EPICS Asyn Exercise 01

SendString device support using the ASYN driver to send a string on Rs232 / TCP-IP / UDP witch contain a random number and a fixed header string.

Document Number MG-001

Revision 1

2-2-2009

Distribution:
[√] Project Only
[√] All
[ ] Project Director
[ ] Technical Director
[ ] Technical Integration Manager

Availability:
[√] Public (No restriction)
[ ] Confidential/ Commercial

Revision History

Revision	Date	Prepared by	Description
0	2 February 2009	Mauro Giacchini	Original Issue

REVIEWED:

....

APPROVED:

.....

Table of Contents

1. Introduction

1.1 Identification

This exercise is a part of a training on job EPICS course.

1.2 Purpose

Write a devSup using ASYN. The devSup has to send every second a string like:

"Header String VAL:0 , random:196"

were the "Header String" is a fixed string defined in the source code, the VAL is the value of the field VAL of the LO record and the random number is a randomic number. That string may be send over RS232s or on a Ethernet port on the localhost machine.

1.3 Document Overview

Section 2 lists the names and locations of referenced documents.
Section 3 lists the hardware and software prerequisites which must be in place prior to carrying out the installation.
Section 4 defines the steps which must be carried out in order to install the software.

1.4 Abbreviations

EPICS	Experimental Physics and Industrial Control System

1.5 Definitions

None

2. Referenced Documents

2.1 External Documents

EPICS: Input/Output Controller Application Developer's Guide, Release 3.14.9

EPICS Record Reference Manual, EPICS Release 3.14

Asyn Driver

How to do serial

2.2 Web-Site References

EPICS web site

http://www.aps.anl.gov/epics/

EPICS web site at LNL:

http://www.lnl.infn.it/~epics/

ProcServer:

http://www-csr.bessy.de/control/SoftDist/procServ/

Asyn:

http://www.aps.anl.gov/epics/modules/soft/asyn/

NetCat web site:

http://netcat.sourceforge.net/

http://www.adamsinfo.com/netcat-tutorial-for-linux-windows-howto-nc/

Wireshark

http://www.wireshark.org/

3. Prerequisites

3.1 Hardware Prerequisites

There must be a minimum of one PC linux Box.

3.2 Software Prerequisites

3.2.1 Linux and EPICS

The Fedora Core 8 Linux operating system must have been previously installed. Note: the software should work on any Linux distro, we only run in on Fedora and Cent OS 5.2. Also, the Asyn v.4.9 modules driver must be installed. MEDM, ProcServer, Netcat and Wireshark may help to test the installation.

3.2.2 Application tar-ball

An application tar-ball must be available in the Training Material Repository, of the EPICS web site of LNL containing the follow file:

- *SendString_dd_mm_yy.tar.gz*

The application in splitted in two pieces: the device support and the ioc. The device support is under the "ONE" folder, and the softioc is under "TESTONE". The core software is the devONE.c. There are two devONE.c: the simplest which use only the GPIB command, and the complex which use the Asyn directives.

4. Installation

4.1 Configurations

Ensure that you have installed EPICS 3.14.9 and Asyn module.

Download and unzip the tarball.

1. Ensure that you have the appropriate EPICS environment variables setup, and a modified .../ONE/configure/RELEASE as follows (for me):

```
# EPICS_BASE usually appears last so other apps can override stuff:
EPICS_BASE=/opt/epics/base-3.14.9
ASYN=/opt/epics/modules/asyn-4.9
```

2. Ensure that you have the appropriate EPICS environment variables setup, and a modified .../TESTONE/configure/RELEASE as follows:

```
# EPICS_BASE usually appears last so other apps can override stuff:
EPICS_BASE=/opt/epics/base-3.14.9
ASYN=/opt/epics/modules/asyn-4.9
ONE=/home/giacchin/playground/ONE
```

then clean up and recompile both: the devSupport (ONE) and the application (TESTONE)

```
$ cd ../
$ make clean all
```

