In addition, a graphical editor for editing MX databases has been developed.

I. INTRODUCTION

MX is a portable beamline control and data acquisition toolkit that has been reported on at previous NOBUGS conferences [1] [2] [3] and elsewhere [4]. MX is being jointly developed by the MR-CAT[5][6] IMCA-CAT[7][8][9] and SER-CAT[10] sectors at the Advanced Photon Source. It is currently in use at 9 beamlines at the Advanced Photon Source and CAMD. MX is written in ANSI C and it has scripting interfaces available for Python and Tcl. MX has an extensive set of device drivers with 68 motor and pseudomotor drivers and with over 380 drivers altogether. The core of MX is designed to be middleware, so it tries to be as independent of the details of both the hardware it is communicating with and the detailed nature of the servers and application programs that are invoking it. As part of this, it also tries to make all drivers for a given class of device look as much alike as is practical. The intention is that drivers writers need not know all that much about MX to get a working driver. MX is also designed to be easy to interface to other control system drivers and to be easy to embed in the applications and servers belonging to other control systems. This paper reports on recent developments with MX and plans for the near future. The MX homepage may be found at `http://mx.iit.edu/`. At the time of the writing of this paper, the current version of MX is version 1.0.2.

II. EXPORTING MX TO OTHER CONTROL SYSTEMS

MX was originally designed with four roles in mind:

- As a portable toolkit for writing data acquisition and control systems.
- As a standalone system capable of controlling entire experiments.
- As a platform for building device control servers to be used by other applications.
- As a way of extending other control systems and of glueing disparate control systems together

Of these, the first two were accomplished quite some time ago. Until recently, the last two items have only been wishlist items for the future. This has now changed, since work has begun on the development of MX based servers for other control systems.

A. External Server Interface

The interface between MX and an external control system server is handled as follows. Each network accessible value is exported by a data structure called an MX_RECORD_FIELD, which contains a pointer to the value of the field. If an external server is attempting to write to an MX record field, all it need do is copy the field's new value to the data pointer

and then invoke *mx_process_record_field()* for that field. *mx_process_record_field()* will then figure out the correct driver to invoke and then report back the status of the operation so that the server can report this to the client. Reading from an MX record field structure is similar except that the server is expected to read the returned value after *mx_process_record_field()* is run. After some additional restructuring is complete, MX will handle callbacks in a similar manner.

The central idea here is that as long as the external server code writes the value to the correct location, MX explicitly does not care how the value got there. In particular, MX does not assume the use of a particular interprocess method or protocol.

B. EPICS

For a number of reasons we have chosen first to export the MX toolkit to EPICS [11] via the portable Channel Access Server. The portable Channel Access Server is an alternative to EPICS iocCore that implements a network interface that looks like a Channel Access server, but does not implement the underlying functionality of a data acquisition server. Since MX already implements most of the underlying features needed for such a server like experiment control logic, device drivers and so forth, the two packages seem like a good match. This new work is being implemented using MX 1.1.0 which is expected to be released before the end of 2004. For MX 1.1.0, the event handlers and processing logic originally found in the MX server have been abstracted out and move to the main MX library, *libMx*, so that it can be used in other servers or event-driven clients.

The initial development of the MX Channel Access server focuses on exporting MX motor drivers to EPICS. The MX Channel Access server does this by eumulating the process variables of the EPICS motor record developed at the Advanced Photon Source. The emulation will not be perfect since there are some EPICS motor process variables that have no meaning in an MX context, just as there are MX motor driver features that have no counterpart in the APS EPICS motor record.

At present, the MX Channel Access server has progressed far enough that MX controlled motors can be moved via standard EPICS MEDM screens and via MX's client-side *epics_motor* record. Future development in the near term will include making sure that the MX Channel Access motor record can be used by Spec's EPICS motor support. Then, the focus will shift to implementing the rest of the EPICS motor PVs that make sense in an MX environment.

The long term goal for MX Channel Access support is to export as much of the MX API as Channel Access PVs as is practical. When possible, MX will try to emulate existing EPICS record types.

