LIGO Laboratory / LIGO Scientific Collaboration

LIGO-T1300690-v1 *LIGO* 2013 August 15

TwinCAT EPICS IOC Documentation

Maggie Tse

California Institute of Technology
LIGO Project

Massachusetts Institute of Technology
LIGO Project

LIGO Hanford Observatory

LIGO Livingston Observatory

http://www.ligo.caltech.edu/

TwinCAT EPICS IOC Documentation*

TwinCAT-EPICS IOC

The TwinCAT-EPICS IOC synchronizes TwinCAT variables with EPICS channels. This software will replace the previously used OPC-EPICS Gateway from HZB. All of the files mentioned below can be found in the Slow Controls SVN repository.

Overview

The IOC handles communication between two interfaces: TwinCAT and EPICS. As such, it makes use of the TwinCAT ADS Communication Library to communicate with the PLC on one side and the libraries of the EPICS Base on the other side.

TwinCAT Communication

The IOC keeps an internal database of the records that it manages. This gets initialized through parsing the tpy file of a TwinCAT project, which is generated every time the project is built. This tpy file will provide the names and memory locations of TwinCAT symbols on one PLC. The IOC uses this information to read and write data via ADS commands to TwinCAT. Reading and writing are done via two scanner threads, each executing a read or write command every *n* ms, with *n* set by tcSetScanRate at IOC startup (see the example st.cmd below). Multiple tpy files can be loaded in a single IOC, i.e. the IOC can manage records on multiple PLCs.

EPICS Communication

To notify EPICS that a record value has been updated, the IOC will generate an interrupt or a callback request to EPICS, telling EPICS to process that record. EPICS record processing includes grabbing the value from the IOC's internal memory, checking alarms, processing linked records, and updating channel access clients (e.g. MEDM). Conversely, when EPICS receives a new value for a record through channel access, it will process the record, and write the new value to the IOC's internal memory.

Synchronization

The setup of the IOC's internal database is designed to prevent the overlap of TwinCAT/EPICS read/write requests on the same record, which could cause data values to become lost.

The internal data value has flags to indicate that the value has been written by one side (TwinCAT or EPICS), but has not yet been read by the other side. Thus, if EPICS has written a new value to the IOC, TwinCAT cannot overwrite that value until it has read the old value. This logic guarantees that data values do not get lost within the IOC, that a value written on one side always makes it to the other side, preventing collisions of write requests. We also prevent the crossing of a read request with a write request by having the read/write scanners on the TwinCAT side use mutexes when accessing the IOC's database. Note that simultaneous reading of the IOC database does not pose any problems, so this case is not prevented.

In addition, the IOC runs a scanner thread for each PLC that slowly crawls through the internal database and pushes data values to EPICS. It will cover the entire database once per minute, which guarantees that the record values seen in EPICS/MEDM are never more than a minute old.

Other Features

At startup, the IOC will parse tpy files to generate EPICS databases. The user can specify which types of records are exported, conversion rules between TwinCAT names and EPICS names, and whether to split databases into multiple files. The user can also allow the parser to generate other lists alongside the EPICS database, for example channel lists and burt

restore files. A full list of the available options can be found on page TwinCAT EPICS Options.

Running the IOC

The startup requires several user inputs. A set of commands is given as a parameter to tcIoc.exe which specifies these inputs. Below is an example:

```
dbLoadDatabase("./tCat.dbd",0,0)              #load database definition file for
TwinCAT device support
tCat_registerRecordDeviceDriver(pdbbase)      #register TwinCAT support with EPICS
callbackSetQueueSize(5000)                    #set the size of the EPICS callback
buffer (default: 2000)
tcSetScanRate(10, 5)                          #set the time between requests to
TwinCAT (10ms here)
                                              #and the slowdown multiple for pushing
values to EPICS (5x here)
tcLoadRecords("C:\SlowControls\Target\H1ECATX1\PLC1\PLC1.tpy", "")    #load symbols
from a tpy file
iocInit()                                     #initialize the IOC
```

A complete list of TwinCAT EPICS commands can be found on page TwinCAT EPICS Commands.

