EPICS Training @ PSI

Controls Section PSI

2010

Dr. Dirk Zimoch and Dr. Elke Zimoch

PAUL SCHERRER INSTITUT

PSI

EPICS

Acknowledgment

Some slides in this talk came form the
„Getting started with EPICS“ lecture series of APS.
My special thanks go to

Ned Arnold

and

John Maclean

from the Advanced Photon Source.
The original talks can be found on
www.aps.anl.gov/epics/docs/GSWE.php

Contents

- Introduction to EPICS
- What are Records
- Set up an IOC on your PC
- Overview of S7 PLC driver
- Include S7 PLC communication into your IOC
- IOC monitoring

What is EPICS?

EPICS is an abbreviation for:
Experimental **P**hysics and **I**ndustrial **C**ontrol **S**ystem

EPICS is:

- A collaboration
- A tool kit
- A control system architecture

The History

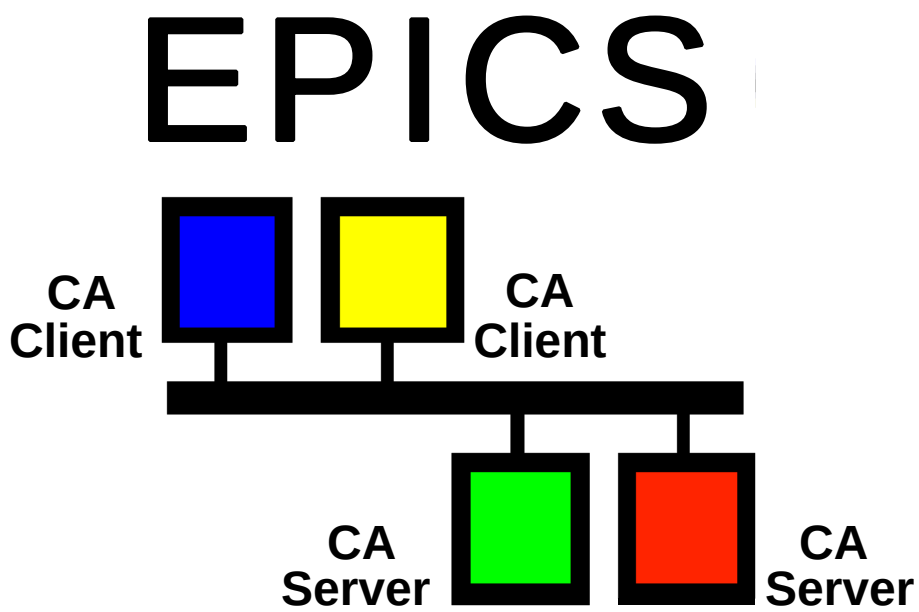
- In1989 started a collaboration between Los Alamos National Laboratory (GTA) and Argonne National Laboratory (APS)
  (Jeff Hill, Bob Dalesio & Marty Kraimer)

GTA: Ground Test Accelerator
APS: Advanced Photon Source

- More than 150 licenses agreements were signed, before EPICS became Open Source in 2004
- Team work on problems, for example over “Tech Talk” mailing list
- Collaborative efforts vary
  - Assistance in finding bugs
  - Share tools, schemes, and advice

The architecture of EPICS

Network based Client/Server Model (hence the EPICS logo)



A **Server** provides information and service

A **Client** uses the service or asks for the information

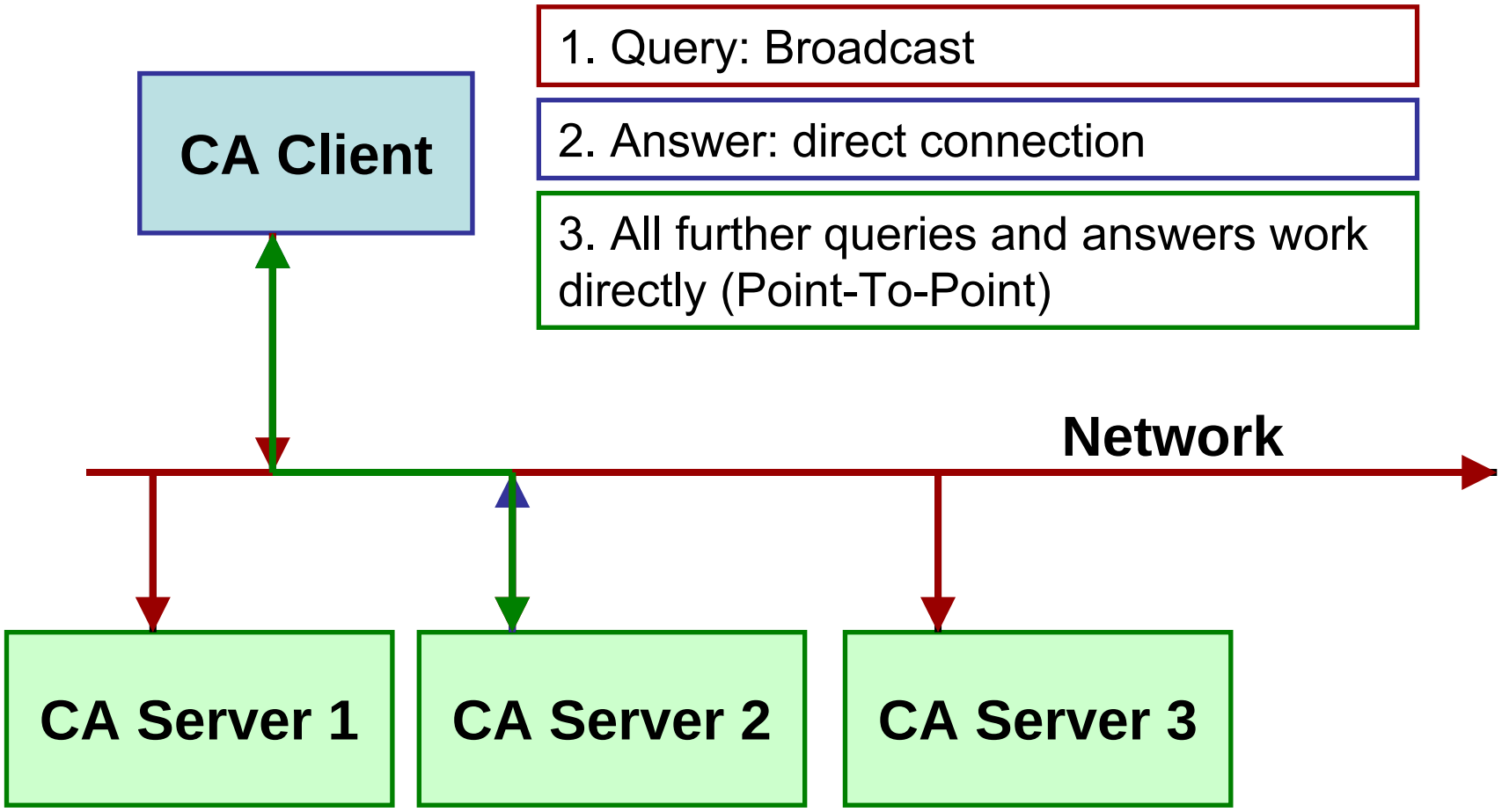
For EPICS, client and server refer to their Channel Access role

i.e. Channel Access Client and Channel Access Server

What is Channel Access

- A protocol how to transfer data
- A single data unit is called **Process Variable**
- A Process Variable has a unique name, which is used to refer to the data
- The detailed operation of Channel Access is unimportant for most programmers (it already works…)
- Channel Access is not dependent on a single programming language