4.2 SendString device support module SetUp

The SendString device software can send the string over: RS232c, TCP/IP, UDP or UDP broadcast. Choose your option as follow:

1. Change to the directory which contains the startup commands:

```
$ cd TESTONE/iocBoot/iocTESTONE
```

2. Adjust the st.cmd setting up your connections type: .

```
## TCP/IP protocol
drvAsynIPPortConfigure("L0","127.0.0.1:5001",0,0,0)

## UDP protocol
#drvAsynIPPortConfigure("L0","127.0.0.1:5001 UDP",0,0,0)

## UDP broadcast protocol
#drvAsynIPPortConfigure("L0","192.68.0.255:5001 UDP*",0,0,0)

## RS232c protocol and serial port setup
#drvAsynSerialPortConfigure("L0","/dev/ttyS0",0,0,0)
#asynSetOption("L0", -1, "baud", "9600")
#asynSetOption("L0", -1, "bits", "8")
#asynSetOption("L0", -1, "parity", "none")
#asynSetOption("L0", -1, "stop", "1")
#asynSetOption("L0", -1, "clocal", "Y")
#asynSetOption("L0", -1, "crtscts", "N")
```

3. Save the modified st.cmd file.
4. Start the application

```
$ ../../bin/linux-x86/TESTONE st.cmd
```

5. Check the output from the “make” command for errors. In particular, check for error messages that might indicate unsupported device types or record types.

6.

4.3 Post-Installation Steps

There are no post-installation steps required for the SendString driver itself. However, you may have a look to the /etc/services to verify which one port is not in use from other service if you plan to use the ethernet port.

4.4 StartUp a Listener Server

First of all you will have to make a listener server on the selected port. To do this you can use three methods: use NetCat _or_ use ProcServer _or_ use the IPEchoServer.

4.4.1 NetCat

Start-up the NetCat listener:

```
[giacchin@epics1 buttami]$ nc -l 127.0.0.1 -p 5001
Header String VAL:0 , random:195Header String VAL:0 ,
random:101Header String VAL:0 , random:117Header String VAL:0 ,
random:113Header String VAL:0 , random:159Header String VAL:0 ,
random:148Header String VAL:0 , random:107Header String VAL:0 ,
random:183Header String VAL:0 , random:165Header String VAL:0 ,
random:154
```

4.4.1 ProcServer

Start-up the ProcServer, and connect using a telnet to the selected port:

```
[giacchin@epics1 buttami]$ procServ 5001 /usr/bin/tail -f
[giacchin@epics1 buttami]$ telnet localhost 5001
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
@@@ Welcome to the procServ process server (procServ Version 2.3.0)
@@@ Use ^X to kill the child, auto restart is ON, use ^T to toggle
auto restart
@@@ procServ server PID: 32122
@@@ Server startup directory: /home/giacchin/playground/buttami
@@@ Child startup directory: (null)
@@@ Child started as: /usr/bin/tail
@@@ Child "/usr/bin/tail" PID: 32123
@@@ procServ server started at: Tue Feb 10 11:07:48 2009
```

```
@@@ Child "/usr/bin/tail" started at: Tue Feb 10 11:07:49 2009
@@@ 1 user(s) and 0 logger(s) connected (plus you)
Header String VAL:0 , random:169Header String VAL:0 ,
random:136Header String VAL:0 , random:125
```

4.4.1 IpEchoServer

Start-up the IpEchoServer, available ont he distribution of asyn:

