EPICS Database

Kay Kasemir,

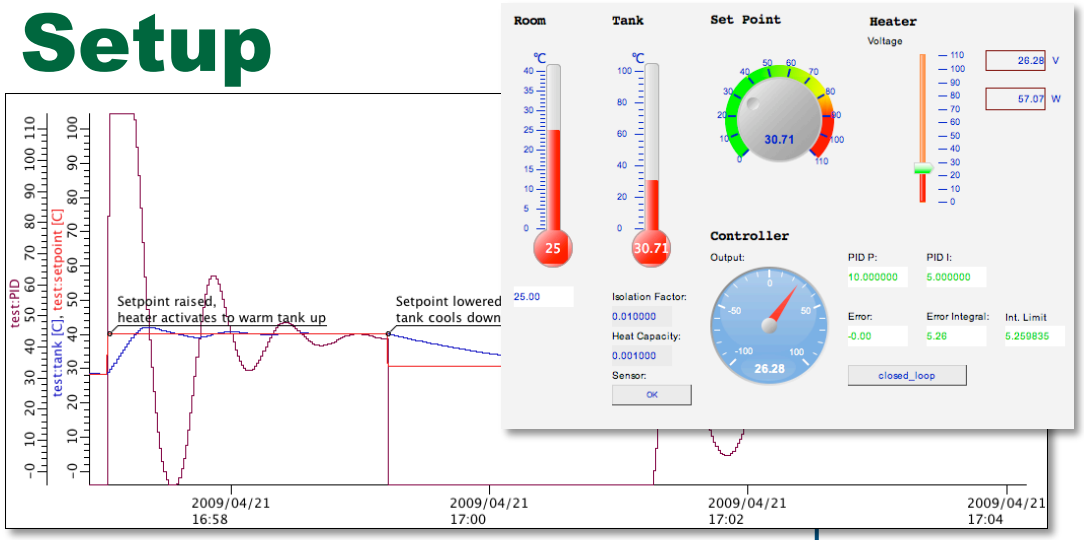
SNS/ORNL

Many slides from Andrew Johnson, APS/ANL

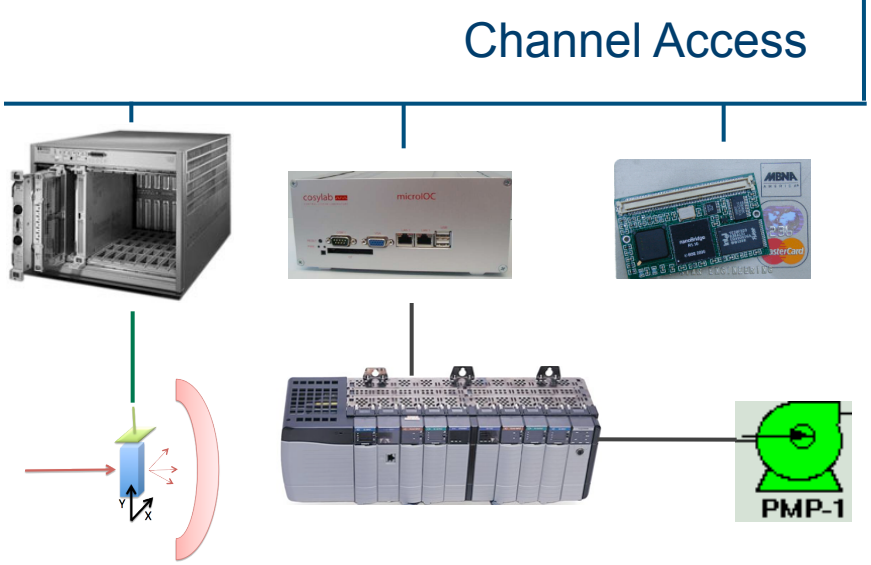
Sept. 2014

Distributed EPICS Setup

- **Operator Interface**



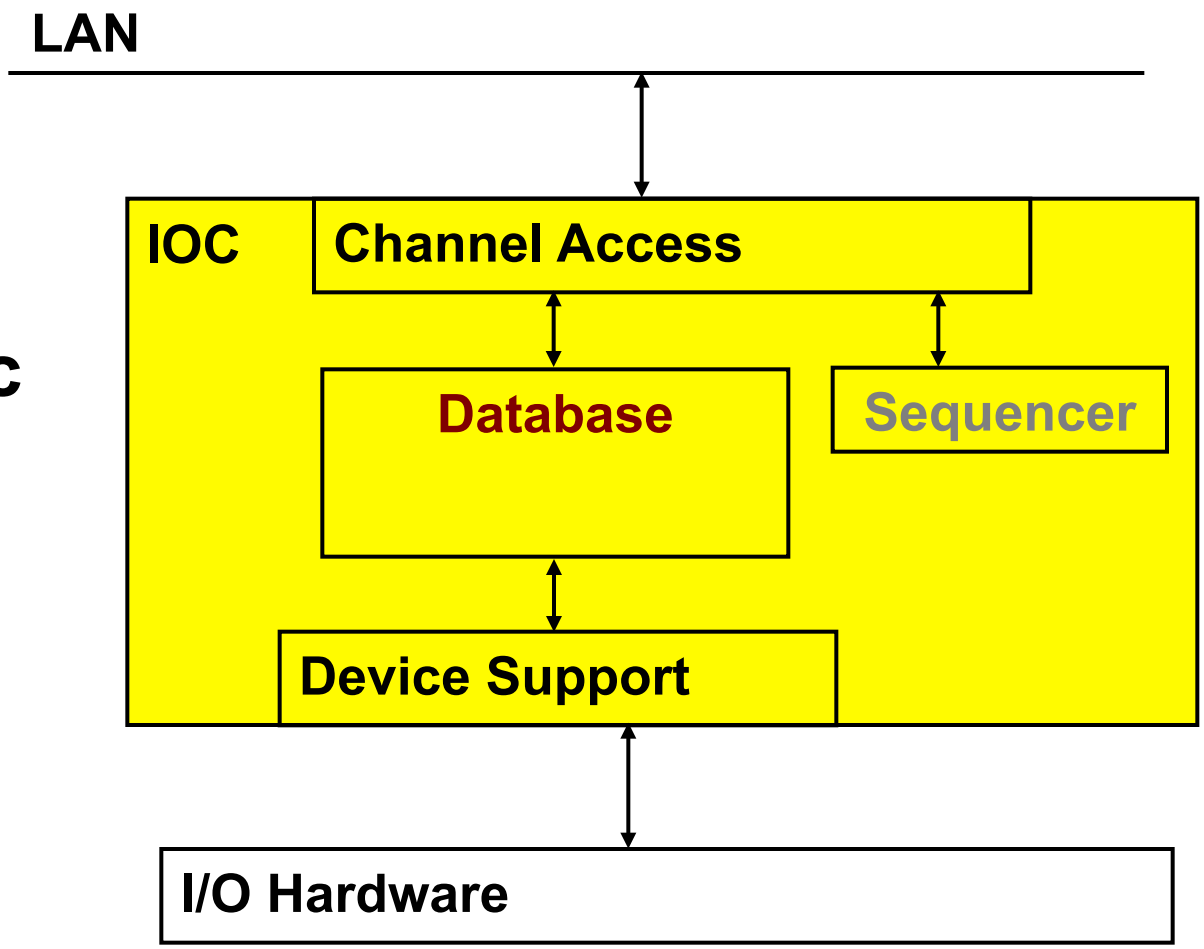
- **Input/Output Controller (IOC)**



OAK RIDGE National Laboratory

IOC

- **Database: Data Flow, mostly periodic processing**
- **Sequencer: State machine, mostly on-demand**



"Hard" IOCs run vxWorks and directly interface to A/D, D/A, LLRF, … hardware.
"Soft" IOCs run on Linux etc. and have no I/O Hardware other than serial or networked devices (Moxa to motor controller, …)

IOC Database

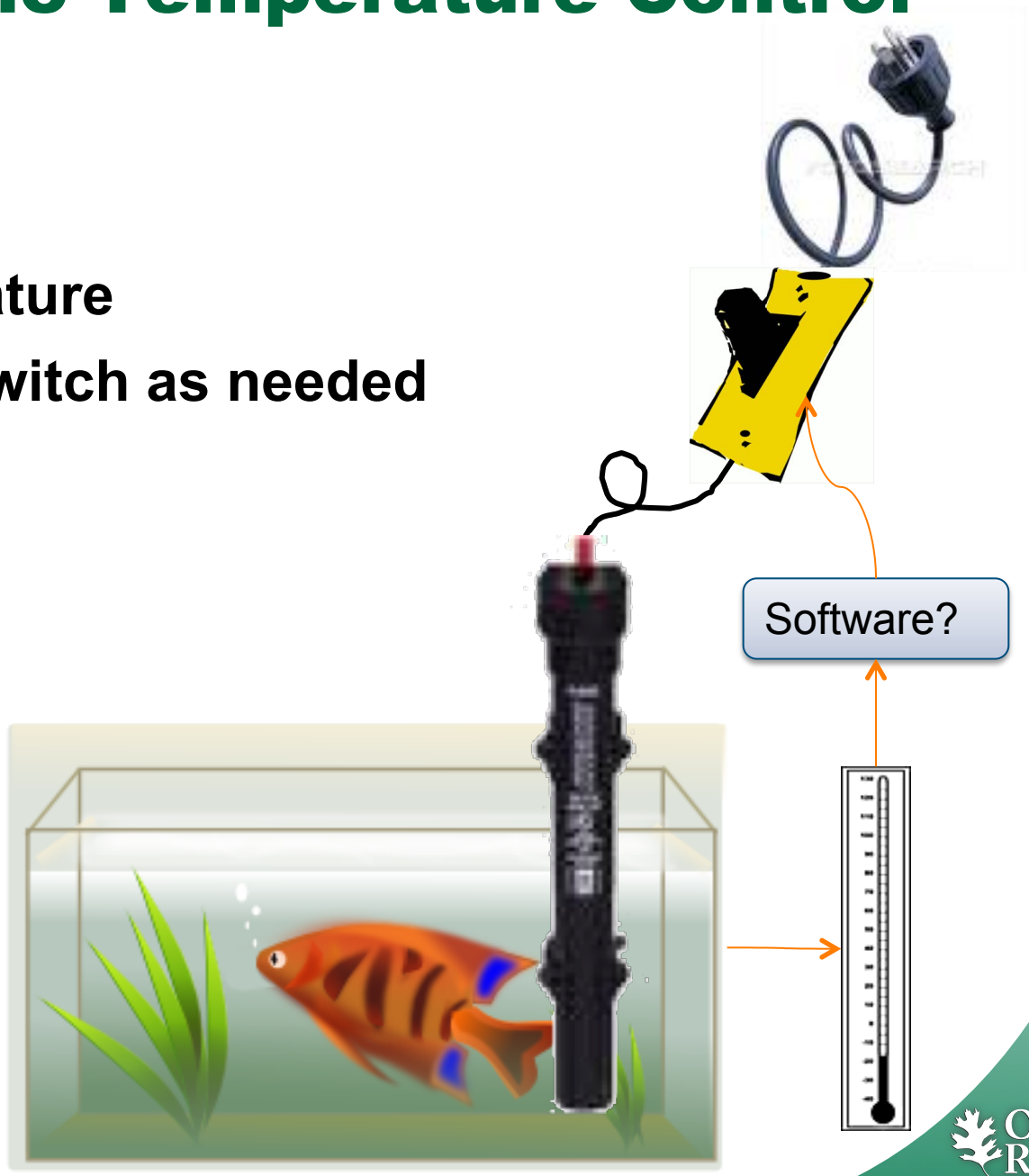
- 'iocCore' software loads and executes 'Records'
  - Configuration of records instead of custom Coding
- All control system toolboxes have (better?)
  - GUI tools
  - Network protocols
  - Hardware drivers

  but few have a comparable database!

Example: Basic Temperature Control

Task:

1. **Read temperature**
2. **Open/close switch as needed**
3. **Repeat**



One Fishy Database

```
record(ai, temp) {
  field(DESC, "Read Temperature")
  field(SCAN, "1 second")
  field(DTYP, "XYZ ADC")
  field(INP, "#C1 S4")
  field(PREC, "1")
  field(LINR, "typeJdegC")
  field(EGU, "Celsius")
  field(HOPR, "100")
  field(LOPR, "0")
  field(SMOO, "0.5")
  field(HIGH, "15")
  field(HSV, "MAJOR")
}


record(calcout, check) {
  field(DESC, "Control Heater")
  field(CALC, "A<10")
  field(INPA, "temp CP MS")
  field(OUT, "switch")
  field(OOPT, "On Change")
}


record(bo, switch) {
  field(DESC, "Heater switch")
  field(DTYP, "XYZ DAC")
  field(OUT, "#C1 S3")
  field(ZNAM, "Open")
  field(ONAM, "Closed")
  field(IVOA, "Set output to IVOV")
  field(IVOV, "0")
}
```