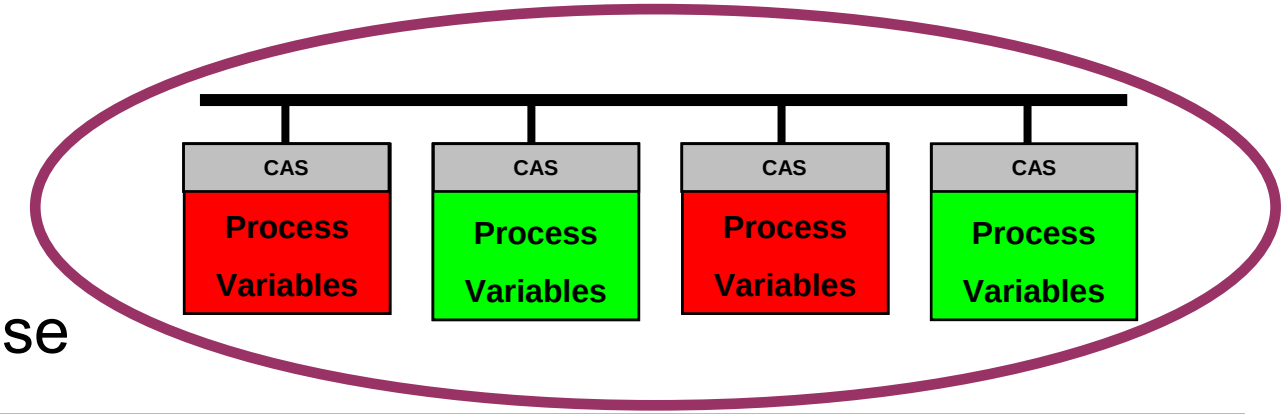
Channel Access network flow



Control System Architecture

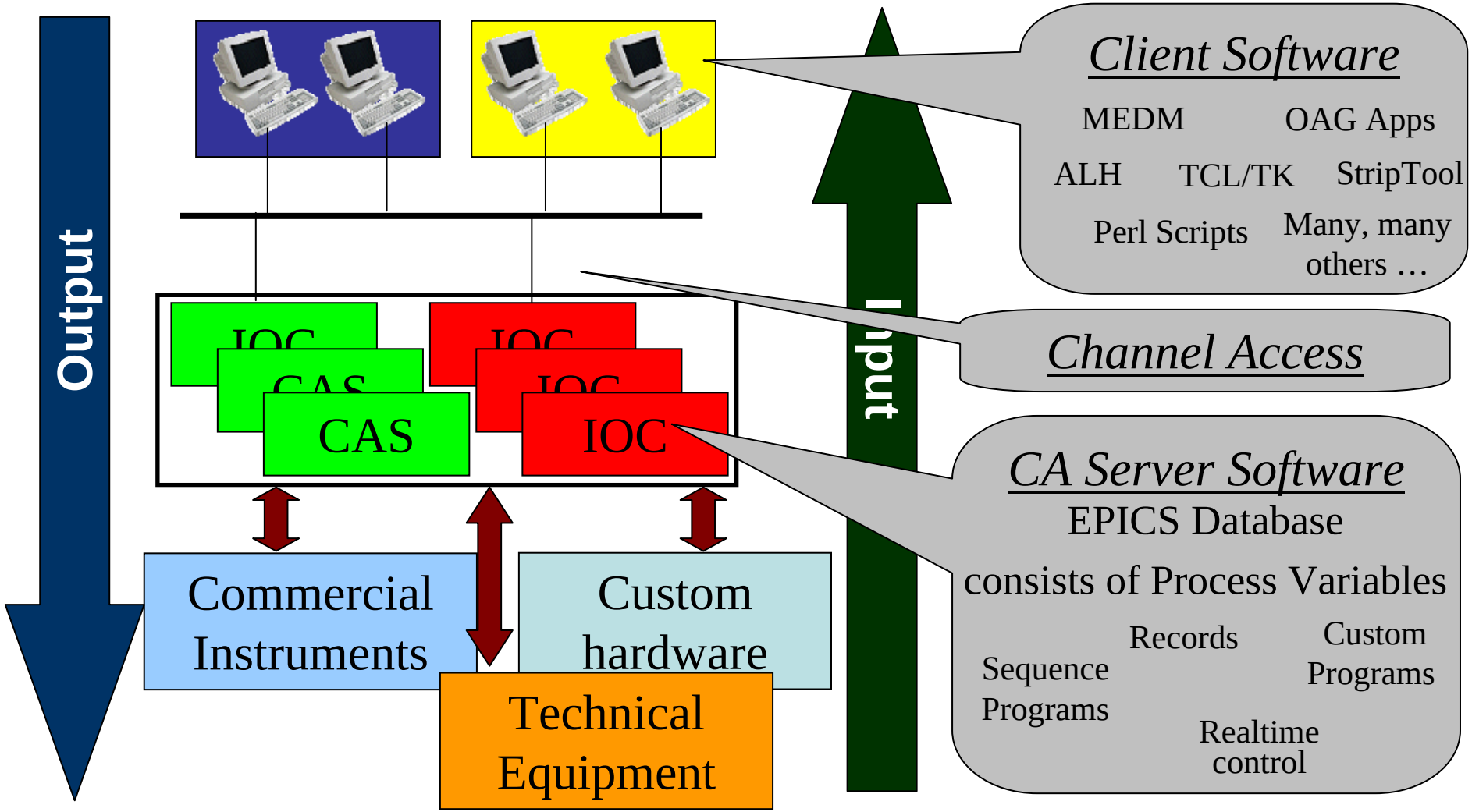
PAUL SCHERRER INSTITUT

- A network based “Client/Server” Model, whose smalest data set is a Process Variable
- The Channel Access Protocol defines, what data (Process Variable) is transferred between server and client
- The entire set of Process Variables establish a *Distributed Real-time Database* of machine status, information and control parameters