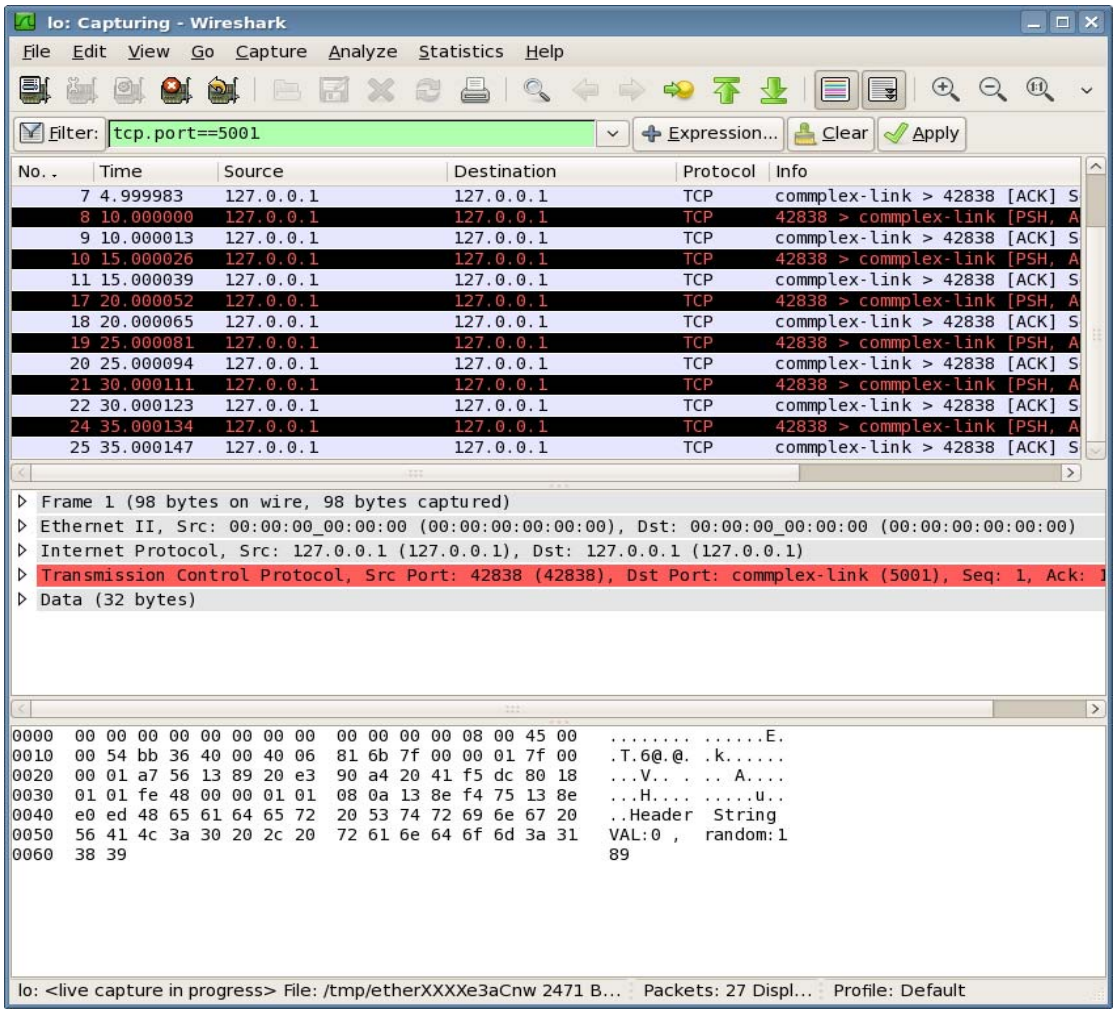
```
[giacchin@epics1 ioctestIPServer]$ cd /opt/epics/modules/asyn-
4.9/iocBoot/ioctestIPServer
[giacchin@epics1 ioctestIPServer]$ ../../bin/linux-x86/testIPServer
st.cmd
< envPaths
epicsEnvSet(ARCH,"linux-x86")
epicsEnvSet(IOC,"ioctestIPServer")
epicsEnvSet(TOP,"/opt/epics/modules/asyn-4.9")
epicsEnvSet(SNCSEQ,"/opt/epics/modules/seq-2.0.11")
epicsEnvSet(EPICS_BASE,"/opt/epics/base-3.14.9")
dbLoadDatabase("../../dbd/testIPServer.dbd")
testIPServer_registerRecordDeviceDriver(pdbbase)
#The following command starts a server on port 5001
drvAsynIPServerPortConfigure("P5001","localhost:5001",2,0,0,0)
drvAsynIPServerPortConfigure("P5002","localhost:5002",1,0,0,0)
#asynSetTraceFile("P5001",-1,"")
#asynSetTraceMask("P5001",-1,0xff)
#asynSetTraceIOMask("P5001",-1,0x2)
dbLoadRecords("../../db/testIPServer.db", "P=testIPServer:")
iocInit()
Starting iocInit
############################################################################
#######
## EPICS R3.14.9 $R3-14-9$ $2007/02/05 16:31:45$
## EPICS Base built Jul  9 2008
############################################################################
#######
iocInit: All initialization complete
ipEchoServer("P5001")
seq("ipSNCServer", "P=testIPServer:, PORT=P5002")
```

```
SEQ Version 2.0.11: Sat Feb  7 10:21:17 2009
Spawning state program "ipSNCServer", thread 0x90f78e8: "ipSNCServer"
epics>
```