C. Spec

One of the author's favorite sayings is that there is no such thing as portable software, just software that has been ported. When new features are implemented in MX, we generally try to implement more than one of them so that the interface developed is not so dependent on the idiosyncracies of the first implementation.

Thus, soon after finishing the external server for EPICS, another external server interface will be developed which will probably be a remote Spec [12] server. MX has recently implemented client-side drivers for connecting to a remote Spec server, so much of the necessary software has already been written. The initial MX Spec server will export MX motor, scaler, and timer records since that is what is currently documented by Spec's author.

Note that the MX Spec server will only be attempting to emulate a subset of the remote Spec server functionality. The vendor's remote Spec server has a Spec command line interface which is accessible to clients. Reimplementing all of the remote Spec server's command line functionality would be an enormous amount of work with no obvious benefit, so we have no plans to do that. All the MX Spec server will aspire to do is to provide MX device support to Spec clients.

Support for exporting MX drivers beyond EPICS and Spec will be done as timer permits and the need arises. A couple of systems that we have envisioned doing this for are TACO [13] and Blu Ice [14] . However, we cannot currently promise when support for these other protocols will be implemented.

D. Multiprotocol Servers

Another server related feature that will be added in the near future will be multiprotocol servers. An example of the need for this would be a situation where some clients use MX protocols and other use EPICS protocols. Since MX servers can also act as MX clients, one could envision handling this by having an MX Channel Access server that talked as a client to an MX server. Or one could have an MX server that talked to a client to an EPICS server. However, either way of implementing this would result in one of the two types of clients having to suffer with their commands being sent over two network hops rather than one. Since network I/O can be expensive in terms of time, MX plans to add support for multiprotocol servers.

Thus, in the example above, you would now have a single process that would simultaneously support both EPICS and MX protocols. So far, it appears that it should be possible to make the MX server, EPICS Channel Access server, Spec server, and Blu Ice device hardware server live alongside each other in the same process with the top level event loop processing each type of event in turn.

Note that we do not plan to abandon the MX network protocol in the forseeable future for one of the other protocols. Having our own protocol means that features can be added to it as necessary. Also, we are able to directly work on optimizing the speed of the protocol without having to worry about creating an incompatible variant of someone else's protocol. In addition, we are able to port the high level MX protocol to alternate low level transports at will as will be discussed later.

III. PERFORMANCE OPTIMIZATION

Initial development of MX focused on the development of functionality, plus correctness of behavior. However, now that MX is close to meeting its primary design goals, we are now working on MX performance improvements. As part of this we have developed a high resolution timing package that provides sub-microsecond timing for x86 Linux, Microsoft Windows, MacOS X, Solaris and Irix. Each of these platforms provides their own different way of implementing high resolution timing such as the Pentium RDTSC instruction, the Power PC timebase registers, and Microsoft Win32's *QueryPerformanceCounters()*, so it has been convenient to create a uniform wrapper for this. The high resolution timer package has made it easier to focus in on the performance implications of small sections and even single lines of code. Although there is still a lot of work remaining to be done on this issue, we have already found and eliminated two important performance bottlenecks in MX.

A. EZCA

The first major performance improvement related to the use by MX of the EZCA [15] package to interface with EPICS. EZCA, as its name implies, attempts to be an easier interface to EPICS than the raw underlying Channel Access protocol. EZCA does meet this goal fairly well and was used in versions of MX prior to MX 1.0 as the primary interface to EPICS. However, after benchmarking with the MX high resolution timer package, it appeared that the default configuration of EZCA is quite slow. When benchmarked at sector 17-BM of the Advanced Photon Source, it was found that EZCA took 60 milliseconds just to get the value from a scaler channel using an m68k based EPICS IOC, which is tremendously slow.

After initial investigation, it developed that the 60 millisecond interval was set by internal parameters of the EZCA package and that it was possible by changing those parameters to speed it up. However, investigation of the EZCA source code showed that it contained a much larger amount of code than we would have expected for the task it was performing. Since the use that MX makes of EPICS is fairly straightforward, it was decided that it would be better and simpler to remove the dependence on EZCA and have the MX EPICS drivers directly communicate with the Channel Access libraries.