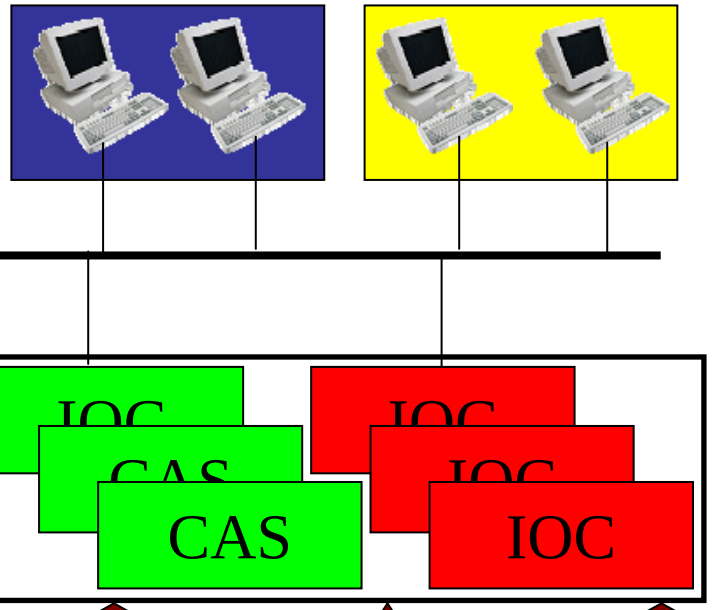


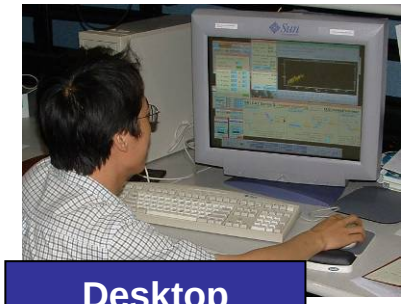
EPICS Database

Standard Model of EPICS

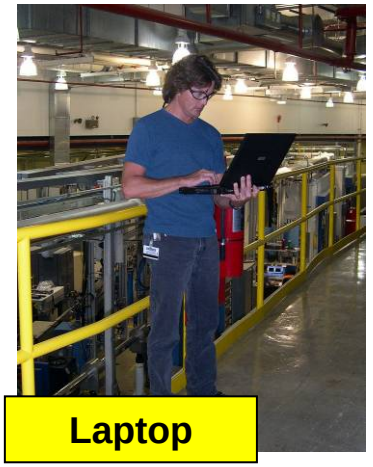


Typical Realisation

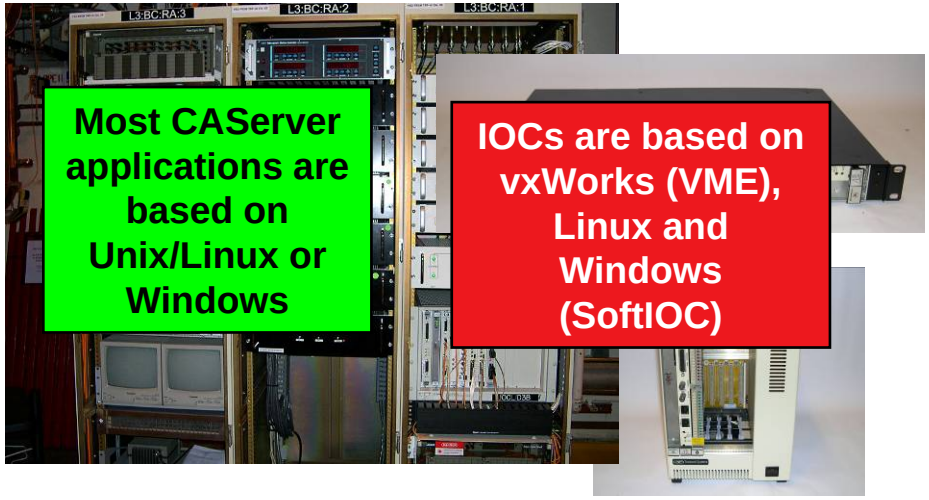




Desktop



Laptop



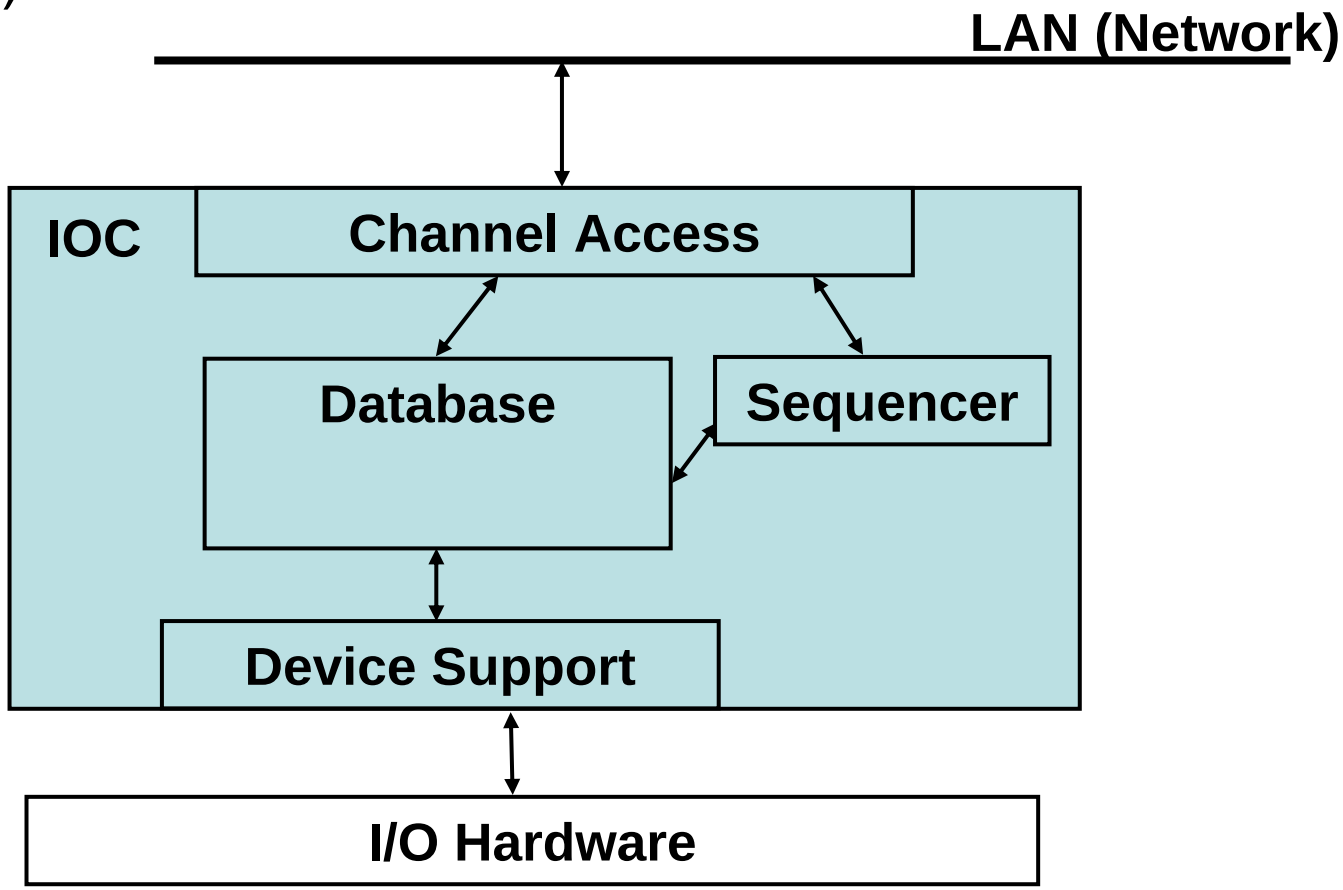
What is an IOC

IOC means
Input **O**utput **C**ontroller

- A special CA Server and CA Client
- A computer running "*IOC Core*"
- This computer may be:
  - VME based, operating system vxWorks (only possibility up to 3.14) or RTEMS
  - PC, operating system Windows, Linux, RTEMS
  - Apple, operating system OSX
  - UNIX Workstation, operating system Solaris
- An IOC normally is connected to input and/or output hardware
- An EPICS control system is based on at least one Channel Access Server (normally an IOC)
- An IOC runs a record database, which defines what this IOC is doing

Inside an IOC

The major software components of an IOC (IOC Core)

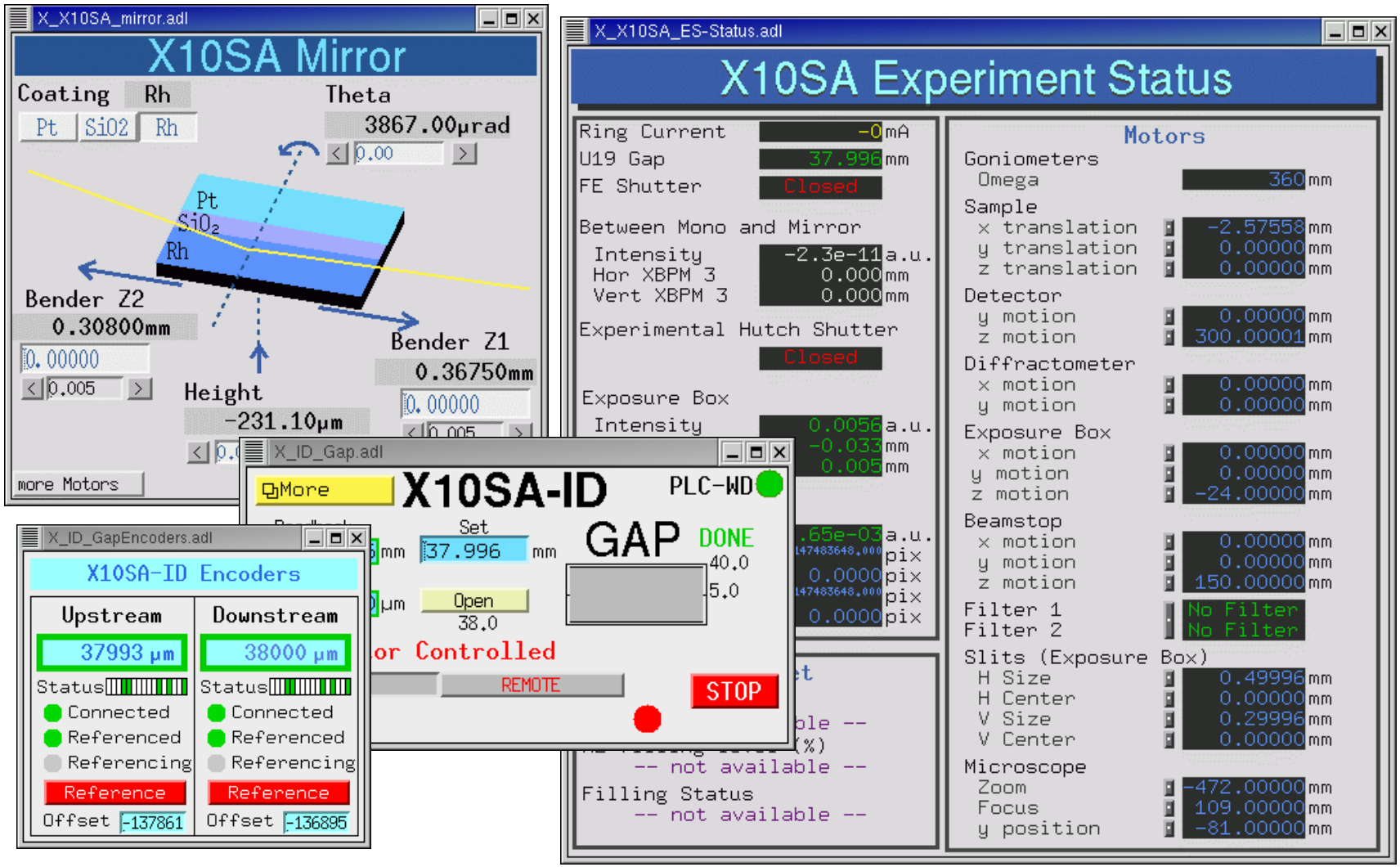


Some CA Clients

(from the EPICS Website - incomplete)

- ALH: Alarm Handler
- BURT: Backup and Restore Tool
- CASR: Host-based Save/Restore
- CAU: Channel Access Utility
- Channel Archiver (SNS)
- Channel Watcher (SLAC)
- EDM: Extensible Display Manager (ORNL)
- JoiMint: Java Operator Interface and Management INtegration Toolkit (DESY)
- Knobs: Knob Manager und KnobConfig, eine Schnittstelle zu SunDials
- MEDM: Motif Editor und Display Manager
- StripTool: Strip-chart Plotting Tool
- and many more …

Examples: medm



CA Commands – command line Client

- Read a PV named <NAME>

  `caget NAME`

- Write a PV named <NAME>

  `caput NAME value`

- Get information about that Record

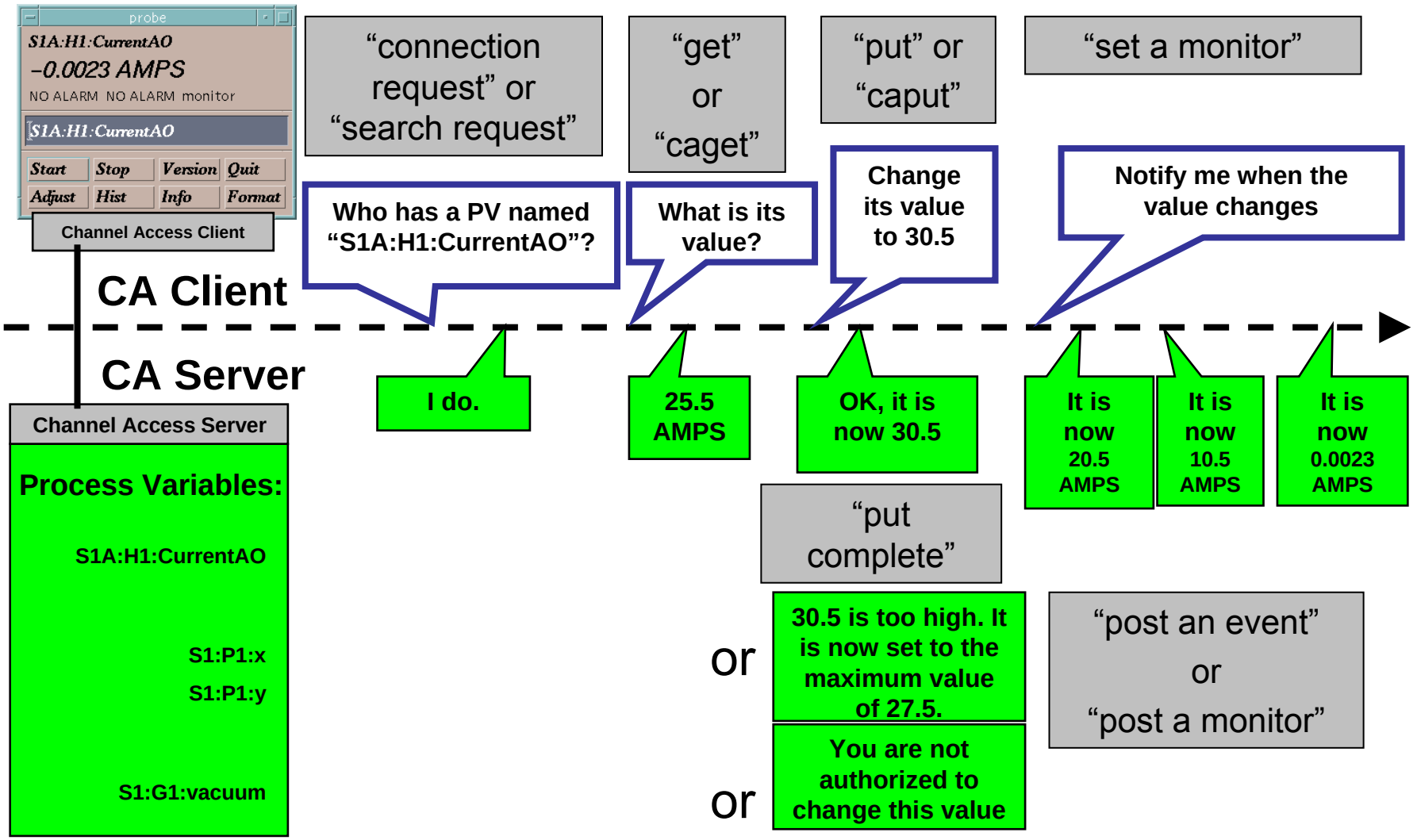
  `cainfo NAME`

- Start a monitor

  `camonitor NAME`

  (Cancel with [Ctrl] + [c])