The above starts the a ioc server which echo the string received on the port 5001: the devONE.c has the option to print on stdout what the server has replyed.

4.4.1 Wireshark

Start-up Wireshark capturing the traffic on “lo” (localhost: 127.0.0.1), during the capturing make filter on the selected port (tcp.port==5001):



that show you are sending the string: 'Header String VAL:0 random 89' on 127.0.0.1:5001

1.2 Using SendString ASYN DevSupport

Goes into the ioc boot directory of the TESTONE application, and boot the ioc.

If everything works correctly you will have:

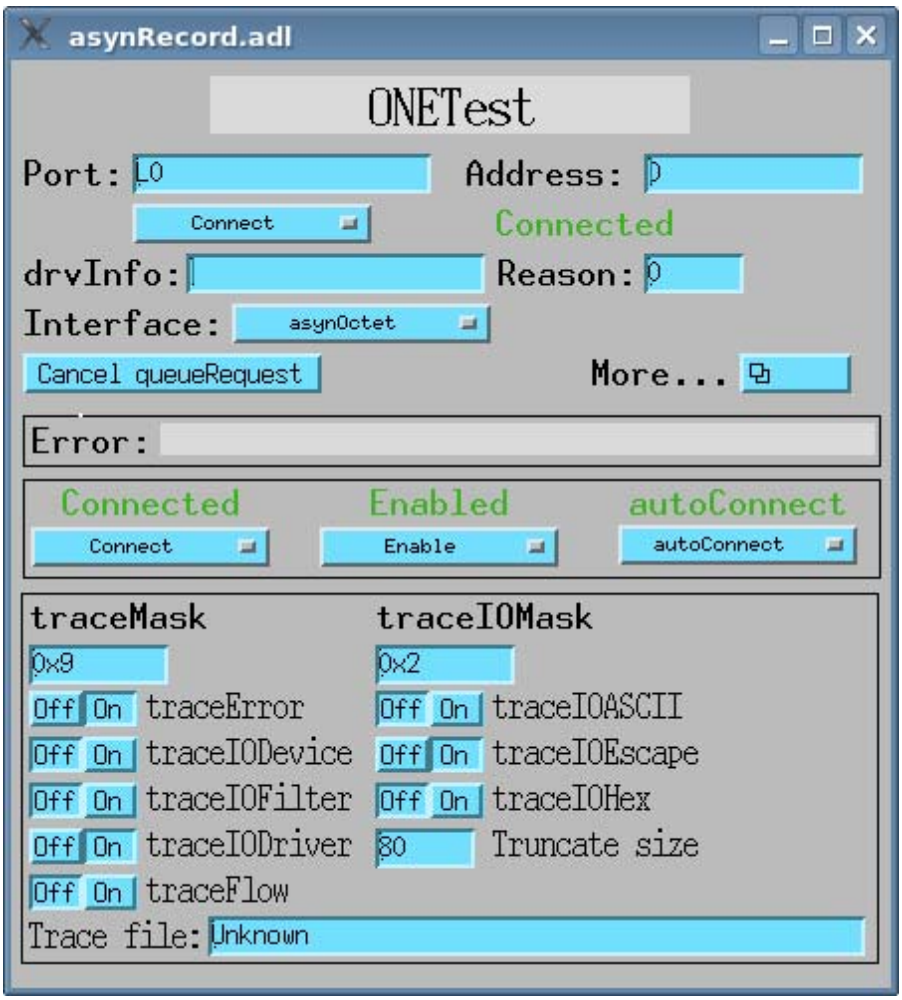
```
[giacchin@epics1 iocTESTONE]$ ../../bin/linux-x86/TESTONE st.cmd
#!../../bin/linux-x86/TESTONE
## You may have to change TESTONE to something else
## everywhere it appears in this file
< envPaths
epicsEnvSet(ARCH,"linux-x86")
epicsEnvSet(IOC,"iocTESTONE")
epicsEnvSet(TOP,"/home/giacchin/playground/TESTONE")
epicsEnvSet(EPICS_BASE,"/opt/epics/base-3.14.9")
epicsEnvSet(ASYN,"/opt/epics/modules/asyn-4.9")
epicsEnvSet(ONE,"/home/giacchin/playground/ONE")
cd /home/giacchin/playground/TESTONE
## Register all support components
dbLoadDatabase("dbd/TESTONE.dbd",0,0)
TESTONE_registerRecordDeviceDriver(pdbbase)
cd /home/giacchin/playground/ONE
dbLoadRecords("db/devONE.db","P=ONE:,R=,L=0,A=0")
## Load record instances
cd /opt/epics/modules/asyn-4.9
dbLoadRecords("db/asynRecord.db","P=ONE,R=Test,PORT=L0,ADDR=0,IMAX=0,OMAX
=0")
## TCP/IP protocol
drvAsynIPPortConfigure("L0","127.0.0.1:5001",0,0,0)
