CPP/CXML - A HOST-BASED SEQUENCER FOR EPICS

P. Gurd, R.Keitel

TRIUMF, 4004 Wesbrook Mall, Vancouver, BC, V6T 2A3, Canada.

Abstract

The EPICS-based control system of the ISAC (**I**sotope **S**eparation and **AC**celeration) Radioactive Beam Facility uses cppe, a host-based sequencer. cppe was adapted from a previous application and uses its own simple sequencing language. This paper describes CPP/CXML, a replacement for cppe, which was implemented as a Perl module on top of the EPICS CA Perl module. CPP/CXML retains important features of cppe, such as process variable declarations, connection checks and cleanup on abort. In addition it leverages the full capabilities of the Perl language and incorporates a state machine processor. Sequences can be executed either in Perl using the Perl procedural API or by defining a state machine using XML.

INTRODUCTION

ISAC [1] the Radioactive Beam Facility at TRIUMF [2] consists of several linear accelerators, which produce beams of short-lived radioactive isotopes by bombarding suitable targets with protons from the TRIUMF 500 MeV cyclotron. Approximately 4000 ISAC devices are controlled by an EPICS based control system [3]. For more than ten years, a host based sequencer cppe has been used for

- Rapid prototyping of slow control algorithms.
- Replacing repetitive operator actions
- Replaying operator actions captured by a macro facility.

This paper describes CPP and CXML, a new set of Perl modules developed for the ISAC control system as a cppe replacement.

cppe

The present system cppe (Command Procedure Processor for EPICS) [4] [5] is a host-based sequencer written in the C language, which was developed for earlier control systems at TRIUMF and was ported to the EPICS environment. Input files for cppe are written in a cppe-specific procedural language. Some features of cppe:

- All variables used, including EPICS process variables (PVs), must be declared before execution starts.
- Input files must pass a syntax check for execution to continue.
- The cppe language provides flow and timing controls based on logical and arithmetic conditions.
- cppe allows macro substitution from command line parameters.
- Procedure locking, abort, and message reporting must be explicitly handled by the procedure designer.

Examples of cppe use in the ISAC control system include ramping of power supplies with user-specified parameters, pumping down beam lines, and synchronized heating of the ISAC targets.

Macro facility

cppe is also used as part of the ISAC macro facility, which allows the capture of operator actions. A Perl tool extracts operator commands from the EPICS Channel Access put log and generates a procedure file, which can be used for timed replay by cppe or as a starting point for developing a more sophisticated procedure.

WHY MOVE TO PERL?

cppe has been a successful component of the ISAC control system. The release by the EPICS community of CA.pm**,** a channel access client Perl module [6], however, provided a welcome opportunity for developing a replacement. Perl is already used as the standard scripting language of the ISAC control system and provides much more capability than the simple cppe language. In addition, switching from cppe to Perl removes one language from the required core competencies of an ISAC control system programmer.

CPP

CPP.pm is a package of Perl routines that implements an application programming interface (API) for procedural host-based sequences. It provides a wrapper around the CA.pm package which encapsulates desired core functionality, so that the sequence designer is not distracted by housekeeping chores and can focus on developing the desired sequence. The CPP.pm module retains important cppe functionality, such as process variable declarations, initial connection checks and cleanup on abort, but improves on cppe by full connection handling and aborting of a procedure if any PV disconnects subsequently. The package provides routines to get data from PVs, send data to PVs, delay procedures and test conditions. The package defines standard methods for operator abort, status reporting, procedure locking, synchronizing and exclusion. It allows parameterized common routines and improves on cppe's cleanup mechanism by providing a way to register a cleanup procedure. CPP.pm provides functions for parameter definition, flow control, and PV access. The CPP.pm API functions are listed in the remainder of this section.

Procedure parameters are defined using the *decodeCommandLine* and *substituteMacros* functions.

In addition to the standard Perl language constructs, procedure flow is controlled by the *startExecution*, *abortProcedure*, *registerCleanup*, *registerAbort*, *cppSleep* and *exitProcedure* functions. In particular, the *startExecution* function allows the declaring of PVs for procedure locking, procedure aborting and reporting of status messages. The *setSleep* and *setWait* functions control the behaviour of the PV functions that follow. The *waitFor* function controls program flow based on a PV value.

connectPVs creates channel access connections and monitors for the PVs used by the PV access functions *getPVString*, *getPVDouble*, *evalCondition*, *setPVString*, *setPVDouble* and *stepPVDouble*.

Messaging functions are *reportMessage*, which sends a message to the message PV and *abortMessage*, which defines a message to report a procedure abort.

Operator interface examples

For a consistent operator interface, a generic screen for the EPICS extensible display manager (edm) was developed (see Fig.1), which can be used by any CPP.pm procedure that includes the abort, lock and report control process variables.