Channel Access Commands



Contents

- Introduction to EPICS
- What are Records
- Set up an IOC on your PC
- Overview of S7 PLC driver
- Include S7 PLC communication into your IOC
- IOC monitoring

Close look at a Measured Value

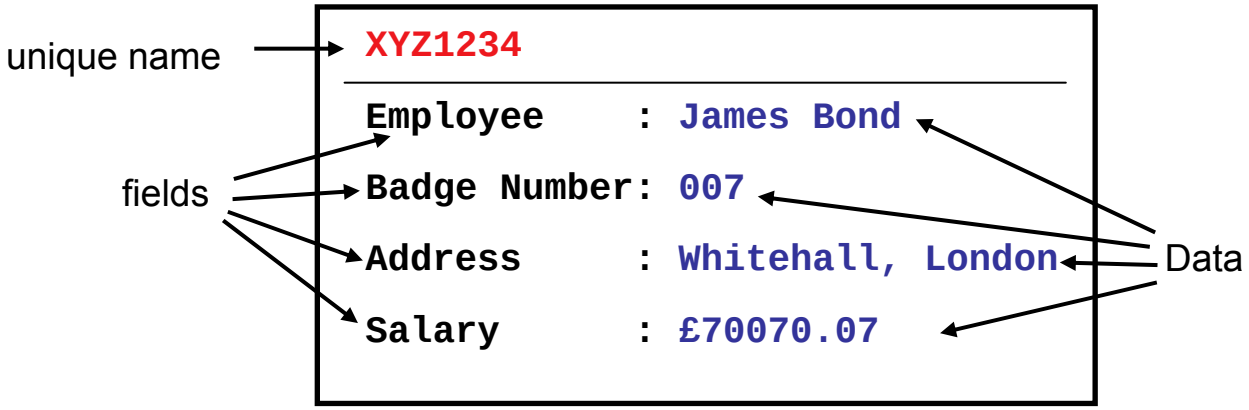
Value with a data type	**295,5**
with a unit	**mA**
with a time stamp	**17.2.2005 14:21:16**
with a severity (alarm state)	**NO_ALARM**
with technical limits	**0 bis 400**
with graphical limits	**0 bis 370**
with a description	**„Beam current in SR“**

A measured value is an object with multiple related information

What are Records?

A Record is an object with

- a unique name
- properties (fields) that contain information (data)
- the ability to perform actions on that data



From Measured Value to Record

record (ai, "ARIDI-PCT:CURRENT") {

A value with a data type

with a unit

with a time stamp

with a severity (alarm status)

with technical limits

with graphical limits

with a description „**Beam current in SR**“

}

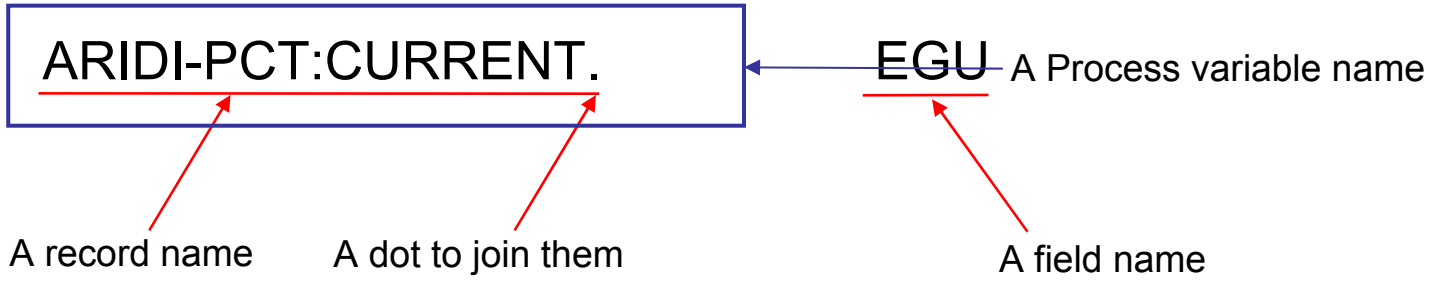
field (EGU, "**mA**")
field (EGUF, "**400**")
field (EGUL, "**0**")
field (HOPR, "**370**")
field (LOPR, "**0**")
field (DESC, "**Beam current in SR**")
field (DTYP, "**HY8401**")
field (INP, "**#C3 S0 @**")

295,5 **ai** **mA** **17.2.2005 14:21:16** **NO_ALARM** **0 bis 400** **0 bis 370**

Readback from hardware:

A Process Variable name

- A PV name is comprised of two parts
  - The record name, and
  - the name of a field belonging to that record
- For example…



- Note that if no field name is given, Channel Access will default to using the .VAL field
- i.e. "`ARIDI-PCT:CURRENT`" = "`ARIDI-PCT:CURRENT.VAL`"

What do Records do?

- Records are active, they do things
  - Get data from other records or from hardware
  - Perform calculations
  - Check values are in range and raise alarms
  - Put data to other records or to hardware
  - Activate or disable other records
  - Wait for hardware signals (interrupts)
- What a record does depends upon its type and the values in its fields
- A wide range of records have already been created
- New record types can be added to a new application as needed
- A record does nothing until it is *processed*

Some Record Types

- Analog in
- Analog out
- Binary in
- Binary out
- Calculation
- Calculation out
- Compression
- Data fanout
- Event
- Fanout
- Histogram
- Motor
- Multi bit binary input
- Multi bit binary output
- PID control
- Pulse counter
- Pulse delay
- Scan
- Select
- Sequence
- String in
- String out
- Subarray
- Subroutine
- Waveform

IOC view of a Record

```
record(ao,"DemandTemp") {
    field(DESC,"Temperature")
    field(ASG,"")
    field(SCAN,"Passive")
    field(PINI,"NO")
    field(PHAS,"0")
    field(EVNT,"0")
    field(DTYP,"VMIC 4100")
    field(DISV,"1")
    field(SDIS,"")
    field(DISS,"NO_ALARM")
    field(PRIO,"LOW")
    field(FLNK,"")
    field(OUT,"#C0 S0")
    field(OROC,"0.0e+00")
    field(DOL,"")
    field(OMSL,"supervisory")
    field(OIF,"Full")
    field(PREC,"1")
    field(LINR,"NO CONVERSION")
    field(EGUF,"100")
    field(EGUL,"0")
    field(EGU,"Celcius")
    field(DRVH,"100")
    field(DRVL,"0")
    field(HOPR,"80")
    field(LOPR,"10")
    field(HIHI,"0.0e+00")
    field(LOLO,"0.0e+00")
    field(HIGH,"0.0e+00")
    field(LOW,"0.0e+00")
    field(HHSV,"NO_ALARM")
    field(LLSV,"NO_ALARM")
    field(HSV,"NO_ALARM")
    field(LSV,"NO_ALARM")
    field(HYST,"0.0e+00")
    field(ADEL,"0.0e+00")
    field(MDEL,"0.0e+00")
    field(SIOL,"")
    field(SIML,"")
    field(SIMS,"NO_ALARM")
    field(IVOA,"Continue normally")
    field(IVOV,"0.0e+00")
}
```

The Record Reference Manual

- In English (American style)
- Explains database concepts and record types
- All fields are explained for common record types

