Table of Contents

Setup EPICS to communicate with Siemens PLC

Intro

The ESTHER Slow Controller system controls the gas injection into the combustion chamber of the shock tube. This system uses the EPICS framework to communicate with a Siemens SIMANTIC S7-1200 PLC using the Open Source EPICS Device Support Library s7nodave. A Control System Studio graphic user interface client is used to visualize and control the whole experiment by connection with the EPICS IOC server.

This tutorial is presented as a reference to setup a server computer running an EPICS server that connects to the S7 PLC. An example EPICS IOC "*testS7*" application and accompanying CSS interface are provided. This example application allows the control and observation of all digital IO and analog inputs of the connected S7-1200 PLC.

System Setup

Hardware Platform

The requirements to setup this testS7 application are:

1. A computer running a 32-bit Operating System (a 32-bit operating system is required to implement polling groups using the s7nodave device support library in EPICS). Two ethernet cards (One connected to the local network without internet access where the PLC is present).
2. A Siemens SIMANTIC S7-1200 PLC powered with 24 VDC.
3. A client computer running the CSS interface.

The setup used for this tutorial was:

1. A Supervisor EPICS server:
    1. Virtual Machine running Debian GNU/Linux 7.7.0 32-bits
    2. Linux Kernel 3.2.0-4-486
    3. Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
    4. 512 MB RAM
    5. Two Bridged Intel Corporation 82540EM Gigabit Ethernet Controllers
        1. eth0 connected to IPFN network with IP 10.136.237.21 subnet 255.255.0.0 (with internet access)
        2. eth1 directly connected to the S7-1200 PLC with IP 192.168.0.70 subvnet 255.255.255.0 (without internet access)
2. A Siemens SIMANTIC S7-1200 PLC powered at 24 VDC using an external power supply and connected to the eth1 of the Supervisor EPICS Server with IP 192.168.0.1 subnet 255.255.255.0.
3. A client computer running CSS on the IPFN network 10.136.xxx.xxx subnet 255.255.0.0.
4. A computer running a Windows Operating System with the Siemens Totally Integrated Automation (TIA) PLC programming software packaged installed to develop and program the PLC firmware.

Software Platform

All the software required for this test application is located in the IPFN ESTHER SVN repository: http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/

EPICS IOC: http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/EPICS/IOC/testS7/

PLC Firmware: http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/TIA/S7-1200/testS7_PLC/

CSS Interface: http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/CSS/S7EPICS_CSS/

Getting Started

Setup Supervisor Example EPICS Server

1. Install clean Debian 7.7.0 32-bit distribution on a computer or virtual machine with bridged network adapter
2. Compile EPICS Base:
    1. Follow the Getting Started of http://epics.nsls2.bnl.gov/debian/
    2. apt-get install epics-dev build-essential
    3. apt-get install epics-synapps-dev epics-iocstats-dev visualdct openjdk-6-jdk sysv-rc-softioc
    4. The latest EPICS base version is then installed to /usr/lib/epics
3. Install s7nodave device support library
    1. cd /opt/epics/modules
    2. wget -c http://oss.aquenos.com/epics/s7nodave/download/s7nodave-1.0.3.tar.gz -O - | tar -zx
    3. cd s7nodave-1.0.3/
    4. vi configure/RELEASE
    5. Replace "EPICS_BASE=/usr/lib/epics" or add to the end of the file
    6. Save and exit
    7. Install libboost if needed: apt-get install libboost-all-dev
    8. make
    9. cd ..
    10. Create symbolink link to folder: ln -s ./s7nodave-1.0.3/ s7nodave
4. Install the EPICS state-machine sequencer library
    1. wget -c http://www-csr.bessy.de/control/SoftDist/sequencer/releases/seq-2.1.17.tar.gz -O - | tar -zx
    2. cd seq-2.1.17/
    3. vi configure/RELEASE
    4. Replace "EPICS_BASE=/usr/lib/epics" or add to the end of the file
    5. Install lexer generator tool re2c: apt-get install re2c
    6. make
    7. cd ..
    8. Create symbolink link to folder: ln -s ./seq-2.1.17/ seq
5. Add testS7 project from the SVN repository
    1. apt-get install subversion
    2. svn co http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/EPICS/IOC/testS7/
    3. cd testS7/
    4. Set the correct library dependencies for the modules:
        1. vi configure/RELEASE