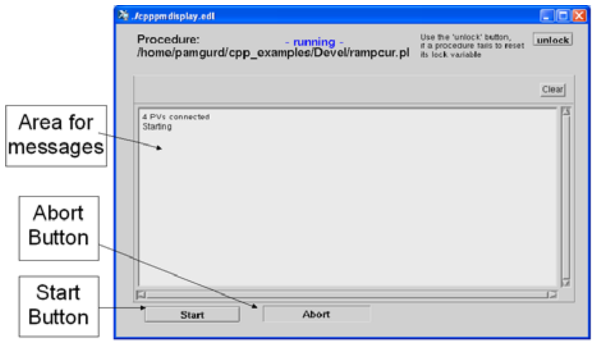


Figure 1: Generic CPP control screen

Application examples

CPP.pm has been deployed successfully in replacing the cppe procedures for setting the ISAC mass separator magnetic fields, heating the ISAC East and West targets (see Fig. 2) and conditioning of the DRAGON (**D**etector of **R**ecoils **A**nd **G**ammas **O**f **N**uclear reactions) electrostatic dipoles high voltage

.

Experience and future directions

The CA.pm Perl module was easy to install and use. Initially there were problems encountered with the last few channel access operations after a procedure was aborted. Those problems were solved with help from the CA.pm author.

At present we continue to replace cppe procedures with the goal of phasing out cppe by end of the year. This requires recreating the macro facility, i.e. to create CPP procedures from captured caPutLog files.

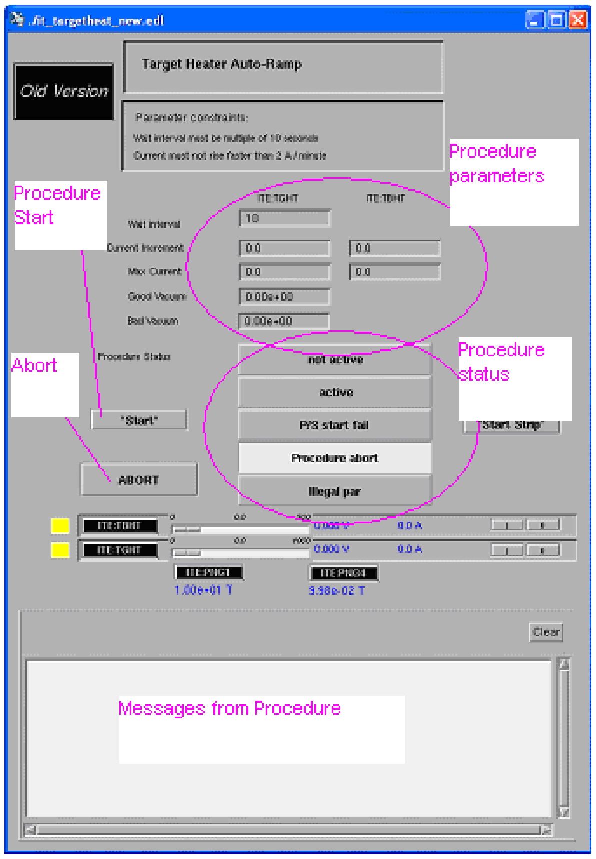


Figure 2: Target synchronized heater control

CXML

Whereas CPP.pm implements an API for procedural sequences, CXML executes a State Machine defined in an XML document. An XML definition developed for telephony - State Chart XML (SCXML): State Machine Notation for Control Abstraction [7] - was modified for controls applications. CXML provides a collection of Perl packages that automatically produce edm screens and database files to support the XML-defined state machine. The state machine can be run either at the host level in Perl using the CPP.pm package with Perl/Tk for optional display, or it can run on an IOC as an EPICS state notation language (SNL) sequence.

Fig. 3 shows how CXML is used to produce supporting EPICS databases, edm display screens, and SNL code, or to run in Perl using CPP.pm. Fig. 4 shows a fragment of an XML specification of a state machine. Fig. 5 shows a generated edm screen, which displays the execution of the state machine.

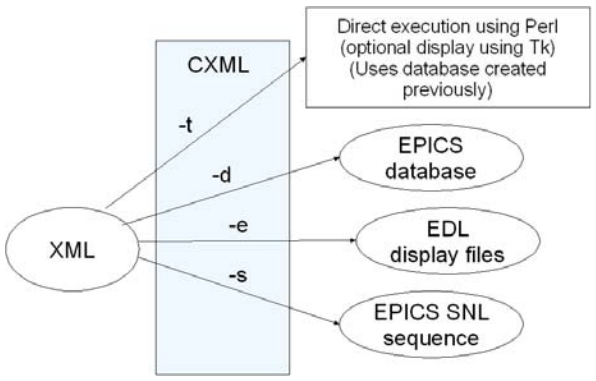

CXML
XML
-t
-d
-e
-s
Direct execution using Perl (optional display using Tk) (Uses database created previously)
EPICS database
EDL display files
EPICS SNL sequence