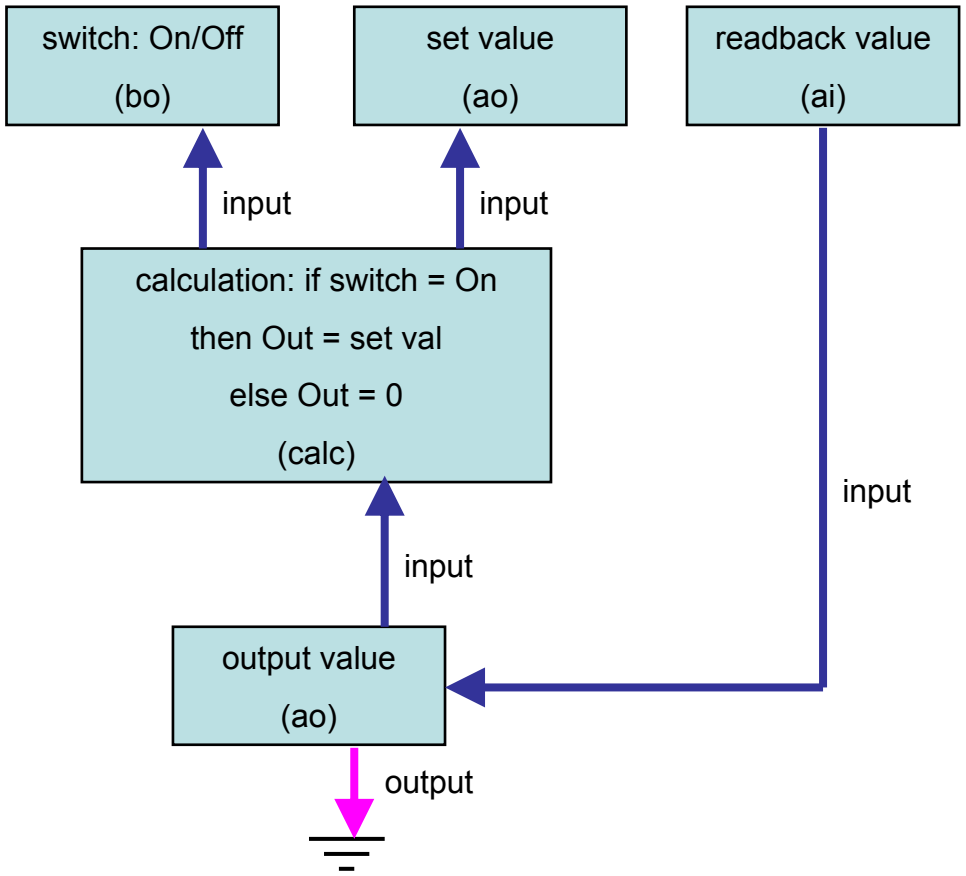
  Test: What does the field HSV of an ai Record
- Common fields for all records are described in the first part
- The explained records are part of the EPICS base release
- More Records exist. Find their documentation in the internet

Record Processing

- Record processing can be periodic or event driven
- Periodic: Standard scan rates are…
  - 10, 5, 2, 1, 0.5, 0.2 and 0.1 seconds
  - Custom scan rates can be configured up to speeds allowed by operating system and hardware
- Event driven: Events include
  - Hardware interrupts
  - Request from another record via links
  - EPICS Events
  - Channel Access Puts

Example

Power Supply: switch on and off and read and set voltage

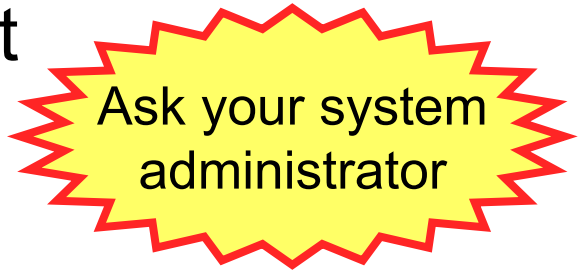


Contents

- Introduction to EPICS
- What are Records
- Set up an IOC on your PC
- Overview of S7 PLC driver
- Include S7 PLC communication into your IOC
- IOC monitoring

Steps towards your IOC (1)

- What you need before you start
  - Build tools: gcc, make, tar, gzip
  - Perl scripting language
  - Development packages of X11 and Motif
  - A text editor (e.g. nedit, kwrite, emacs)



- Browse EPICS home http://aps.anl.gov/epics
  - Get latest Base
  - Get Extensions: Config files 3.14, medm
- From PSI EPICS page http://epics.web.psi.ch
  - Get s7plc driver

Steps towards your IOC (2)

- Extract Base and Extension packages

```
tar xvfz baseR3.14.12-rc1.tar.gz
tar xvfz extensionsTop_20070703.tar.gz
tar xvfz medm3_1_5.tar.gz –C extensions/src
```

- Build Base

```
export EPICS_HOST_ARCH=linux-x86
cd base-3.14.12-rc1
make
export PATH=~/base-3.14.12-rc1/bin/linux-x86:$PATH
```

- Test the installation

```
softIoc
```

Steps towards your IOC (3)

- Build Extensions (here: medm)

```
cd extensions/configure
```

  – Edit the RELEASE file

```
nedit RELEASE
```

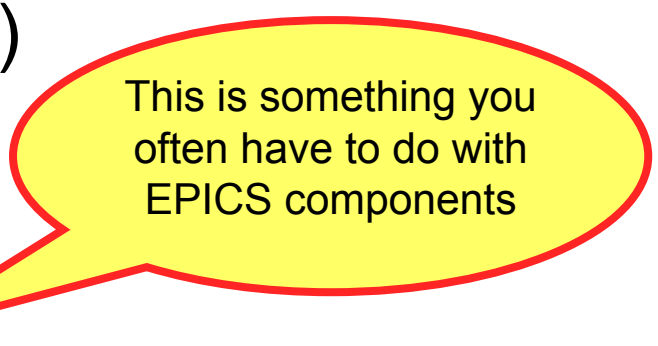
    - Set correct path to EPICS Base, e.g.

      EPICS_BASE=/home/epics/base-3.14.12-rc1

```
make
cd ../src/medm3_1_5
make
export PATH=~/extensions/bin/linux-x86:$PATH
```

- Test the installation

```
medm
```



Steps towards your IOC (4)

- Make environment variables permanent
  - Edit ~/.basrc and add variables:

```
export EPICS_HOST_ARCH=linux-x86
export EPICS_BASE=/home/epics/base-3.14.12-rc1
export EPICS_EXTENSIONS=/home/epics/extensions
export PATH=$EPICS_BASE/bin/$EPICS_HOST_ARCH:\
  $EPICS_EXTENSIONS/bin/$EPICS_HOST_ARCH:$PATH
```

- EPICS can also be installed centrally
  - e.g. /opt/epics/base-3.14.12 and /opt/epics/extensions
  - or /usr/local/epics/…

Steps towards your IOC (5)

- Ready to start !
- We will start with the power supply simulation
- Go to your home directory, create a new directory for your IOC and create some files. You will need 2 files for the beginning.

```
cd
mkdir myFirstIoc
cd myFirstIoc
nedit st.cmd PowerSupplySimu.db &
```

Create your records from the Example

Comment

```
# interface records
record (bo, "PowerSupply1:Switch") {
      field (DESC, "Switch On or Off")
      field (ZNAM, "Off")
      field (ONAM, "On")
      field (FLNK, "PowerSupply1:Calculation")  # after switch set calc output
}
record (ao, "PowerSupply1:SetValue") {
      field (DESC, "Nominal set value")
      field (EGU,  "V")
      field (PREC, "2")
      field (FLNK, "PowerSupply1:Calculation")  # after switch set calc output
}
record (ai, "PowerSupply1:Readback") {
      field (DESC, "Actual readback from HW")
      field (EGU,  "V")
      field (PREC, "2")
      field (INP,  "PowerSupply1:OutputValue")
      field (SCAN, ".1 second")
}
```

Create your records from the Example 2

```
# private records (do not use them on clients)

record (calc, "PowerSupply1:Calculation")
{
      field (DESC, "Combine Switch and SetValue")
      field (INPA, "PowerSupply1:Switch")
      field (INPB, "PowerSupply1:SetValue")
      field (CALC, "A=0?0:B")  # if switch=Off then Null else setValue
      field (FLNK, "PowerSupply1:OutputValue")  # after calculation write output
}

record (ao, "PowerSupply1:OutputValue")
{
      field (DESC, "Value send to HW")
      field (EGU,  "V")
      field (PREC, "2")
      field (DOL,  "PowerSupply1:Calculation")  # get the output value
      field (OMSL, "closed_loop")
}
```

Startup script st.cmd

st.cmd - /afs/psi.ch/user/z/zimoch_e/playground/TrainingMaterial/

File Edit Search Preferences Shell Macro Windows Help

/afs/psi.ch/user/z/zimoch_e/playground/TrainingMaterial/st.cmd byte 246 of 246 L: 14 C: 0

```
 1
 2 # Load drivers and dbd files ==============
 3
 4 # Load your record files ==================
 5
 6 dbLoadDatabase("PowerSupplySimu.db")
 7
 8 # this is a comment...
 9 #dbLoadTemplate("PowerSupplySimu.subs")
10
11 # start the EPICS IOC =====================
12
13 iocInit
14
```

Start your IOC

- In the directory where you have your
  - PowerSupplySimu.db file
  - the st.cmd file
- use the command

```
softIoc st.cmd
```

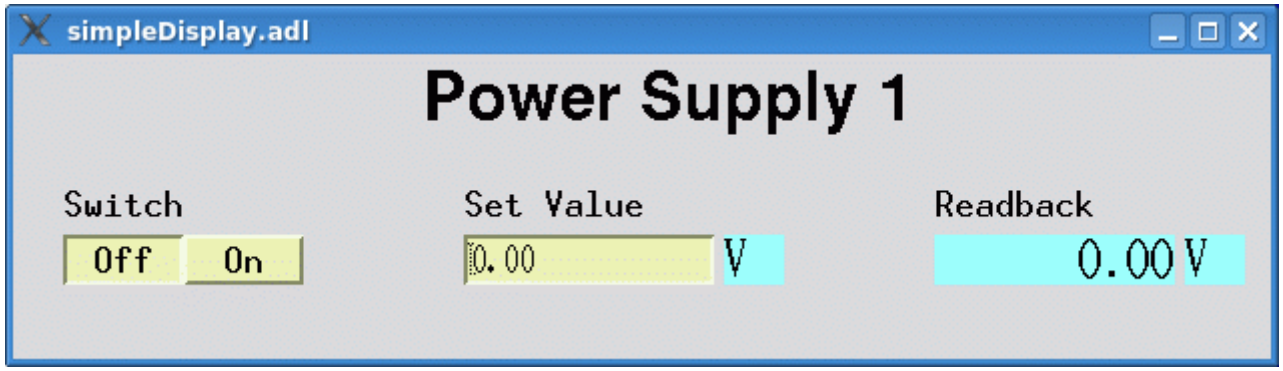
Read your records from Client side

- From the command line:

```
caget PowerSupply1:Readback
caput PowerSupply1:SetValue 6.3
caput PowerSupply1:Switch On
camonitor PowerSupply1:Readback
```

