Recent Developments for the MX Beamline Control Toolkit

William M. Lavender

Illinois Institute of Technology
Chicago, Illinois 60616

http://mx.iit.edu/
lavender@agni.phys.iit.edu

Plan of the Talk

- Review of MX
- Exporting MX to Other Control Systems
- Performance Optimization
- New Drivers
- Graphical User Interfaces

What is MX?

- A portable beamline control toolkit: *Linux, Solaris, Irix, Windows, MacOS X, Cygwin, etc.*
- Written in C, with Python and Tcl interfaces available.
- Designed as middleware.
- Comes with a set of servers and clients.
- Has an extensive set of device drivers:
  68 motor and pseudomotor drivers,
  with over 380 drivers altogether.
- MX servers and clients use the same set of drivers.
- Easy to interface to other people's drivers.
- Easy to embed in other applications and servers.

Current Usage of MX

MX is currently used at 9 beamlines at APS and CAMD.

- 3 beamlines use MX as a standalone control system.
- 6 beamlines use MX together with EPICS controlled devices.

So far, the primary usage pattern has been MX clients calling out to servers based on MX, EPICS, and other protocols.

However, we plan to change that.

Exporting MX to Other Control Systems

- Each network accessible value is stored in an MX record field object.
- When writing to a record field object, the foreign server should write to the object data pointer, and then invoke *mx_process_record_field().*
- When reading, invoke *mx_process_record_field()* and then read from the data pointer.
- MX record processing logic explicitly does not care how values are transferred to or from the object data pointer.
- This interface is designed to make it easy for foreign servers to use MX device drivers and processing logic.

MX Channel Access Server

- Uses the EPICS portable Channel Access Server for the interface to EPICS clients.
- Uses MX event handlers, processing logic, and device drivers for the hardware interface.
- Our goal is to emulate the PV interface for existing EPICS records when possible.
- The first version supports the use of MX-controlled motors via an APS EPICS motor record compatible interface.
- Has already been used to control MX motors from MEDM and the MX client-side *epics_motor* record.

MX Spec Server

- An easier way to make software interfaces versatile is by porting to more than one target at the beginning.
- MX's second foreign server target will implement the remote Spec protocol.
- MX already implements the client side of remote Spec protocol.
- With the client side done, most of the code to implement a remote Spec server is already written.
- MX will only act as a Spec device server and will not attempt to emulate the Spec command line interface.

Other Foreign Servers

- Additional protocols will be added as needed and time permits.
- TACO/TANGO? Blu Ice? ???
- The original MX protocol will continue to be developed as well, since having a protocol that we are free to modify as needed can be very useful.

Multiprotocol Servers

- If you have a mix of clients that use different protocols, a multiprotocol server is useful.
- Eliminates the need for an extra network hop where one server type needs to be a client for another server type.
- Appears to be relatively easy to do for most of the protocols discussed so far.
- A top level event loop services each type of client protocol in turn.

Performance Optimization

- MX originally focused on getting functionality correct.
- Most of the original design goals have now been met, so it is time to focus on making MX as fast as possible.
- A portable high resolution timing package with sub-microsecond resolution has been developed which runs on x86 Linux, Microsoft Windows, MacOS X, Solaris, and Irix.
- Profiling with programs like *gprof* is also very productive.

Performance – EZCA

- Intended as an “easy” Channel Access interface requiring less knowledge of how EPICS works.
- MX used EZCA up until MX 1.0.
- High resolution timing measurements show that the default configuration of EZCA is quite slow.
- *ezcaGet()* for an EPICS scaler channel in an m68k-based IOC had a roundtrip time of 60 milliseconds.

Performance – EZCA *(cont.)*

- Apparently it is possible to greatly speed up EZCA if you modify its default configuration.
- For MX 1.0, we decided that it was simpler to just remove EZCA and talk to Channel Access directly.
- With just that change, the 60 millisecond time mentioned above was reduced to 1.5 milliseconds (a factor of 40!).
- Replacing the m68k IOC with a PowerPC IOC improved it further to 0.5 milliseconds (only a factor of 3 though).
- MX now communicates with EPICS two orders of magnitude faster than before.

Performance – MX Server

- The MX server's performance has been significantly improved in MX 1.1 *(not yet released).*
- MX 1.1 has switched to binary data transfers.
- Switched to binary handles rather than transmitting ASCII record field names.
- Discovered that *mxserver* was often queueing events that really should have been handled immediately.
- Discovered that a delay introduced to make certain early drivers work correctly was unnecessarily slowing down other drivers that did not need the delay.

Performance – MX Server *(cont.)*

- The MX server in MX 1.1 now runs a couple of orders of magnitude faster than in previous versions of MX.
- A *get_position* command from an MX client to an MX server using software emulated motors now only takes around 75 microseconds on a 750 Mhz Pentium III.
- Individual MX drivers can now specify a minimum time between commands to the hardware controller.
- If the MX server and client are both on the same machine, faster transfers mechanisms than TCP/IP are possible.
- Running MX protocol over Unix domain sockets reduces the 75 microseconds above to 50 microseconds.

New MX Drivers

133 drivers have been added since the NOBUGS 2002 meeting for a current total of 386. Here are a few highlights:

- MODBUS and USB interfaces are now supported.
- SIS 1100/3100 PCI-to-VME interface.
- ALS style sample changing robot.
- CCD control via MarCCD remote mode.
- Quad BPM control.
- New motor controllers – *U500, SmartMotor, MDrive, PM600, Microglide, Phidget, etc.*
- Spec controlled motors, scalers, and timers.

Graphical User Interfaces

- Crystallography user interfaces have been moving toward Blu Ice style tabbed interfaces.
- Jim Fait of SER-CAT is writing a new Python-based tabbed GUI called *sergui* that was described in his talk earlier.
- The older Tcl-based *imcagui* of IMCA-CAT has also been remodeled into the tabbed format.
- Rigaku/MSC has been developing a newer IMCA-CAT Java-based GUI called JDirector that will interface to the beamline via a new Java interface to MX.

Graphical User Interfaces (cont.)

- Ken McIvor of IIT has been writing several new Python-based GUIs for materials science.
- These include GUIs for microfocussing, XAFS, and lithography experiments.
- Ken has also written a new GUI for editing MX databases.

Conclusions

- MX has made a substantial amount of progress since the last NOBUGS meeting.
- Development of MX-based servers for foreign control systems such as EPICS is well under way.
- Work is under way to make MX a high performance control system with many substantial improvements already made.
- Several authors are currently working on MX-based beamline control GUIs.

Acknowledgements

H. Bellamy	GCPCC, CAMD
B. Bunker	MR-CAT, Notre Dame
J. Chrzas	SER-CAT, U. Georgia
J. Fait	SER-CAT, U. Georgia
A. Howard	IIT
L. Keefe	IMCA-CAT, IIT
I. Koshelev	IMCA-CAT, IIT
J. Kropf	MR-CAT, Argonne
N. Leyarovska	APS, Argonne
K. McIvor	IIT
G. Rosenbaum	SER-CAT, U. Georgia
C. Segre	MR-CAT, IIT
J. Terry	MR-CAT, IIT