Figure 3: CXML options

CXML Database

A <definition> entity was added to the SCXML specification to allow creation of an EPICS database to support the state machine. PVs that may be needed include abort, start, lock and message PVs as well as other support PVs for the state machine such as limits, step sizes and wait times. Latches can be defined to catch momentary conditions. The database can include parameters for efficient reuse and standardization of code.

```
<state id="Start">
   <onentry id="Start.entry">
      <log expr="Entering voltage ramp"/>
      <send target="par(PS):DRVON"
         targettype="epics-pv"
         data="1"/>
      <send delay="2 s"/>
   </onentry>
   <transition cond="pv(par(PS):STATON) &lt; 0.5"
     target="All_done">
      <log expr="power supply did not turn on" />
      <send target="CXML:par(PS):RAMPSTAT"
         targettype="epics-pv"
         data="2"/>
   </transition>
```

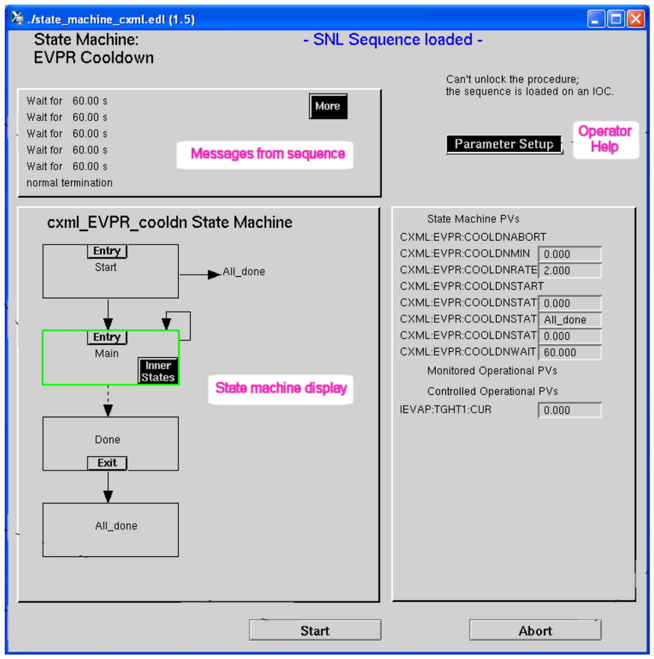
Figure 4: Fragment of CXML state machine definition

CXML Application Examples

CXML has been deployed at TRIUMF in three test areas: the laser ion source, the ion source test stand and the target preparation (evaporator) stations. In all three source test areas, EPICS SNL sequences were created using CXML to ramp up the two heaters of the source in tandem while checking the vacuum. In the laser ion source, sequences were also created to ramp the bias voltage up and down while checking the vacuum readings.

CXML Outlook

At present, the ISAC production control system does not use any SNL sequences. Given the successful deployment of the test system examples, this could be revisited in the light of future requirements. A visual state machine editor, possibly integrated with the EPICS database tool, would help making the case for including state machines.



State Machine:
EVPR Cooldown

- SNL Sequence loaded -

Can't unlock the procedure;
the sequence is loaded on an IOC.

```
Wait for   60.00 s
Wait for   60.00 s
Wait for   60.00 s
Wait for   60.00 s
Wait for   60.00 s
normal termination
```

cxml_EVPR_cooldn State Machine

State Machine PVs

CXML:EVPR:COOLDNABORT	
CXML:EVPR:COOLDNMIN	0.000
CXML:EVPR:COOLDNRATE	2.000
CXML:EVPR:COOLDNSTART	
CXML:EVPR:COOLDNSTAT	0.000
CXML:EVPR:COOLDNSTAT	All_done
CXML:EVPR:COOLDNSTAT	0.000
CXML:EVPR:COOLDNWAIT	60.000

Monitored Operational PVs

Controlled Operational PVs

IEVAP:TGHT1:CUR	0.000

Figure 5: State machine display

REFERENCES

[1] http://www.triumf.ca/about-triumf/triumf-faq/isac-backgrounder

[2] http://www.triumf.ca/about-triumf/triumf-faq/triumf-backgrounder

[3] R. Keitel, D. Bishop, M. Leross, R. Nussbaumer, C. Payne, K. Pelzer, J. Richards, W. Roberts, E. Tikhomolov, G. Waters, "ISAC Control System Update", ICALEPCS07, Knoxville

[4] R. Keitel, D. Bishop, D. Dale, H. Hui, S. Kadantsev, M. Leross, R. Nussbaumer, J. Richards, E.Tikhomolov, G. Waters, "Status Update on the ISAC Control System", ICALEPCS01, San Jose

[5] cppe manual, http://isacwserv.triumf.ca/isac/ pubdoc/cppe/CPPEPICS.DOC}

[6] CA - Perl 5 interface to EPICS Channel Access http://www.aps.anl.gov/epics/base/R3-14/11-docs/CA.html

[7] State Machine Notation for Control Abstraction (http://www.w3.org/TR/scxml)