2. Replace or add "SNCSEQ=/opt/epics/modules/seq"
3. Replace or add "S7NODAVE=/opt/epics/modules/s7nodave"
4. Replace or add "EPICS_BASE=/usr/lib/epics"
5. Save and exit

5. make

6. Execute the testS7 example IOC
    1. cd iocBoot/ioctests7/
    2. chmod +x st.cmd
    3. ./st.cmd

testS7 IOC Example Application

The EPICS testS7 IOC Application consists in the following sections inside the [testS7] root folder:

- A database file containing the records of the process variables for the expected IOC application. These are in the [testS7]/tests7App/Db/dbS7.db file.
- The state-machine sequencer program that is located in the [testS7]/tests7App/src/snctests7.stt file.
- The EPICS executable startup script in [testS7]/iocBoot/ioctests7/st.cmd

Records Database

In this example IOC application, all the available S7-1200 PLC digital and analog IO are set in the application database records in the file [testS7]/tests7App/Db/dbS7.db.
The standard S7-1200 PLC contains 10 digital outputs ports, 14 digital inputs ports and 2 analog inputs ports.
The digital inputs and outputs can only have the values 1 or 0 and must be read by the IOC application using a binary in (bi) or written to using a binary out (bo) record.
A PLC digital input port can only be read and not written to, so only a bi record is set in the IOC application database.
A PLC digital output port can be both read and written to. This means that to read the status of the PLC digital output, the IOC application must use the bi record for that PLC port. And to set the status of that port, the IOC application must use the corresponding bo port.
The mnemonic to consider here is that the bi and bo records correspond to what is being read or written by the IOC application, and not the PLC.

In the case of the S7-1200 PLC, the port is for each record is defined by setting the value "@myPLC Q0.1" in the INP or OUT fields, for example. In this case the "myPLC" value is the alias given to the PLC in the [testS7]/
iocBoot/ioctests7/st.cmd file (s7nodaveConfigureIsoTcpPort("myPLC", "192.168.0.1", 0)), and the Q0.1 corresponds to the digital output group 0, bit 1.

The S7-1200 PLC have the following address aliases:

Digital Outputs (prefix Q):
- Group 0 bit 0 -> Q0.0
- ...
- Group 0 bit 7 -> Q0.7
- Group 1 bit 0 -> Q1.0
- Group 1 bit 1 -> Q1.1

Digital Inputs (prefix I):
- Group 0 bit 0 -> I0.0
- ...
- Group 0 bit 7 -> I0.7
- Group 1 bit 0 -> I1.0
- Group 1 bit 6 -> I1.5

Analog Inouts:
- Analog Input 0 -> IW64
- Analog Input 1 -> IW66

Poll Groups

All the input records in the database have a SCAN field. The IOC application will scan the value of each record at the interval defined in the SCAN field. For a SCAN field of "0.1 second" of a input record, the IOC application will query the PLC for that record's value once every 100 miliseconds.
However, EPICS treats each query independently, so by setting the SCAN field of every input record in this way, EPICS will query each value thorugh the local network and wait for the response before moving on to the next query, resulting in delayed responses.
The s7nodave library allows records to be assinged to polling groups. All the records in a pollgroup are queried in the same network packet, greatly reducing response lag.
In order to assign an input record to a poll group using the s7nodave library, one must set the SCAN field to "I/O Intr", and the INP field to "@myPLC(PG=[POLLGROUP_NAME]) [PLC_ADDR]". This can be seen in the example below for the PV S7:biI1_5poll.

```
record(bi, "S7:biI1_5poll")
{
    field(SCAN, "I/O Intr")
    field(DTYP, "s7nodave")
    field(INP, "@myPLC(PG=1s) I1.5")
    field(DESC, "S7 PLC I1.5 Binary Input POLLING")
}
```

Ommiting (PG=[POLLGROUP_NAME]) will assign the PV record to the "default" poll group.