Analog Input Record

SCAN Field

Binary Output Record

Database = Records + Fields + Links

- IOC loads and executes one or more databases
- Each database has records
- Each record has
  - Name (unique on the whole network)
  - Fields (properties, can be read, most also written at runtime)
  - Often device support to interface to hardware
  - Links to other records

Records are Active

- Records ‘do’ things
  - Get data from other records or hardware
  - Perform calculations
  - Check value ranges, raise alarms
  - Write to other records or hardware

  What they do depends on record type and field values
- … when they are processed
  - Records process periodically or when triggered by events or other records

  No action occurs unless a record is processed

Record Types

- ai/ao: Analog input/output
  - Read/write number, map to engineering units
- bi/bo: Binary in/out
  - Read/write bit, map to string
- calc: Formula
- mbbi/mbbo: Multi-bit-binary in/out
  - Read/write 16-bit number, map bit patterns to strings
- stringin/out, longin/out, seq, compress, histogram, waveform, sub, …

Common Fields

- Design Time
  - NAME: Record name, unique on network!
  - DESC: Description
  - SCAN: Scan mechanism
  - PHAS: Scan phase
  - PINI: Process once on initialization?
  - FLNK: Forward link
- Runtime
  - TIME: Time stamp
  - SEVR, STAT: Alarm Severity, Status
  - PACT: Process active
  - TPRO: Trace processing
  - UDF: Undefined? Never processed?
  - PROC: Force processing

Record Scanning

- **SCAN field:**
  - **When processed by other records:**
    "Passive" (default)
  - **Periodically:**
    ".1 second", ".2 second", ".5 second",
    "1 second", "2 second", "5 second", "10 second"
  - **On event:**
    "Event" (EVNT field selects the event),
    "I/O Intr" (if device support allows this)
- **PHAS field**
  - **Adds order to records that are on the same periodic scan**
    - **First PHAS=0, then PHAS=1, …**
- **PINI**
  - **Set to "YES" to force record once on startup.**
    Good idea for "operator input" records that are hardly ever changed, so they have an initial value.
- **PROC**
  - **Writing to this field will process a record**

Example "first.db"

```
# The simplest record that 'does' something
# and produces changing numbers
record(calc, "$(S):random")
{
   field(SCAN, "1 second")
   field(INPA, "10")
   field(CALC, "RNDM*A")
}
```

- **Execute:** `softIoc –m S=$USER –d first.db`
- **In another terminal:** `camonitor $USER:random`
- **Try** `dbl, dbpr, dbpf`

Common Input/Output Record Fields

- DTYP: Device type
- INP/OUT: Where to read/write
- RVAL: Raw (16 bit) value
- VAL: Engineering unit value

Output Only:

- DOL: Desired Output Link.
  Output records read this link to get VAL, then write to OUT…
- OMSL: .. if Output Mode Select = closed_loop
- IVOA: Invalid Output Action
- DRVL, DRVH: Drive limits

More from "first.db"

```
# A ramp from 0 to 'limit', were limit
# can be configured via a separate record
record(ao, "$(S):limit")
{
   field(DRVH, "100")
   field(DOL,  "10")
   field(PINI, "YES")
}

record(calc, "$(S):ramp")
{
   field(SCAN, "1 second")
   field(INPA, "$(S):ramp")
   field(INPB, "$(S):limit")
   field(CALC, "A<B ? A+1 : 0")
}
```

Using 'output'.
'input' would also work, since there's no hardware to read or write, but only 'output' has DRVH…

Reading inputs:
A = my own current value
B = value of ..limit record

Which record is scanned?

Analog Record Fields

- EGU: Engineering units name
- LINR: Linearization (No, Slope, breakpoint table)
- EGUL, EGUF, ESLO, EOFF: Parameters for LINR
- LOLO, LOW, HIGH, HIHI: Alarm Limits
- LLSV, LSV, HSV, HHSV: Alarm severities

Binary Record Fields

- ZNAM, ONAM: State name for “zero”, “one”
- ZSV, OSV: Alarm severities

Record Links

- Input links may be
  - Constant number: “0”, “3.14”. “-1.6e-19”
- Input or Output links may be
  - Name of other record’s field: “other”, “other.VAL”, “other.A”
    - If other record is in same IOC: “Database link”
    - If name not found: “Channel Access link”
  - Hardware link
    - Details depend on device support
    - DTYP field selects device support
    - Format examples: “@plc12 some_tag”, “#C1 S4”
- A record’s FLNK field processes anther record after current record is ‘done’

Record (Database, Channel Access) Links

- Format: “record.field {flags}”
  - VAL is default for field
- Flags:
  - PP: Process a passive target record
    - INP, DOL: Before reading
    - OUT: After writing
  - NPP: non-process-passive (default)
  - MS: Maximize severity
  - NMS: non-MS (default)
  - MSS: Maximize Severity and Status
  - MSI: .. when severity = INVALID
- Example:

```
field(“INP”, “other_rec.VAL PP MS”)
```