- Or use a GUI (medm):

```
medm simpleDisplay.adl
```



Templates and substitutions

- Problem: Imagine you have a second power supply
  There are two possible solutions:
  1. Copy the db file and change all names
  2. Change from db files to a combination of substitution and template files

  ... We will show the second approach

substitution file

PAUL SCHERRER INSTITUT

PowerSupplySimu.subs - /afs/psi.ch/user/z/zimoch_e/playground/TrainingMate

File Edit Search Preferences Shell Macro Windows Help

/afs/psi.ch/user/z/zimoch_e/playground/TrainingMaterial/PowerSupplySimu.subs byte 0 of 212 L: 1 C: 0

```
 1 #-----------------------------------------------------
 2 # records for EPICS Training at 19. Nov. 2010
 3
 4 file PowerSupplySimu.template {
 5       {
 6       Name = PowerSupply1
 7       }
 8       {
 9       Name = PowerSupply2
10       }
11 }
12
```

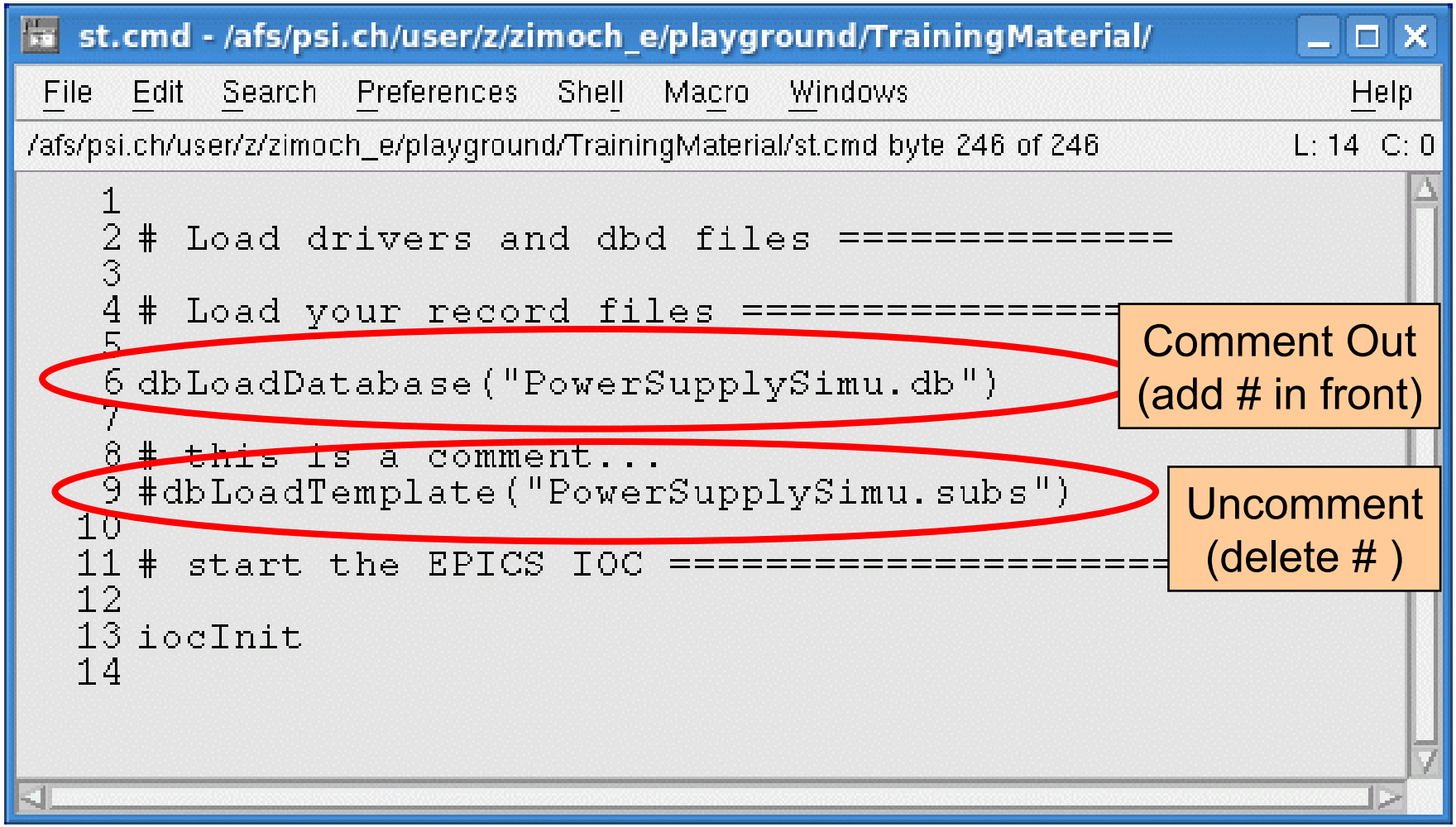
template file

```
# interface records
record (bo, "$(Name):Switch") {
    field (DESC, "Switch On or Off")
    field (ZNAM, "Off")
    field (ONAM, "On")
    field (FLNK, "$(Name):Calculation")  # after switch set calc output
}
record (ao, "$(Name):SetValue") {
    field (DESC, "Nominal set value")
    field (EGU,  "V")
    field (PREC, "2")
    field (FLNK, "$(Name):Calculation")  # after switch set calc output
}
record (ai, "$(Name):Readback") {
    field (DESC, "Actual readback from HW")
    field (EGU,  "V")
    field (PREC, "2")
    field (INP,  "$(Name):OutputValue")
    field (SCAN, ".1 second")
}
```

template file

```
# private records (do not use them on clients)

record (calc, "$(Name):Calculation")
{
    field (DESC, "Combine Switch and SetValue")
    field (INPA, "$(Name):Switch")
    field (INPB, "$(Name):SetValue")
    field (CALC, "A=0?0:B")  # if switch=Off then Null else setValue
    field (FLNK, "$(Name):OutputValue")  # after calculation write output
}

record (ao, "$(Name):OutputValue")
{
    field (DESC, "Value send to HW")
    field (EGU,  "V")
    field (PREC, "2")
    field (DOL,  "$(Name):Calculation")  # get the output value
    field (OMSL, "closed_loop")
}
```

Startup script st.cmd



Contents

- Introduction to EPICS
- What are Records
- Set up an IOC on your PC
- Overview of S7 PLC driver
- Include S7 PLC communication into your IOC
- IOC monitoring

Theory of Operation

- PLC and IOC exchange blocks of PVs over TCP
- Blocks can have arbitrary length and layout.

Example:

```
offs pv          type
  0  status      (16 bits)
  2  voltage     (long int)
  6  temperature (float)
 10  message     (string[12])
 22
```

- PLC sends data periodically (e.g. 0.1 sec)
- IOC sends data on change
- **Only complete blocks are transmitted.**

Input records

- SCAN should be “I/O Intr”.
  - Record processes whenever PLC sends new data.
- On any communication error the connection is closed and reopened
  - Records get INVALID alarm
- Connection monitor record
  - Special DTYP “S7plc stat” for bi record.
  - Connection monitor: 1 when connected.
  - No alarm by driver.

Output records

- PINI should be “YES”.
  - Initializes output block before first data is sent.
  - Otherwise uninitialized values will be sent as 0.
- Driver looks for changes periodically.
  - All changes during one interval are collected.
  - If nothing has changed, nothing is sent.
  - Limits network traffic but adds latency.

Driver setup

- Give the PLC a name
- Ask programmer of PLC for
  - IP address and port
  - Byte order: big endian or little endian
  - Block size and send period PLC→IOC
  - Block size and max latency IOC→PLC
- Configure driver in startup script:

```
s7plcConfigure ("PlcName", "IpAddress", TcpPort,
PlcToIocSize, IocToPlcSize, bigEndian,
ReceiveTimout_ms, SendPeriod_ms)
```

big=1
little=0

approx
x 5

Example

- PLC “dev-x” at address 192.168.0.10
- TCP server port 2000
- 22 byte input from PLC approx every 100 msec
  - allow factor 5 for timeout → 500 msec
- 2 byte output to PLC with latency 100 msec
  - all outputs within 100 msec in one transfer
- Big endian byte order

```
s7plcConfigure ("dev-x","192.168.0.10", 2000,
    22, 2, 1, 500, 100)
```

Record setup

- Ask programmer of PLC for
  - Layout of PVs (offset, data type)
  - Meta data (limits, bit shifts, units, etc.)
- DTYP is “S7plc”.
- INP / OUT link

  `"@PlcName/offset T=type L=low H=high B=bit"`
  - Not all parameters required in all cases.
  - L and H used in analog records for conversion.
  - B used in binary records for bit number.