This change has greatly speeded up MX's communication with EPICS. When tested with the same m68k based EPICS IOC, the 60 millisecond time shrank down to 1.5 milliseconds, which is a factor of 40 improvement. Ironically, the switch from a m68k IOC to a PowerPC IOC which was done around the same time only resulted in another factor of 3 improvement down to 0.5 milliseconds. The net result is that MX 1.0 now communicates with EPICS around two orders of magnitude faster than before.

B. MX Server

The other important protocol for MX clients is the MX network protocol used to communicate with MX servers. Some of the optimizations added in MX 1.1 have included switching from ASCII to binary data formats for network communication and the use of binary handles to represent MX record fields in remote servers rather than the previous practice of sending the ASCII name for each network call. However, the most important improvements so far have been in the MX server itself.

One important change was the removal of a delay that had been inserted into early versions of MX to make certain drivers work more reliably. The problem was that this delay was also slowing down other drivers that did not need the slowdown. The other change was that the MX server was queueing almost all client requests for later execution. Closer examination has shown that it is better to service many of these requests immediately and only queue requests for long running operations and for individual drivers that need a delay.

With these changes, internal time overhead in the MX server that sometimes were in the tens of millisecond range have been reduced to tens of microseconds. For example, for MX 1.1, a series of *get_position* requests by an MX client to an MX server using software emulated motor records now only takes around 75 microseconds per request on a 750 MHz Pentium III. Interprocess communication on the same computer can be even faster than that, so we have also been exploring the use of alternate transport mechanism for MX communication within a single computer. Initial testing with MX protocol running over Unix domain sockets reduced the 75 microsecond value above further down to around 55 microseconds, so there is indeed some room for improvement here. We plan to explore other single computer interprocess communication methods as well, although MX communication between computers will probably always be over TCP/IP.

C. Callback Support

At present, MX is largely a remote procedure call style system, using callbacks in only restricted situation. Since polling is often a source of inefficiency, we plan in MX 1.2 to improve the support for callbacks to the point where one could envision doing most MX network I/O through monitoring of record fields via callbacks. The callback system will be made general enough to be used by MX clients as well as MX servers to make it easier to write event-driven MX clients. The callback support will also be used for external servers such as the EPICS portable Channel Access Server, which have a big need for this type of functionality due to the use of event driven EPICS GUIs such as MEDM, which rely heavily on asynchronous monitor callbacks.

D. Additional Performance Testing

There are many other performance measurements that we plan to make in the near future. Some of these will revolve around particular devices that have large I/O requirements. One particular system we plan to focus on is the X-Ray Instrumentation Associates [16] MCA electronics for multielement detectors. MCAs for multielement detectors can generate large amounts of data, so one of our priorities will be to make this run as fast as possible.

IV. NEW MX DRIVERS

MX has been designed to make it relatively easy to write new drivers, so a large number of drivers have been added since the NOBUGS 2002 meeting.

A. Summary of New Drivers

Here are the highlights of the new drivers:

- Analog and Digital I/O:
  - Crossbow Technology CXTILT02 digital inclinometer
  - Linux *parport* digital I/O
  - MODBUS analog and digital I/O
  - Omega iSeries temperature and process controllers
- CCD:
  - CCD control via MarCCD remote mode
- Counters/Timers:
  - Spec scaler/timer
- Current Amplifiers:
  - Advanced Photon Source ADCMOD2
  - Oxford Danfysik IC PLUS and QBPM
  - UDT Tramp
- GPIB:
  - IOtech Micro488EX
- MODBUS:
  - MODBUS/TCP (tested with Wago 750)
- Motors and Pseudomotors:
  - Aerotech Unidex 500
  - Animatics SmartMotor
  - Delta Tau PMAC coordinate system axis
  - Intelligent Motion Systems MDrive
  - McLennan PM600
  - Oceaneering Space Systems μ-glide
  - Oxford Cryosystems 600 series cryostream controller
  - Oxford Instruments ITC500 temperature controller
  - Phidget stepper motor
  - Spec motor
- Multichannel Encoders (used for quick scans):
  - Delta Tau PMAC multichannel encoder
- Multimeters
  - Keithley 2400 series
  - Keithley 2700 series
- Relays/Filters/Shutters:
  - X-Ray Instrumentation Associates PFCU filter/shutter controller
- RS-232:
  - Wago 750 RS-232 interface
- Sample Changers:
  - ALS style sample changing robot
- Scans:
  - Pseudomotor step scan
- Single Channel Analyzers:
  - Oxford Danfysik Cyberstar X1000
- USB:
  - Linux libusb interface
- VME:
  - SIS1100/3100 PCI-to-VME bus interface