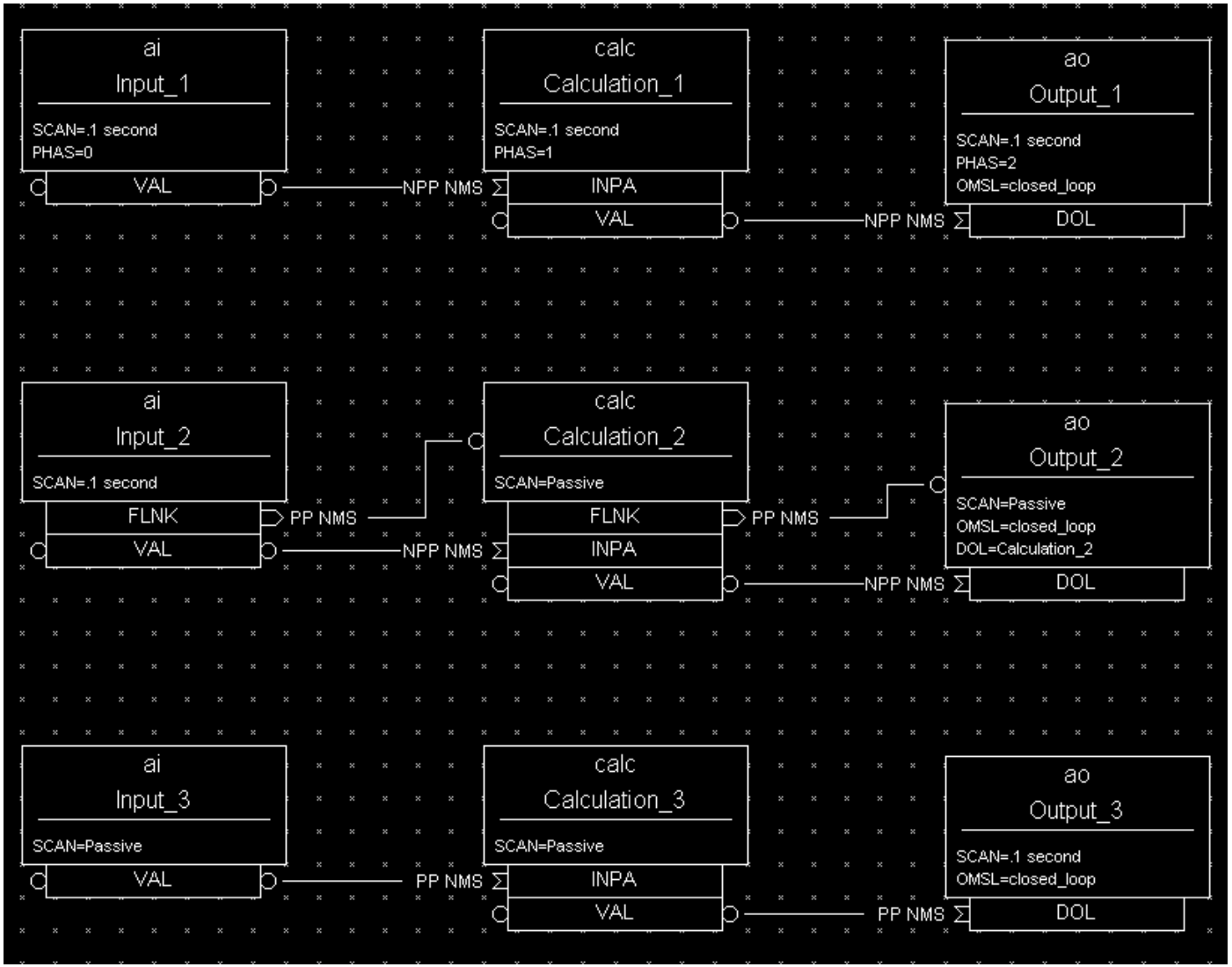
Additional Channel Access Link Flags

- **CA: Force CA link, even though target in same IOC**
- **CP: For INP link, process on received CA monitor**
- **CPP: CP, but only if SCAN=Passive**

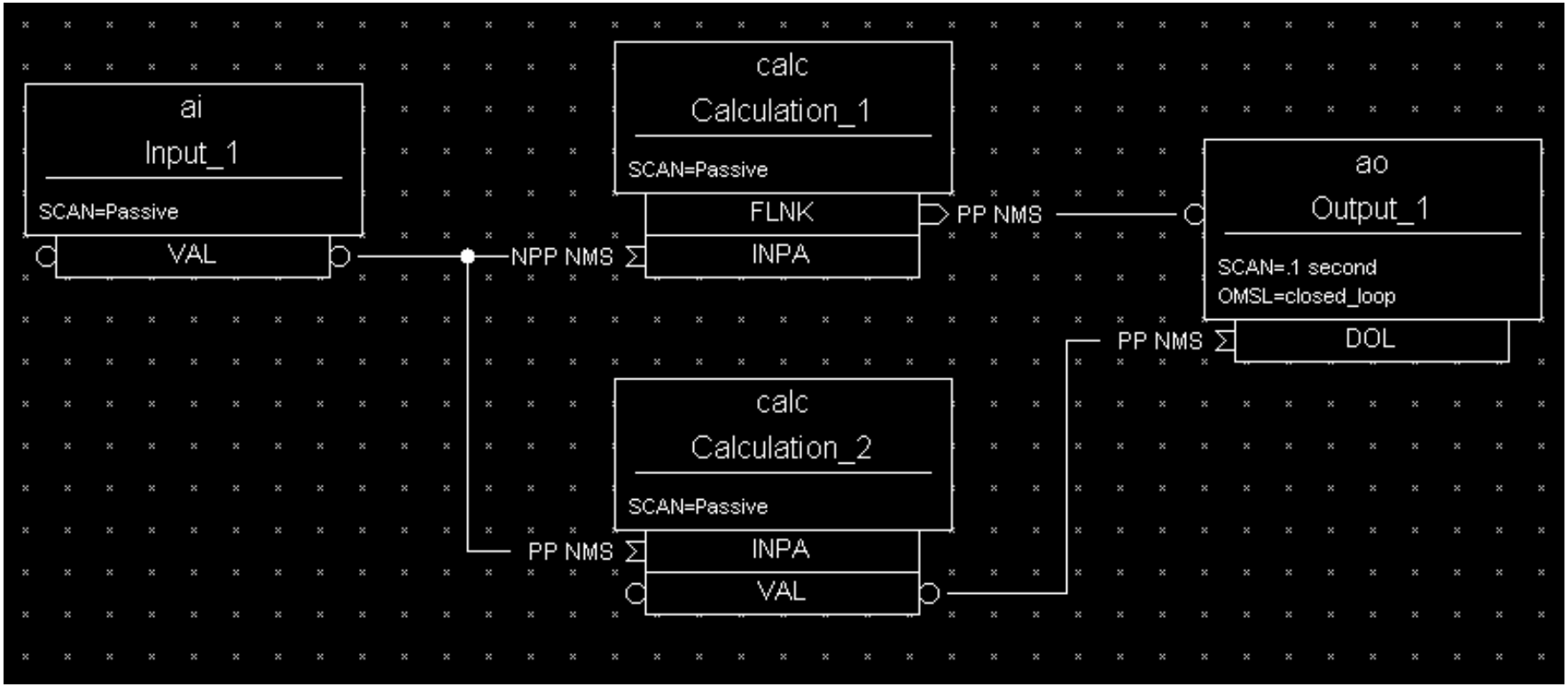
.. This won't make any sense until we learn more about Channel Access…

The IOC Application Developer's Guide, Chapter 5 on Data Base .. Processing is worth reading.