Sequencer Program

The example sequencer program inside this testS7 is a simple state-machine that monitors the value of a PLC port and sets another port to high or low accordingly. This program sequencer can be seen in the [testS7]/tests7App/src/snctests7.stt file:

```
program snctests7

double v;
short light;
```

```
assign v to "S7:biQ1_0";
monitor v;
assign light to "S7:boQ0_7";
ss ss1 {
   state init {
        when (delay(1)) {
          printf("snctests7: Startup delay over\n");
          printf("snctests7: Going to low\n");
        } state low
   }
   state low {
        when (v >= 0.5) {
                printf("snctests7: Changing to high %f\n",v);
                light = TRUE;
                pvPut(light);
        } state high
   }
   state high {
           when (v <= 0.5) {
                printf("snctests7: Changing to low %f\n",v);
                light = FALSE;
                pvPut(light);
        } state low
   }
}
```

IOC startup script

The executable startup script in [testS7]/iocBoot/ioctests7/st.cmd starts the EPICS server IOC. This script is presented below.

It configures the connection with the name "myPLC" to the PLC on 192.168.0.1. Configures the "default" pollgroup with a 0.1 second interval and the "1s" poll group with a 1 second interval. Then loads the database records substitutin the macro "S7_DEFAULT_SCAN_INTERVAL" with "I/O Intr". Finally it starts the sequencer program "seq snctests7".

```
cd ${TOP}
## Register all support components
dbLoadDatabase "dbd/tests7.dbd"
tests7_registerRecordDeviceDriver pdbbase

#Configure PLC Connection
s7nodaveConfigureIsoTcpPort("myPLC", "192.168.0.1", 0)

# Configure Poll Group
s7nodaveConfigurePollGroup("myPLC", "default", 0.1, 0)
s7nodaveConfigurePollGroup("myPLC", "1s", 1, 0)

## Load record instances
```

```
#dbLoadRecords("db/dbS7.db","S7_DEFAULT_SCAN_INTERVAL=.1 second")
dbLoadRecords("db/dbS7.db","S7_DEFAULT_SCAN_INTERVAL=I/O Intr")

cd ${TOP}/iocBoot/${IOC}
iocInit

## Start any sequence programs
seq snctests7
```

Compilation

This section assumes a correct setup of the system as explained in the Getting Started. After changing these files, the whole IOC must be recompiled from the root directory.

In order to compile an IOC using the s7nodave library the following steps must be complete:

- The S7NODAVE=[path_to_module] must be set in the configure/RELEASE file
- The [testS7]/tests7App/src/Makefile must include the lines:
  - tests7_DBD += s7nodave.dbd
  - tests7_LIBS += s7nodave
- The PLC connection must be configured in the startup script [testS7]/iocBoot/ioctests7/st.cmd
- The poll groups must be configured in the startup script [testS7]/iocBoot/ioctests7/st.cmd

In order to compile an IOC with a sequencer program the following steps must be complete:

- The SNCSEQ=[path_to_module] must be set in the configure/RELEASE file
- The [testS7]/tests7App/src/Makefile must include the lines:
  - snctests7_SNCFLAGS += +r
  - tests7_DBD += snctests7.dbd
  - tests7_SRCS += snctests7.stt
  - tests7_LIBS += seq pv
  - PROD_HOST += sncProgram
  - sncProgram_SNCFLAGS += +m
  - sncProgram_SRCS += sncProgram.st
  - sncProgram_LIBS += seq pv
  - sncProgram_LIBS += $(EPICS_BASE_HOST_LIBS)
- The [testS7]/tests7App/src/snctests7.dbd file must include the correct "registrar(snctests7Registrar)" command.
- The startup script [testS7]/iocBoot/ioctests7/st.cmd must include the command "seq snctests7"

Control System Studio Interface

The testS7 example application is accompanied with a Control System Studio interface that runs on a client computer on the same network as the EPICS IOC. This interface is available in the IPFN ESTHER repository http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/CSS/S7EPICS_CSS/.