Example

- 16 bit status word at offset 0

```
record (mbbiDirect, "$(DEV):status") {
   field (DTYP, "S7plc")
   field (INP,  "@dev-x/0 T=WORD")
   field (NOBT, "16")
   field (SCAN, "I/O Intr")
}
```

Example

- 24 bit integer DAC value in 4 bytes at offset 2
  - 0x00000000 = -24.0V
  - 0x00FFFFFF = + 24.0V

```
record (ai, "$(DEV):voltage") {
   field (DTYP, "S7plc")
   field (INP,  "@dev-x/2 T=INT32 L=0 H=0x00FFFFFF")
   field (EGUL, "-24")
   field (EGUF, "24")
   field (LINR, "LINEAR")
   field (EGU,  "V")
   field (PREC, "1")
   field (SCAN, "I/O Intr")
}
```

Example

- Single precision float temperature (in °C) at offset 6
- User wants °F (= °C * 1.8 + 32)

```
record (ai, "$(DEV):temperature") {
   field (DTYP, "S7plc")
   field (INP,  "@dev-x/6 T=FLOAT")
   field (ASLO, "1.8")
   field (AOFF, "32")
   field (EGU,  "°F")
   field (SCAN, "I/O Intr")
}
```

Example

- 12 byte string message at offset 10

```
record (stringin, "$(DEV):message") {
   field (DTYP, "S7plc")
   field (INP,  "@dev-x/10 L=12")
   field (SCAN, "I/O Intr")
}
```

Example

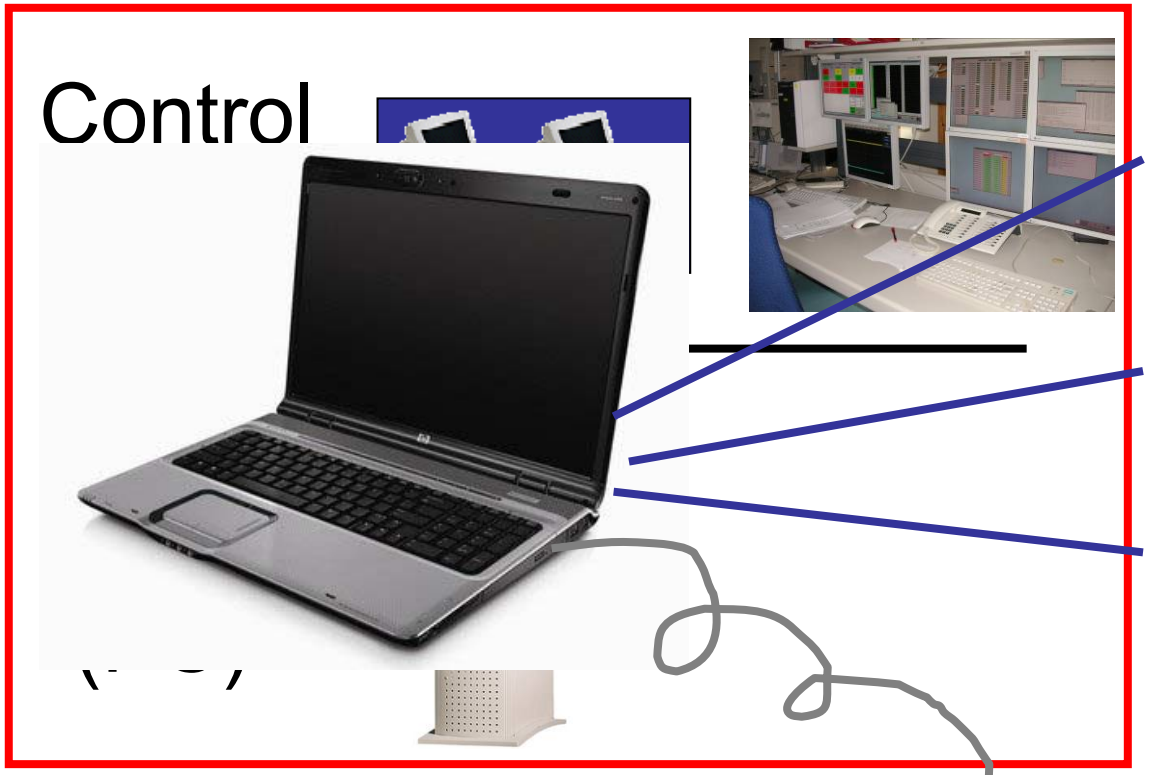
- 2 byte command output, bits 4 and 5 for switch
- 01: switch on, 10: switch off

```
record (mbbo, "$(DEV):switch") {
   field (DTYP, "S7plc")
   field (OUT,  "@dev-x/2 T=WORD")
   field (NOBT, "2")
   field (SHFT, "4")
   field (ZRVL, "2")
   field (ZRST, "OFF")
   field (ONVL, "1")
   field (ONST, "ON")
   field (PINI, "YES")
}
```

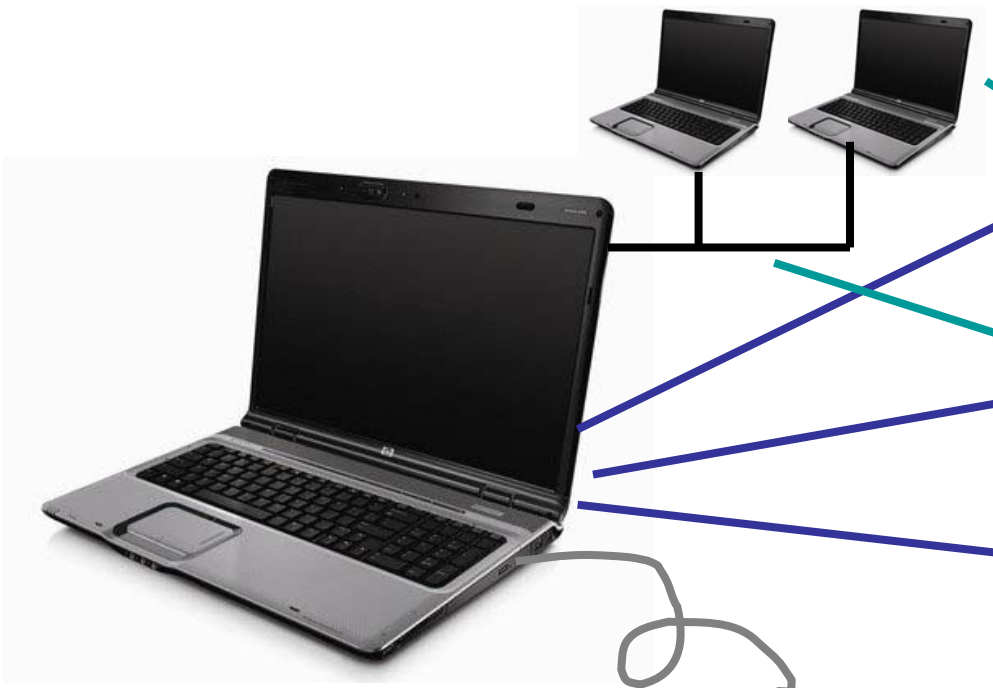
Contents

- Introduction to EPICS
- What are Records
- Set up an IOC on your PC
- Overview of S7 PLC driver
- Include S7 PLC communication into your IOC
- IOC monitoring

Architecture



Architecture continued



EPICS Clients

Channel Access

EPICS Server

TCP connection

S7 PLC

“Hardware”

Build the driver

- Create a diver module area

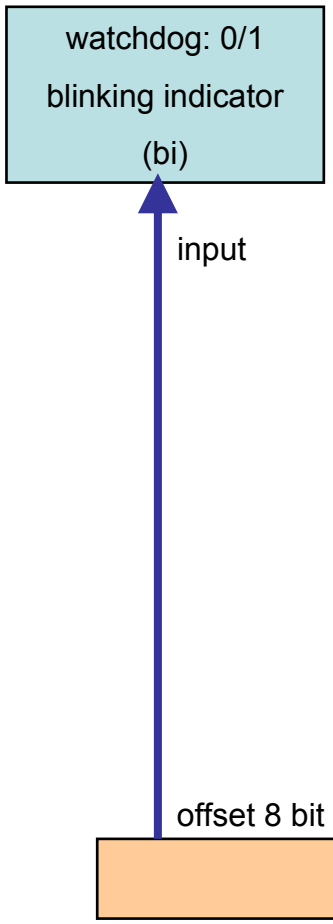
```
cd
mkdir modules
cd modules
makeBaseApp.pl –t support dummy
rm –rf dummyApp
make
```

- Build the s7 driver

```
tar xvfz ~/s7plc.tgz
cd s7plc
make
```

Example

Blinking watchdog



File PLC.template

```
record (bi, "$(PLC):Watchdog") {
    field (DESC, "Blinks when PLC runs")
    field (DTYP, "S7plc")
    field (INP,  "@$(PLCname)/8 T=BYTE B=7")
    field (ZNAM, "Tick")
    field (ONAM, "Tack")
}
```

File PowerSupply.subs

```
file PLC.template {
    { PLC = PLC1 }
}
file PowerSupplySimu.template ...
```