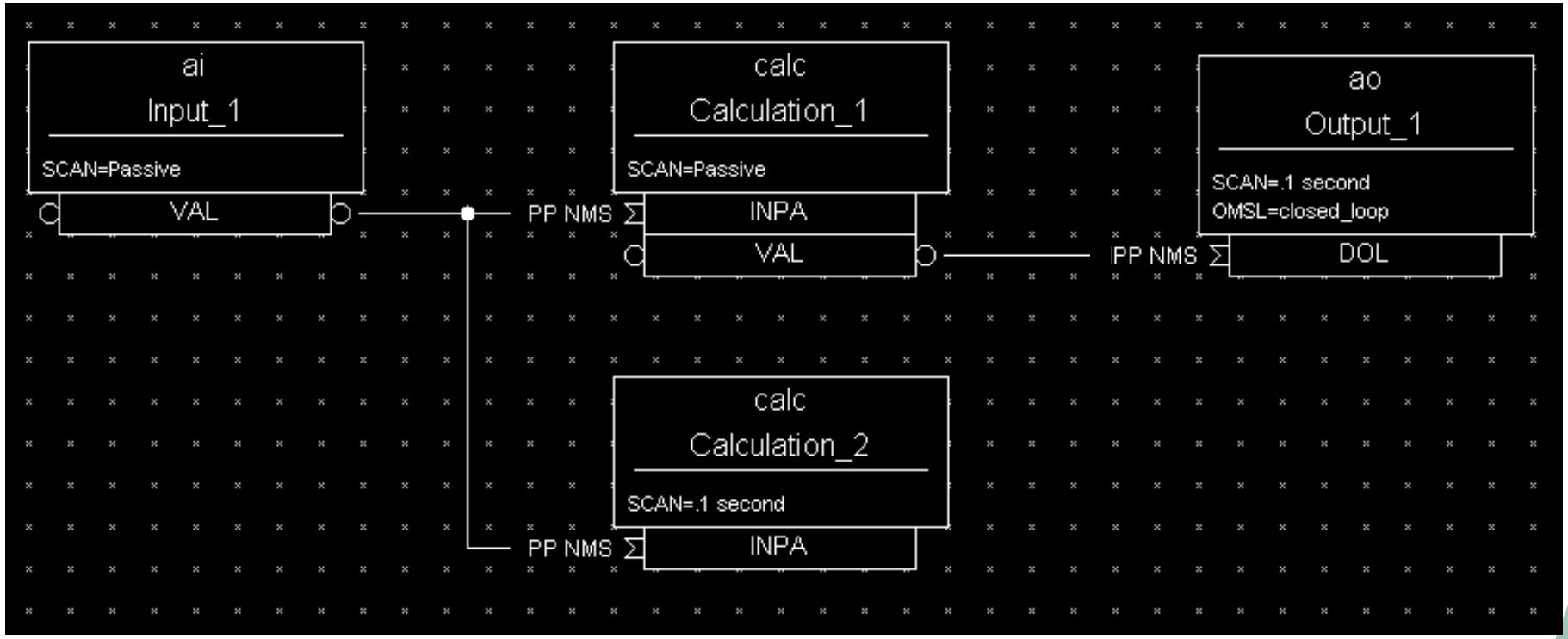
Processing chains



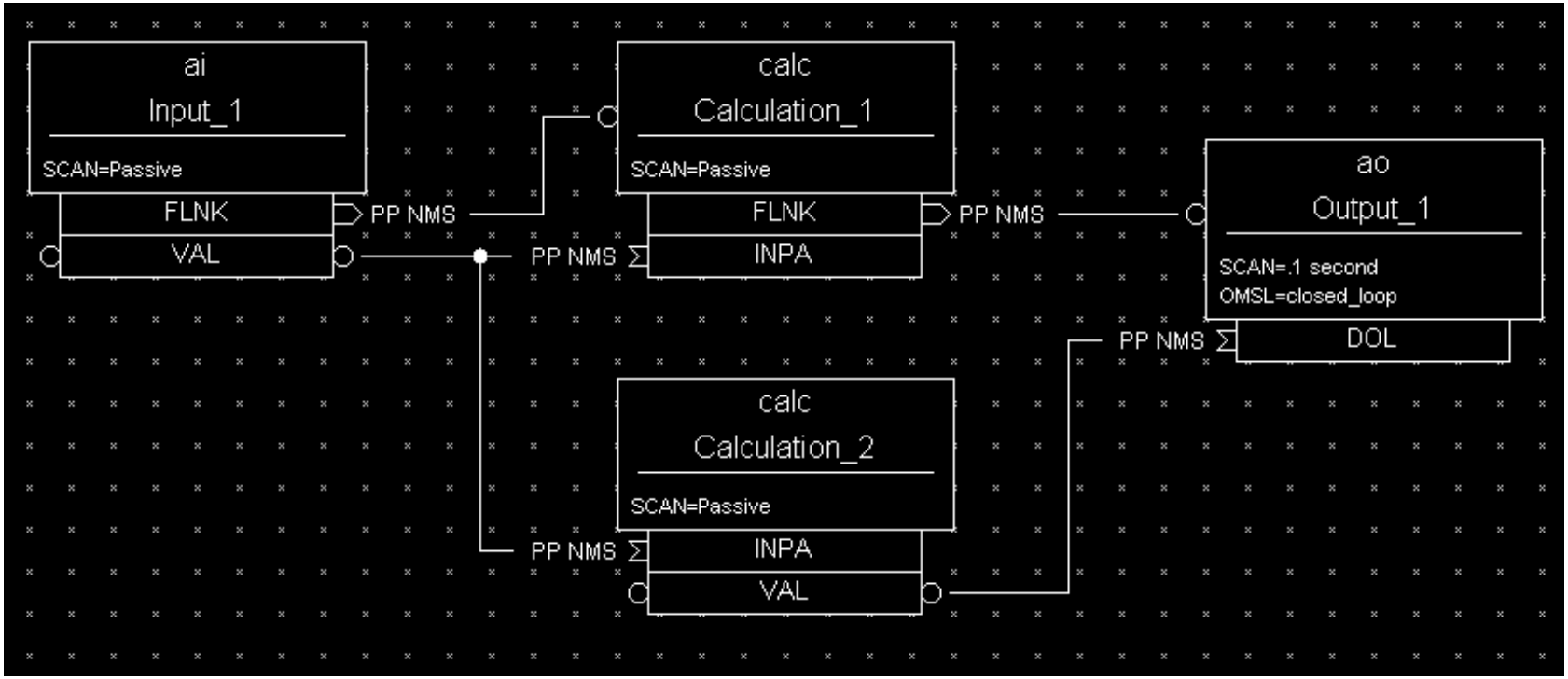
Which record is never processed?



How often is Input_1 processed?