> "Control System Studio is an Eclipse-based collection of tools to monitor and operate large scale control systems, such as the ones in the accelerator community. It's a product of the collaboration between different laboratories and universities.", http://controlsystemstudio.org/

First CSS must be installed. Download and install the latest CSS version for the OS of the client computer. CSS requires Java to be installed.

> CSS: http://cs-studio.sourceforge.net/nsls2/nsls2.html
>
> Java: https://java.com/en/download/index.jsp

Then download the example CSS interface project from the IPFN ESTHER repository:

> http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/CSS/S7EPICS_CSS/

Import into the CSS workspace by File -> Import -> General -> Existing Projects into Workspace, then select the root directory of the downloaded project.

Now setup CSS to access the correct EPICS server. Go to Preferences -> CSS Core -> EPICS. Then add the IP of the machine running the example EPICS IOC application to the addr_list textbox. Multiple EPICS servers can be added by separating IP addresses by spaces.

Open the S7IO.opi operator interface from the navigator panel in editor mode (Right-click -> Open with -> OPI Editor). If necessary set the window perspective in Window -> Open Perspective -> Other -> Data Browser; this opens both the Navigator and Properties panels.

Inspect the interface by selecting each monitor LED, textbox or button and verify the properties panel for that element. Notice the PV Name value, which includes the PV set in the database records file, such as "S7:biI0_0".

Now open the same OPI in run mode by right-click -> Open with -> OPI Runtime, or clicking the green play symbol at the top of CSS. At this point, if everything is setup correctly and the testS7 example IOC is running on the EPICS server, then you should see the LEDs blinking as they are in the PLC. If CSS can not connect to the EPICS server, either the IP address is not set or the IOC is not running, then each interface element has a bright pink textbox saying "Disconnected".

In the example interface, all the LEDs are monitors, have the PV Names of the binary inputs (bi) records explicited in the records database of the IOC, and the ai0 and ai1 textboxes have the PV names of the corresponding analog inputs. The boolean button in the middle of the interface has the PV Name of the binary output of the Q0.2 PLC port: "S7:boQ0_2". Clicking this button will alter the state of that port, and this afects the state machine running inside the PLC.

As an exercise, add interface elements from the side Palette, such as a new boolean button, and attribute a bo PV Name to it.

Totally Integrated Automation PLC Firmware

Creating a new firmware and programming the PLC requires having a windows computer running the proprietary Siemens Totally Integrated Automation (TIA) software package. You can use the available trial or by using a purchased license (here).

After having TIA installed, download the PLC Firmware project from the IPFN ESTHER repository and open it in TIA.

> http://metis.ipfn.ist.utl.pt/svn/cdaq/ESTHER/Software/TIA/S7-1200/testS7_PLC/

This project is writen using Ladder Logic (http://en.wikipedia.org/wiki/Ladder_logic) and consists in a timed blinking of LEDs. There are two timers setup with a 500 ms interval that feed an output port that also has a LED on the S7-1200 PLC. At the input of each timer there is a condition that blocks the execution of a timer as long as the other is running. There is a condition on the Timer 1 rail that is connected to the value of the port Q0.2. Remember that this port is connected to the boolean button of the CSS interface and, as such, allows a user to stop or allow the execution of the blinking state machine on the PLC. A counter element is connected parallel to the Timer 1 and feeds its count output to a memory field. This memory field is also acessible in the records database with the PV Name "S7:memTestRead", and by the CSS interface.