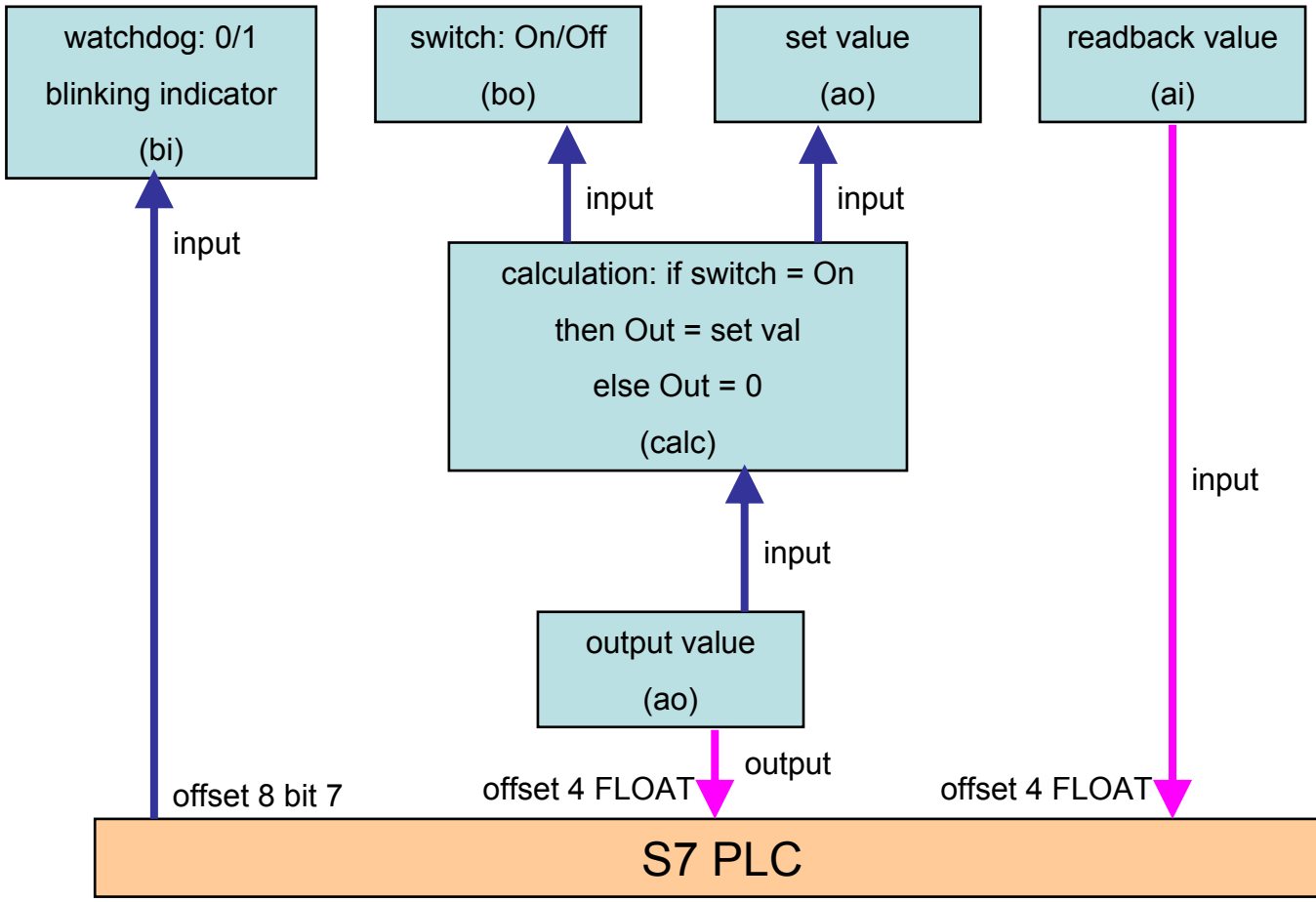
Include driver to st.cmd

```
# Load drivers and dbd files ==============
dlload /home/epics/modules/lib/linux-x86/libs7plc.so
dbLoadDatabase /home/epics/modules/dbd/s7plc.dbd
s7plc_registerRecordDeviceDriver

# Configure the hardware
#s7plcConfigure "PLCname","IPaddr",IPport,inSize,outSize,
#          bigEndian,recvTimeout,sendIntervall
s7plcConfigure "PLC1","192.168.0.10",2000,10,10,1,500,100

# Load your record files ==================
dbLoadTemplate("PowerSupply.subs")

# start the EPICS IOC =====================
iocInit
```

Example

1. Blinking watchdog

2. Modify Power Supply



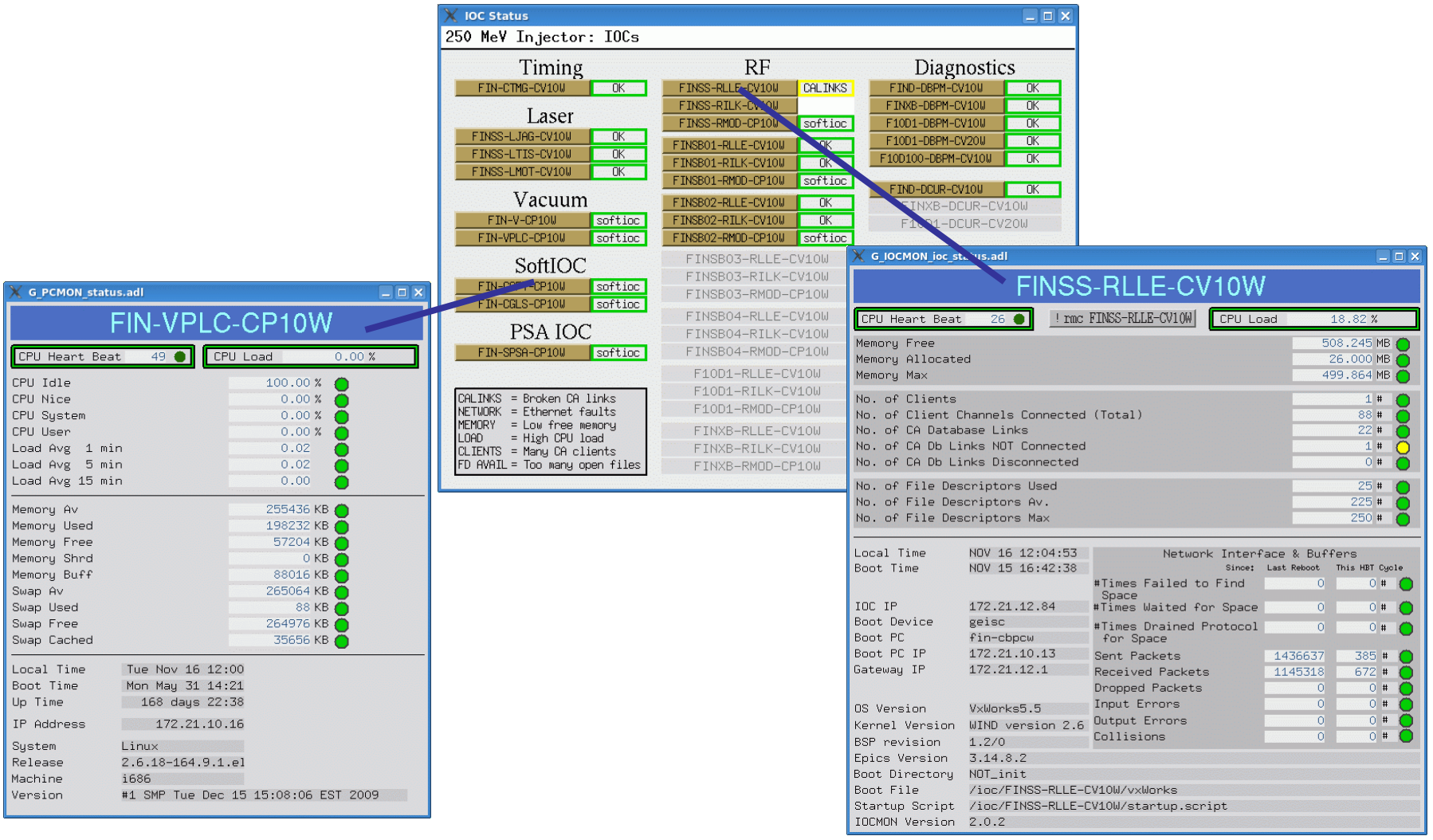
Modify template file

```
record (ai, "$(Name):Readback")
{
    field (DESC, "Actual readback from HW")
    field (EGU,  "V")
    field (PREC, "2")
    field (DTYP, "S7plc")
    field (INP,  "@$(PLCname)/4 T=FLOAT") # read from PLC
    field (SCAN, "I/O Intr")
}
record (ao, "$(Name):OutputValue")
{
    field (DESC, "Value send to HW")
    field (EGU,  "V")
    field (PREC, "2")
    field (DOL,  "$(Name):Calculation")  # get the output value
    field (OMSL, "closed_loop")
    field (DTYP, "S7plc")
    field (OUT,  "@$(PLCname)/4 T=FLOAT") # write to PLC
}
```

Contents

- Introduction to EPICS
- What are Records
- Set up an IOC on your PC
- Overview of S7 PLC driver
- Include S7 PLC communication into your IOC
- IOC monitoring

How does it look like at PSI



Alternatives

- vxStats (APS)
  - http://www.aps.anl.gov/epics/modules/soft/vxStats
  - Very old. Only vxWorks, no PC support
- iocmon (PSI)
  - http://epics.web.psi.ch/software/iocmon
  - vxStats plus extensions, only vxWorks, no PC support
- pcmon (PSI)
  - http://epics.web.psi.ch/software/pcmon
  - very rudimentary, only Linux
- devIocStats (SLAC)
  - http://www.slac.stanford.edu/comp/unix/package/epics/site/devIocStats
  - Many different operating systems

What's missing?

- How to install EPICS to a central location
  - Make it available for all users
- How to compile drivers into the ioc program
  - Avoids loading of driver in startup script
- How to make a driver loadable
  - Allows loading of driver in startup script
    (Most drivers don't allow this out-of-the-box)
- How to build EDM
  - Alternative to MEDM
  - Nicer but more complicated to install

10 really neat things about EPICS

1. It is free
2. It is Open Source
3. There are lots of users
4. All a client needs to know to access data is a PV name
5. You can pick the best tools out there …
6. … or build your own
7. The boring stuff is already done
8. There is a lot of expertise available close by
9. A good contribution becomes internationally known
10. It does not matter if you need 10 PVs or 10 million PVs

EPICS Web Page

The central site for EPICS information

- Documentation
- CA Clients
- Device support (driver)
- Tech-talk (mailing list and archive)

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