LCLS How-to Use Autosave

LCLS How-to use Autosave and catPuLog

Manual
Author's Web Site
EPICS Extensions Home Page
EPICS Extensions Download
EPICS Extensions at LCLS
EPICS at LCLS

Author	Zenon M. Szalata and Ernest L. Williams Jr.
Date	Feb 4, 2008
EPICS Version	R3.14.8.2 and higher
Autosave Version	R4.4
OS	RTEMS/VxWorks
Description	This note summarizes steps needed to add autosave support to an IOC

Introduction

The LCLS project will need a facility for bumpless reboots as well as a facility for logging channel access puts to the IOC errorLog facility.

Bumpless Reboots

Autosave automatically saves the values of EPICS process variables (PVs) to files on a server, and restores those values when the IOC is rebooted. The autosave package is distributed and maintained by Argonne National Laboratory.

Autosave is a two-part operation: run-time save, and boot-time restore. The run-time part (save_restore.c) is started by commands in the IOC startup file, and persists while the IOC is running. Its primary job is to save PV values to files on a server, but it also supports manual restore and other management operations. The boot-time part (dbrestore.c) is invoked during iocInit, via an EPICS initHook. It restores PV values from files written by the run-time part, and does not persist after iocInit() returns.

In addition to the autosave software, the autosave module contains a client program, asVerify, to compare written autosave files with current PV values. This program can also write an autosave file from which PV values can be restored.
Please Read the documentation before executing this "How To": http://aps.anl.gov/bcda/synApps/autosave/autoSaveRestore_R4-4.html

Channel Access Put Logging

CaPutLog is an EPICS support module that provides logging for Channel Access put operations. The current version works with EPICS 3.14.8.2 and later and is almost a complete re-write of the original (3.13 compatible) version that was written by V. Korobov (DESY). It is based on the generic iocLogClient that is part of the EPICS libCom. Starting a log server requires exactly the same steps as for the regular iocLogServer, except that you probably want to use a different port. On the IOC side there are three routines to be called from the iocShell. See the How to below.

1. Add the following line to the $(TOP)/configure/RELEASE

AUTOSAVE = <top directory of the autosave support module>
The current version is based on autosave 4-4:
AUTOSAVE_MODULE_VERSION=Development

The following version has been built for use with EPICS base tag: base-R3-14-8-2-lcls2
AUTOSAVE_MODULE_VERSION=autosave-R4-4-lcls1

2. Add the following two lines to the xxxApp/src/Makefile

xxx_DBD += autosaveSupport.dbd
...
xxx_LIBS += autosave

The first of the two adds device support to the xxx.dbd file, the second
includes the autosave objects into the IOC binary.

3. Add this line to the xxxApp/Db/Makefile

DB_INSTALLS += $(AUTOSAVE)/db/save_restoreStatus.db

```
This is just a convenience operation which copies the file to the
$(TOP)/db of the IOC. The save_restoreStatus.db file is used by autosave
to keep its own statistics in the IOC.
```

At this point you should be able to build the IOC without errors.
final step is to add the autosave support to the st.cmd file.

4. Add the following lines to the $(TOP)/iocBoot/iocxxx/st.cmd

```
save_restoreSet_status_prefix( "XYZ")
```

The following 2 lines are optional

```
save_restoreSet_IncompleteSetsOk( 1)
save_restoreSet_DatedBackupFiles(1)
```

1. /data is a NFS mount point such that in this example the autosave files will be kept in
2. /data/autosave directory.

```
set_savefile_path( "/data","autosave")
```

1. I chose the request files to be located in the directory where st.cmd file is also located.

```
set_requestfile_path( "iocBoot/iocxxx")
```

1. data is restored from this file before record initialization

```
set_pass0_restoreFile( "<file0>.sav")
```

1. data is restored from this file after record initialization.

```
set_pass1_restoreFile( "<file1>.sav")
...
dbLoadRecords( "db/save_restoreStatus.db","P=XYZ")
...
iocInit()
...
```

1. optional, needed if the IOC takes a very long time to boot.

```
epicsThreadSleep(1.0)

create_monitor_set( "<file0>.req",30,"P=<prefix>")
create_monitor_set( "<file1>.req",30,"P=<prefix>")
...
```

additional monitor and/or other sets can be created as needed.
<file0> and <file1> as specified in the set_pass0_restoreFile() and
set_pass1_restoreFile(), can be the same file, in which case only one
create_monitor_set() is used.

*NOTE: that <prefix> is the value to be used in the request files to construct
PV names as they are defined in the xxx.db file for this IOC.*

*NOTE: for details on all autosave function calls as well as the format of the
request file, consult the documentation provided with the autosave distribution.*

5. The autosave files are written to NFS mounted file system. If this needs to
be done from the st.cmd file, here is how this can be done.

a. rtems

```
setenv( "NFS_FS","<uid>.<gid>@<ip_node_address>")
setenv( "NFS_DATA","<absolute path of the directory to be mounted>")
nfsMount( getenv("NFS_FS"),getenv("NFS_DATA"),"/data")
```

where I chose /data to be the mount point to be consistent with the
argument to set_savefile_path() see above.

b. vxWorks

```
hostAdd( "<nodename>","<ip_node_address>")
netDevH=iosDevFind( "<nodename>:")
nfsMount( "<nodename>","<directory path to be mounted>","/data")
nfsAuthUnixSet( "nodename",<uid>,<gid>,0)
```

If the IOC and the NFS server are on different networks, a routing
information may need to be specified. I think this is done in vxWorks
with the mRoutAdd() function. I did not use it.

c. soft IOC

6. An info node, in an EPICS database, is similar to a field specification, but it has the word info instead of field; and it has an arbitrary name, instead of the name of a field in the record. Here's an EPICS database containing a single record with two info nodes:

record(ao, "$(P)test1") {
```
field(DTYP, "Soft Channel")
info(autosaveFields, "PREC EGU DESC")
info(autosaveFields_pass0, "VAL")
```
}

From this information, makeAutosaveFiles() will write the following two files:

info_settings.req
$(P)test1.PREC
$(P)test1.EGU
$(P)test1.DESC

info_positions.req
$(P)test1.VAL

CAPUTLOG_MODULE_VERSION=Development

For EPICS BASE: base-R3-14-8-2-lcls2
CAPUTLOG_MODULE_VERSION=caPutLog-R1-0