To start the IOC,

1. Install using the PowerShell script in SlowControls\Scripts\Common\install_tcioc.ps1
2. Run the start script SlowControls\EPICS\Utilities\Bin\start.bat

On some systems it may be required to install the Visual C++ Redistributable for Visual Studio 2012

Performance

Several checks on the performance of the IOC have been made to verify that it will be able to reliably handle all ~20,000 Slow Controls channels for extended periods of time.

Test system

Hardware:

- Processor: Intel Xeon CPU X5650
- Cores: 6 HT
- Threads: 12
- Speed: 2.67GHz
- Mmeory: 12 GB; 2.99GB usable

Software:

- OS: Windows 7
- Version: 32-bit operating system
- TwinCAT: 2.11

Speed tests

- TwinCAT *(test performed on 6/21/2013)*
  This test was performed to see how much data we can read from TwinCAT in one request before overloading the system.
    - 1 channel
        - 1.076ms to read data

        - TwinCAT System Real Time Usage: was not monitored
    - 1000 channels (~10kB)
        - 1.084ms to read data out in one request
        - TwinCAT System Real Time Usage: no noticeable change
    - 3,200 channels (~30kB)
        - 1.087ms to read data out in one request
        - TwinCAT System Real Time Usage: +1-2%
    - 7,500 channels (~70kB)
        - 1.099ms to read data out in one request
        - TwinCAT System Real Time Usage: +3-4%
    - 15,000 channels (~150kB)
        - 1.121ms to read data out in one request
        - TwinCAT System Real Time Usage: +4-5%
- TwinCAT *(test performed on 6/20/2013)*
  This test was performed to see how generating individual requests for each channel can overload the TwinCAT system. In this example we specified the memory location for each channel, instead of requesting one large memory region as above. This method proved to be too taxing on the TwinCAT system, so we do not use it in our IOC. Compare to the above performance figures.
    - 1000 channels
        - 1.306ms to get data for all channels
        - TwinCAT System Real Time Usage: +20%
    - 4000 channels
        - 1.483ms to get data for all channels
        - TwinCAT System Real Time Usage: +60-80%
- EPICS *(test performed on 7/30/2013)*
    - It takes ~1.33s to process 1,000,000 records
    - Thus in a 10ms cycle it can process ~7500 records

Memory usage

(test performed on 8/9/2013)

- Running on the corner EtherCAT machine (H1ECATC1) (17000 records)
    - IOC uses 41MB of memory
    - 3% of CPU time
- Running on the end-X EtherCAT machine (H1ECATX1) ([xxx] records)
    - IOC uses [xxx]MB of memory
    - [xxx]% of CPU time

High volume performance

(test performed on 8/9/2013)

- The IOC can safely handle a burt restore on the corner station, which restores all the EPICS records that are not read-only (~2000 in this case).
- The IOC can safely handle sequences of commands generated at a fast rate by the ezca tool.

Performance over time

(test performed on 8/9/2013)

- The IOC has safely run for ~70 hours continuously on H1ECATC1 without any noticeable changes in memory usage or performance.

Future Features

While the core functionality of the IOC is already in place and fully tested, several additional features are being developed and will be implemented soon:

- IOC status records:
  The idea is to keep track of things related to the IOC itself and export these to EPICS. Possible interesting figures are:
    - Average number of records updated per unit of time
    - Average amount of time per TwinCAT read/write request

Upgrades/Adaptations

- TwinCAT 3 compatibility
- The code has enough abstraction that it can easily be adapted to work with other non-TwinCAT PLC controllers as well as other non-EPICS industrial control systems.

Code Documentation

The program was written in C++ and built using Visual Studio 2012, the TwinCAT ADS Communication Library that comes with TwinCAT PLC v2.11.1551, and EPICS Base version 3.14.12.3. Code documentation is generated by Doxygen and can be found at this webpage:

Instructions for building the ioc from the source can be found at TwinCAT EPICS IOC Source Build.

---

* This document was created from a snapshot of the aLIGO wiki "aLIGO: TwinCAT EPICS IOC Documentation" at 11:00 AM PDT on 2013 August 15. Conversions done by V. Sandberg.
https://awiki.ligo-wa.caltech.edu/aLIGO/TwinCAT%20EPICS%20IOC%20Documentation