A couple of these will be discussed further below.

B. CCD Camera Control

It has become clear that many users do not want to control some components of a beamline with one style of user interface, while other parts are controlled by other styles of user interfaces. For this reason, there is a substantial degree of interest in merging the entire user interface of a beamline into one GUI. On macromolecular crystallography beamlines, the system most affected by this is CCD camera control. CCD cameras from a commercial vendor generally come with their own custom user interface that is specific to their systems. Thus, in order to have a single interface, it is necessary to either merge beamline control into the CCD camera GUI or merge CCD camera control into the beamline control GUI.

SER-CAT at the Advanced Photon Source has chosen to integrate CCD camera control into their beamline control GUI. This has required, in turn, that MX provide support for controlling the CCD camera. At present, all of the CCD cameras at SER-CAT are sold by MarUSA [17] which means that only one driver need be written. This driver run from a copy of *mxserver* that is started by a beamline staff member or user by selecting remote operation from the MarCCD user interface. The *remote_marccd* driver makes use of the remote mode documented by MarUSA. Although the beamline user does not need to use the MarCCD user interface after the MX server is started, it is necessary that the MarCCD user interface be running somewhere.

C. Sample Changing Robot

The ability to change samples under remote control has now become very important to high throughput crystallography experiments. SER-CAT is currently in the process of installing a new sample changing robot based on a design from the Advanced Light Source [18] . This has prompted the development of a new MX device class for sample changing robot control which has been successfully tested offline with the new robot. The current plan is to integrate robot control into the new SER-CAT GUI in the near future.

V. GRAPHICAL USER INTERFACE DEVELOPMENTS

A variety of new MX-based graphical user interfaces are currently in use or under development for both macromolecular crystallography and materials science experiments. A recurring theme in new crystallography GUI development has been the use of the tabbed notebook style of user interfaces popularized by the Blu Ice system from SSRL. Another trend is that new MX GUIs are tending to be developed in Python rather than Tcl/Tk, since Python is perceived by many to be easier to understand. We will discuss some of this in further detail below.

A. Sergui

James Fait of SER-CAT is currently developing a new WxPython-based MX crystallography interface called *sergui*, some of the features of which have been touched on in the previous section. This package is a very ambitious and comprehensive user interface and is described in much greater detail in another talk [19] at this conference.

B. Imcagui

The existing Tcl/Tk-based *imcagui* interface used at IMCA-CAT has now been revised to make use of the tabbed notebook format. The functionality of this new version is largely the same as the old multiwindow interface, but is perceived as being more user friendly than the previous format. This remodeling of the user interface has shown the virtue of the use of high level GUI packages such as Tcl/Tk, [incr Tcl], and Iwidgets since the amount of work required to implement this change was relatively small. Our judgement is that this would have been true as well if the original package had been based on Python.

C. JDirector

IMCA-CAT has also implemented a new Java-based GUI called JDirector [20] that has been designed and developed by Rigaku/MSC. The main interaction of this system with MX is that JDirector's beamline control interface will soon be switching to a new Java interface to MX that is to be written during the upcoming winter. A Java interface to MX will also be valuable since many new graduates from computer science programs are currently most familiar with Java.