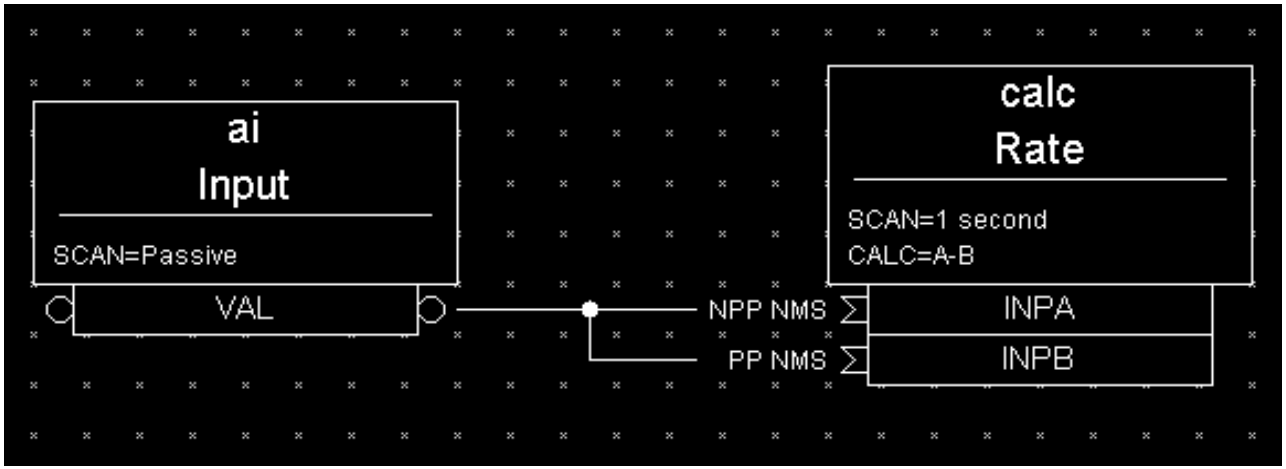
How long will this take?



- **PACT: Processing Active**

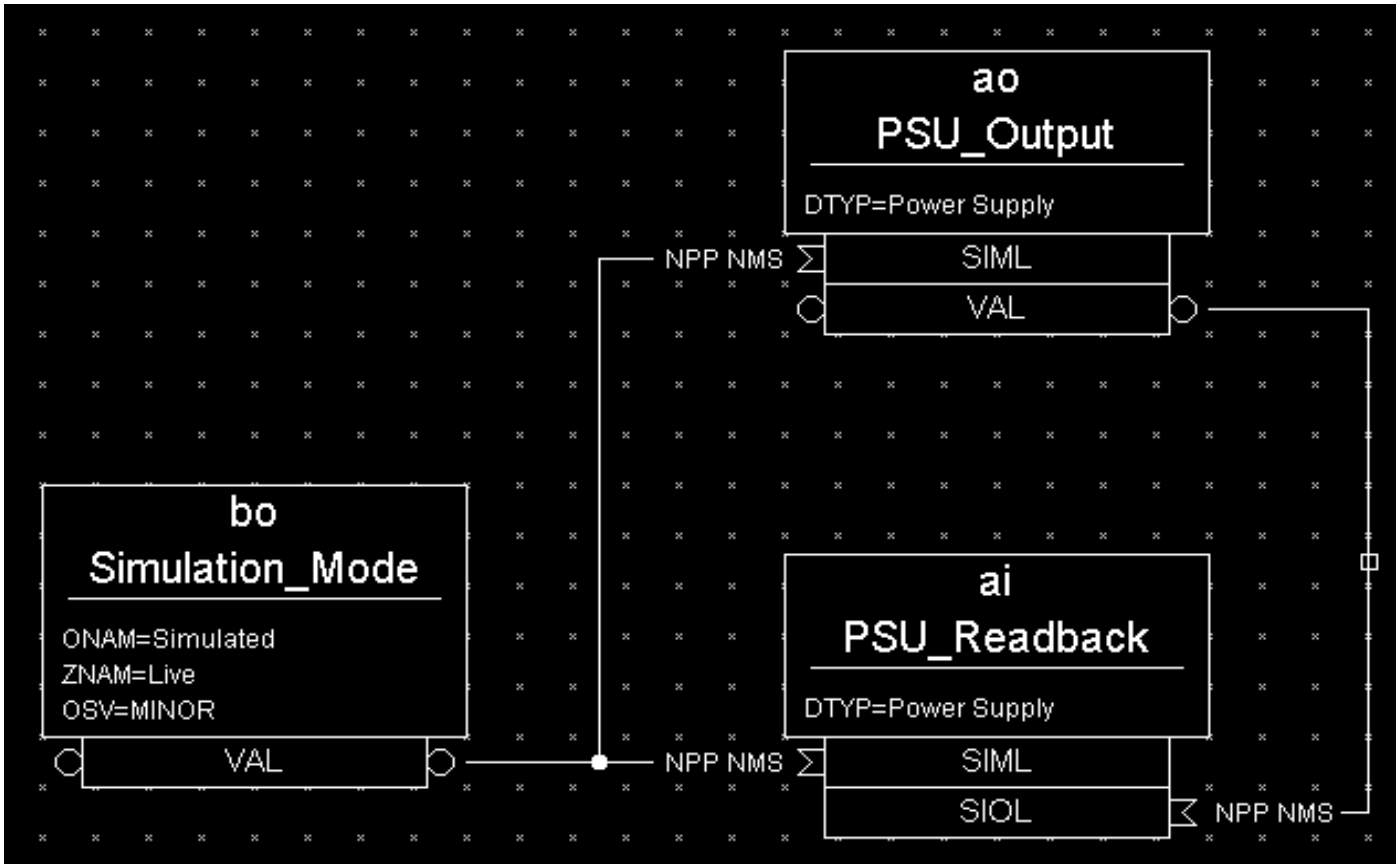
Rate Of Change Example

Calculating “Rate-of-Change” of an Input



INPA fetches data that is 1 second old because it does not request processing of the AI record. INPB fetches current data because it requests the AI record to process. The subtraction of these two values reflects the ‘rate of change’ (difference/sec) of the pressure reading.

