AN OBJECT ORIENTED FRAMEWORK OF EPICS FOR MICROTCA BASED CONTROL SYSTEM

Z. Geng#, SLAC, Menlo Park, California, U.S.A.

Abstract

EPICS (Experimental Physics and Industrial Control System) is a distributed control system platform which has been widely used for large scientific devices control like particle accelerators and fusion plant. EPICS has introduced object oriented (C++) interfaces to most of the core services. But the major part of EPICS, the run-time database, only provides C interfaces, which is hard to involve the EPICS record concerned data and routines in the object oriented architecture of the software. This paper presents an object oriented framework which contains some abstract classes to encapsulate the EPICS record concerned data and routines in C++ classes so that full OOA (Objected Oriented Analysis) and OOD (Object Oriented Design) methodologies can be used for EPCIS IOC design. We also present a dynamic device management scheme for the hot swap capability of the MicroTCA based control system.

INTRODUCTION

Object oriented analysis and design method has become more and more popular in software engineering field, especially after the appearance of UML (Unified Modelling Language) [1] and MDA (Model Driven Architecture) [2].

EPICS has introduced object oriented (C++) interfaces to most of the core services including the operating system abstraction layer. But the major part of EPICS, the run-time database, only provides C interfaces, which is hard to involve the EPICS record concerned data and routines in the object oriented architecture of the software.

The Object Oriented framework for EPICS (OOEPICS) provides several base classes to encapsulate the details of the record processing. The EPICS records and the devices are designed as objects which enable the direct translation of the UML model into EPICS applications. The framework hides most of the details of the EPICS which enable the user to develop his EPICS device driver without knowing much of EPICS.

The OOEPICS framework was originally designed for a MicroTCA based control system which requires dynamic objects creation and deletion to support the hot swap capability of the system. The EPICS base was extended to be able to load the EPICS records dynamically during run time.

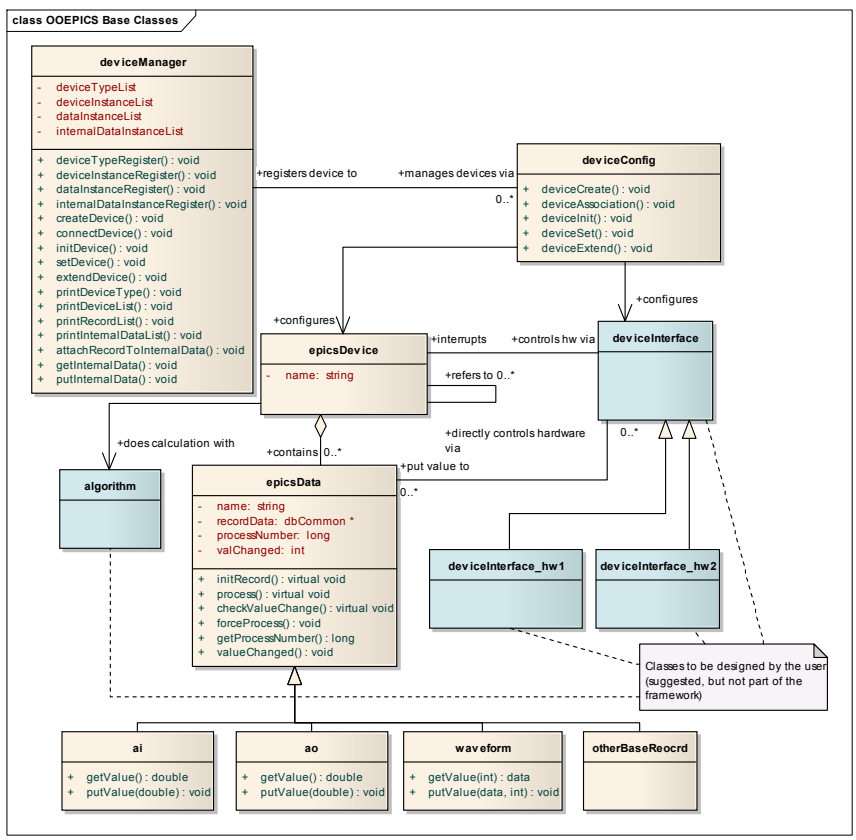

Figure 1: Base classes of OOEPICS framework.

#gengzq@slac.stanford.edu

OOEPICS FRAMEWORK

The OOEPICS framework consists of several base classes and a tool for source code generation which provides a template for the user application.

Base Classes

Figure 1 shows the base classes of the OOEPICS framework.

The base classes setup the basic architecture of the software. Any EPICS applications using the OOEPICS framework can be designed by inheriting the base classes for specific devices.

- epicsData is the base class for EPICS records and device support routines, which realizes some common functionality such as creating EPICS database when the object is created. From that, child classes are derived for all basic EPICS records, such as ao, ai, waveform and so on.
- epicsDevice is the base class for the devices to be controlled which may contains epicsData objects as interfaces to the Channel Access clients.
- deviceConfig realizes common interfaces to configure a device, including registering the device type, creating instances of the device objects and initializing the devices like starting up the threads for the device control.
- deviceManager manages all devices and provides IOC shell commands for device management.

EPICS Records

The key part of the OOEPICS framework is to define an object for each record. The EPICS records were designed more or less based on the object oriented concepts. Table 1 compares the EPICS records and the C++ classes.