D. Material Science GUIs

Ken McIvor of IIT is currently developing a set of new WxPython-based materials science GUIs for MR-CAT. A new GUI for control of an X-ray lithography system has been in use for some time now at APS sector 10-BM, with GUIs for XAFS and microfocussing under development as well. In addition, Ken has developed a new MX database editor that is designed to eliminate the need for beamline staff to understand the format of MX database files in great detail. We are looking forward to new developments by him.

VI. CONCLUSIONS

It is clear that MX has made a substantial amount of progress since the last NOBUGS meeting in 2002. A variety of new applications and drivers are now available and the performance of the newest releases of MX are significantly higher than before. We also look forward to making the large package of MX drivers available for other control systems such as EPICS and Spec.

The author wishes to acknowledge the help and support of many people in the development of MX. In particular, I would like to acknowledge Lisa Keefe and Irina Koshelev of IMCA-CAT, Carlo Segre, Bruce Bunker, Jeff Terry, and Jeremy Kropf of MR-CAT, John Chrzas, James Fait, and Gerd Rosenbaum of SER-CAT, Andy Howard and Ken McIvor of IIT, Nadia Leyarovska of Argonne National Laboratory, and Henry Bellamy of CAMD.

I would also like to acknowledge the various organizations that helped fund this work. Use of the IMCA-CAT beamlines 17-ID and 17-BM at the Advanced Photon Source were supported by the companies of the Industrial Macromolecular Crystallography Association through a contract with Illinois Institute of Technology. Work performed at MRCAT is supported, in part by funding from the Department of Energy under grant number DE-FG02-04ER46106. Use was made of the Southeast Regional Collaborative Access Team (SER-CAT) 22-ID and 22-BM beamlines at the Advanced Photon Source, Argonne National Laboratory. Supporting institutions may be found at http://www.ser-cat.org/members.html. Use of the Advanced Photon Source was supported by the U. S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. W-31-109-Eng-38.

[1] W. M. Lavender, "MX: A Portable Toolkit for Data Acquisition and Control", NOBUGS 1997 workshop, Argonne National Laboratory (1997) at `http://www.aps.anl.gov/xfd/bcda/nobugs/welcome.new1.html`.

[2] W. M. Lavender, "MX: A Portable Toolkit for Distributed Data Acquisition and Control", NOBUGS 2000 workshop, Daresbury Laboratory (2000) at `http://srs.dl.ac.uk/nobugs/nobugs3/`.

[3] W. M. Lavender, "Materials Science and Protein Crystallography Using the MX Beamline Control Toolkit", NOBUGS 2002 workshop, National Institute of Standards and Technology (2002) at ???.

[4] W. M. Lavender, "MX: A Beamline Control System Toolkit", Synchrotron Radiation Instrumentation: Eleventh US National Conference, *AIP Conf. Proc.* **521**, 332 (2000).

[5] `http://ixs.csrri.iit.edu/mrcat/`

[6] C. U. Segre, N. E. Leyarovska, L. D. Chapman, W. M. Lavender, P. W. Plag, A. S. King, A. J. Kropf, B. A. Bunker, P. Dutta, R. S. Duran, J. Kaduk, "The MR-CAT Insertion Device Beamline at the Advanced Photon Source", Synchrotron Radiation Instrumentation: Eleventh US National Conference, *AIP Conf. Proc.* **521**, 419 (2000).

[7] `http://www.imca.aps.anl.gov/`

[8] J. F. Fait, W. M. Lavender, "Data Collection at the IMCA Beamlines", American Crystallographic Association Annual Meeting, 59 (1999).

[9] L. J. Keefe, J. Chrzas, J. L. Rios-Steiner, K. S. McCarthy, W. Lavender, K. J. Kim, A. J. Howard, "The User Program of the Industrial Macromolecular Crystallography Association CAT", American Crystallographic Association Annual Meeting, (2001).

[10] `http://www.ser-cat.org/`

[11] `http://www.aps.anl.gov/epics/`

[12] `http://www.certif.com/`

[13] `http://www.esrf.fr/???`

[14] `somewhereatSSRL`

[15] `somewhereatAPS`

[16] `http://www.xia.com/`

[17] `MarUSAreference`

[18] ALS robot reference.

[19] Sergui reference

[20] JDirector reference