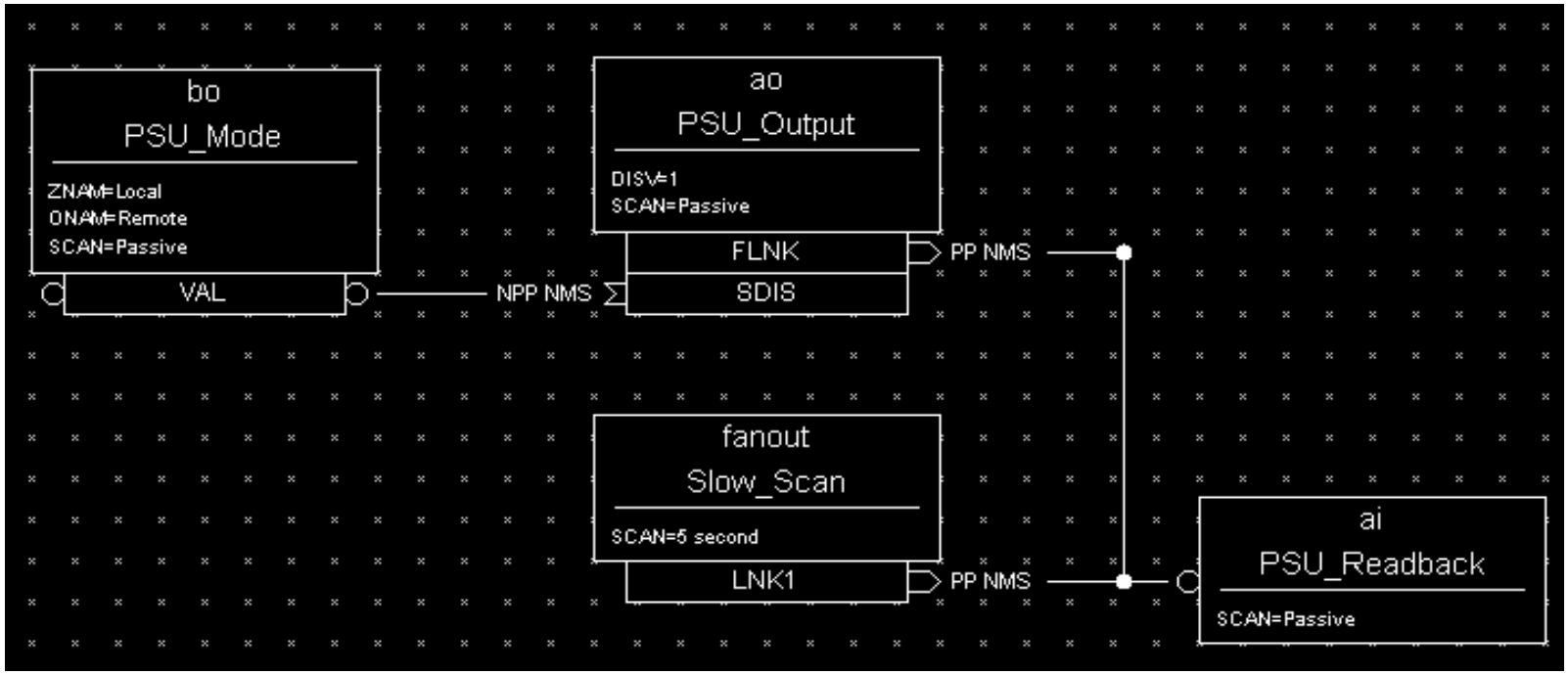
Simulation Mode



When in simulation mode, the AO record does not call device support and the AI record fetches its input from the AO record.

Multiple Scan Triggers

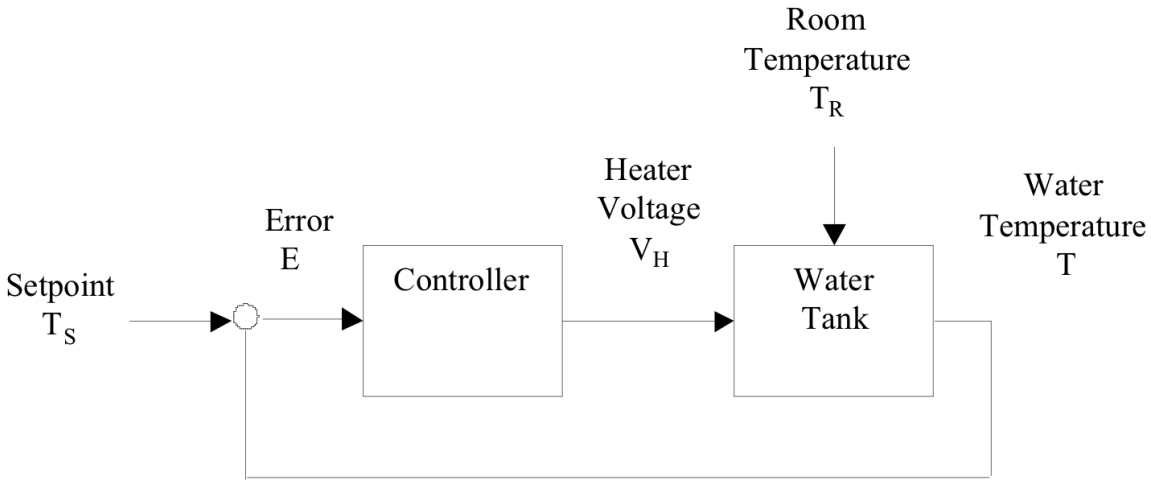
Slow Periodic Scan with Fast Change Response



The AI record gets processed every 5 seconds AND whenever the AO record is changed. This provides immediate response to an operator's changes even though the normal scan rate is very slow. Changes to the power supply settings are inhibited by the BO record, which represents a Local/Remote switch.

Heater Control Simulation

- **Typical control l**



- **PID Controller**

$$O(n) = K_P \; E(n) + K_I \sum_i E(i)\, dT + K_D \, [E(n)-E(n-1)]/dT$$

  - Error readings $E(n)$
  - Output $O(n)$
  - Proportional, Integral, Derivative Gains K_x

User Inputs to Simulation

- Macros
- Analog *output* for user *input* because of DRVL/DRVL

```
record(ao, "$(user):room")
{
    field(DESC, "Room Temperature")
    field(EGU, "C")
    field(HOPR, "40")
    field(LOPR, "0")
    field(DRVL, "0")
    field(DRVH, "40")
    field(DOL, "25")
    field(PINI, "YES")
}
```

```
record(ao, "$(user):setpoint")
{
    field(DESC, "Temperature Setpoint")
    field(EGU, "C")
    field(HOPR, "0")
    field(LOPR, "100")
    field(DRVL, "0")
    field(DRVH, "100")
    field(PREC, "1")
    field(DOL,  "30")
    field(PINI, "YES")
}
```

Simulated Tank Temperature

```
# supervisory: user can adjust voltage
# closed_loop: PID (in separate control.db) sets voltage
# When PID is INVALID, go back to 0 voltage
record(ao, "$(user):heat_V")
{
    field(DESC, "Heater Voltage")
    field(EGU, "V")
    field(DRVL,"0")
    field(DRVH,"110")
    field(DOL, "$(user):PID MS")
    field(OMSL,"closed_loop")
    field(IVOA, "Set output to IVOV")
    field(IVOV, "0")
}

# ~1100 Watt heater when run with 110V:
# P = U I = U^2 / R,  R~12 Ohm
record(calc, "$(user):heat_Pwr")
{
    field(DESC, "Heater Power")
    field(EGU, "W")
    field(INPA, "$(user):heat_V PP NMS")
    field(CALC, "A*A/12.1")
}
```