Table 1: EPICS Records and C++ Classes

EPICS Records	C++ Classes
Fields	Attributes
Record Support	Methods (platform independent)
Device Support	Methods (platform dependent)

The class of epicsData is defined for the EPICS record. They are linked together by the record name. The object of the class will load the record and obtain the address of the record data during creation and the record will get the address of the object and use its methods as device support.

EPICS Devices

The epicsDevice class is an empty class where the user codes need to be implemented for the control of specific devices.

The epicsDevice class may contain of:

- epicsData objects for uplink control and monitoring.
- Special logic to perform the device control, probably state dependent.
- Algorithms to do signal processing, calibration, optimization and so on.
- Common logical interfaces to interact with physical device.

Device Manager and Configuration

The deviceManager and deviceConfig provide a common way to create, associate, initialize and set up a device control module in user IOC.

The deviceManager class also provides the EPICS API that can be called in the IOC shell. In principle, the user does not need to create any IOC shell commands.

The deviceConfig class should be realized by the developer for specific devices.

Code Generation

The tool of "epics_driver_template" is designed to generate the source code template from the EPICS database file.

The tool will compile the database file into C strings and then use them to initialize the epicsData objects. A derived device class will be generated with the epicsData objects as attributes. A derived deviceConfig class will be generated which allows to define the procedure to create, initialize and setup the device objects.

Development Procedure

Figure 2 shows the procedure for developing EPICS device driver modules with the OOEPICS framework.

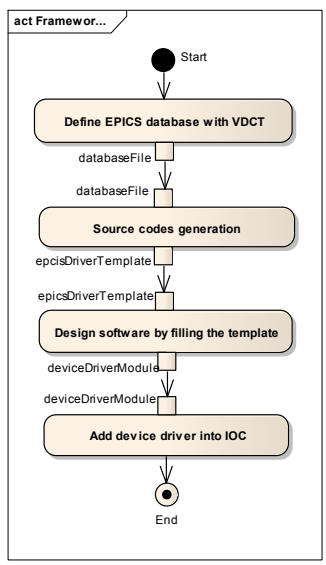

Figure 2: Design procedure with OOEPICS.

DYNAMIC RECORDS LOADING

MicroTCA is hot swappable which allows plugging in new boards without rebooting the crate. The EPICS IOC needs to support such features, which means, loading the device drivers for the new boards without rebooting the IOC software.

The existing EPICS base does not support dynamic record loading. All records must be loaded before executing the iocInit() command. In order to support loading records after iocInit(), the EPICS base needs to be extended.

EPICS Base Extension

Figure 3 shows the architecture of part of the EPICS base concern to the records.

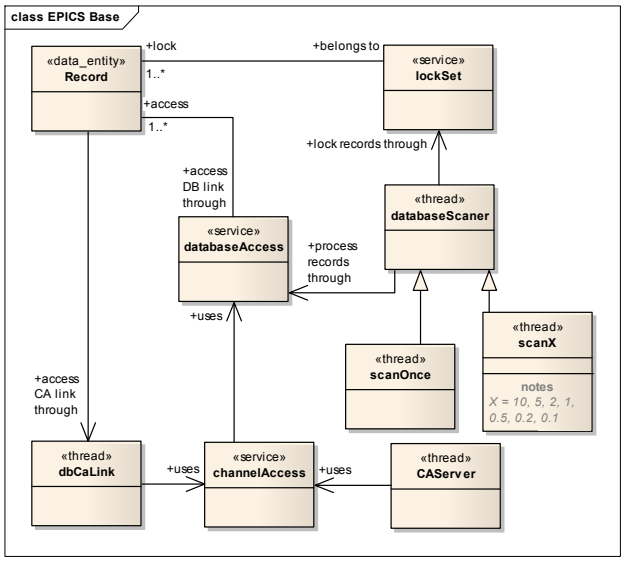

Figure 3: EPICS base modules acting on the records.

Several challenges need to be resolved to load a record during run time:

- Setup the locksets for the newly loaded records. If the new records have no links, simply create new lockset for each of them. If the new records have links with existing records within the same lockset, add the new records to the lockset. And if the new records have links with existing records within different locksets, merge the locksets and add the new records to it.
- Convert PV links to DB link or CA link during record initialization. When converting the PV links, the destination records may have not been loaded yet, so, when a new record is loaded, the record instances loaded previously need to be reconsidered to finish the link conversion.
- Modify the links of existing records. When a new record is loaded, the existing records may want to change their link to point to the new record. In this case, the lockset of the record to be modified should be split firstly and then merged with the lockset of the new link destination.

For solving the problems listed above, a new source file (dbRecordDynamic.c) is added to the EPICS base [3].

The OOEPICS framework and the dynamic records loading functions have been successfully tested in a soft IOC controlling a Newport motion control stage.

CONCLUSION

The OOEPICS framework provides a way to fully access the EPICS records from the user code. The object oriented technology can directly map to EPICS design. It also provides a common way to create, initialize and setup the device driver from the IOC shell. It also enables the user to do EPICS development without knowing much of EPICS, avoiding the long learning curve of EPICS.

The dynamic records loading development provides a good support for hot-swappable system control which enables a complete EPICS based solution for MicroTCA system in the following projects at SLAC.

REFERENCES

[1] http://www.uml.org/
[2] http://www.omg.org/mda/
[3] https://blueprints.launchpad.net/epics-base/+spec/dynamic-record-loading