## UDP protocol
#drvAsynIPPortConfigure("L0","127.0.0.1:5001 UDP",0,0,0)
## UDP broadcast protocol
#drvAsynIPPortConfigure("L0","192.68.0.255:5001 UDP*",0,0,0)
## RS232c protocol and serial port setup
#drvAsynSerialPortConfigure("L0","/dev/ttyS0",0,0,0)
#asynSetOption("L0", -1, "baud", "9600")
#asynSetOption("L0", -1, "bits", "8")
#asynSetOption("L0", -1, "parity", "none")
```

```
#asynSetOption("L0", -1, "stop", "1")
#asynSetOption("L0", -1, "clocal", "Y")
#asynSetOption("L0", -1, "crtscts", "N")
cd /home/giacchin/playground/TESTONE/iocBoot/iocTESTONE
iocInit()
Starting iocInit
############################################################################
#
## EPICS R3.14.9 $R3-14-9$ $2007/02/05 16:31:45$
## EPICS Base built Jul  9 2008
############################################################################
#
iocInit: All initialization complete
asynSetTraceMask("L0",-1,0x9)
asynSetTraceIOMask("L0",-1,0x2)
epics> Received reply from the echoServer :
2009/02/10 09:36:09.401 127.0.0.1:5001 write 32
Header String VAL:0 , random:195
Received reply from the echoServer :
2009/02/10 09:36:14.401 127.0.0.1:5001 write 32
Header String VAL:0 , random:125
```

I've prepared a bash script to start the AsynRecord

```
$ ../TESTONE/medm ./tester.sh
```

the follow are going to appear on your desktop:



With that medm screen you can modify on flight the traceMask and traceIOMask for example. The tester.sh scrip is based on only one line:

```
medm -x -macro "P=ONE,R=Test,PORT=L0,ADDR=0" /opt/epics/modules/asyn-4.9/medm/asynRecord.adl
```