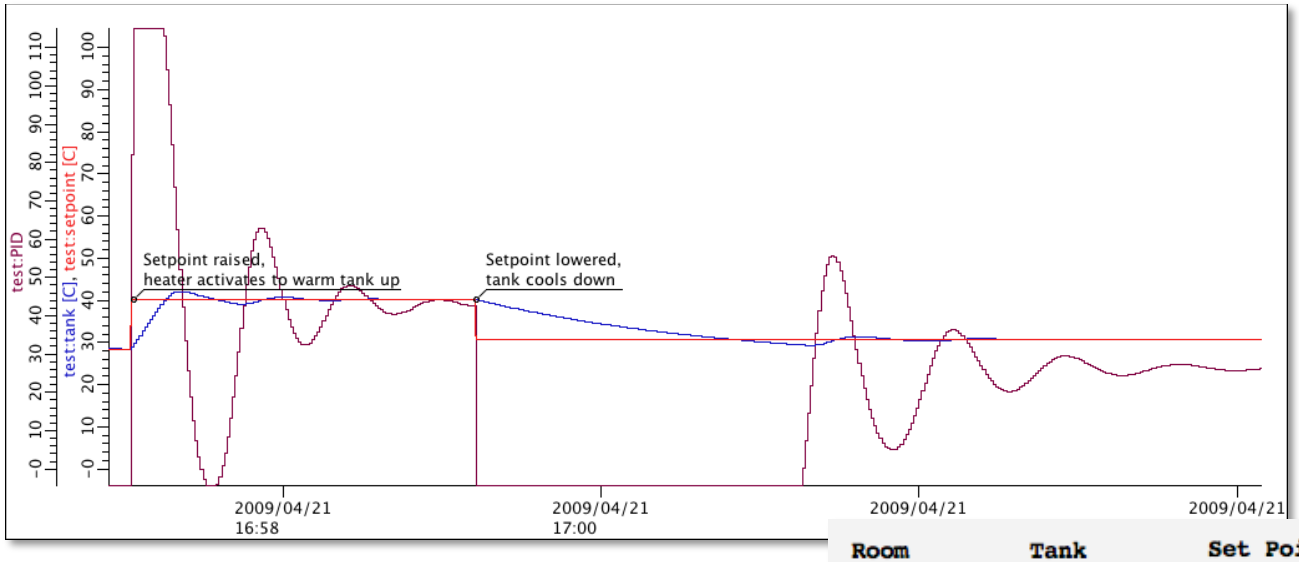
```
# Every second, calculate new temperature
# based on current temperature,
# room temperature and heater
#
# A - current temperature
# B - room temperature
# C - heater power
# D - isolation factor (water <-> room)
# E - heat capacity (would really depend on water volume)
#
# Very roughly with
# T(n+1) = T(n) + [Troom-T(n)]*Isolation_factor
#          + heater_pwr * heat_capacity
record(calc, "$(user):tank_clc")
{
    field(DESC,"Water Tank Simulation")
    field(SCAN,"1 second")
    field(INPA,"$(user):tank_clc.VAL")
    field(INPB,"$(user):room")
    field(INPC,"$(user):heat_Pwr PP NMS")
    field(INPD,"0.01")
    field(INPE,"0.001")
    field(CALC,"A+(B-A)*D+C*E")
    field(FLNK,"$(user):tank")
}
```

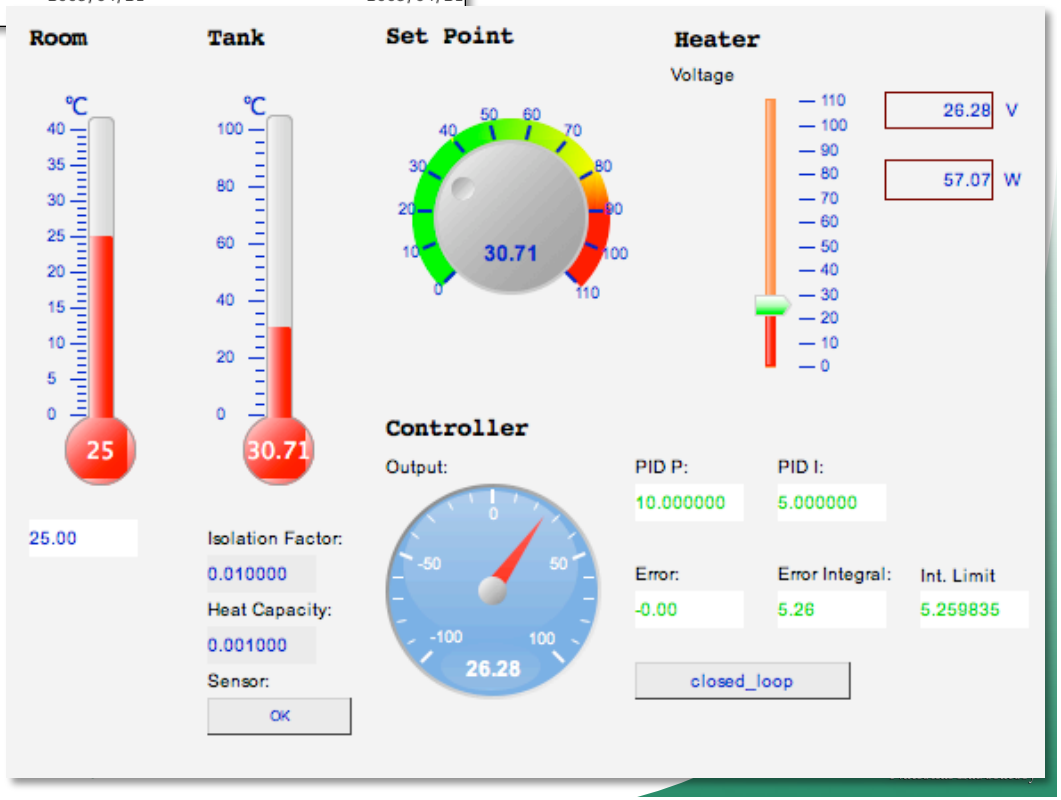
PID (without D) by Hand

```
# Error computation's SCAN drives the rest
record(calc, "$(user):error")
{
    field(DESC, "Temperature Error")
    field(SCAN, "1 second")
    field(INPA, "$(user):setpoint")
    field(INPB, "$(user):tank MS")
    field(CALC, "A-B")
    field(PREC, "1")
    field(FLNK, "$(user):integral")
}
# Integrate error (A) but assert that
# it stays within limits (C)
record(calc, "$(user):integral")
{
    field(DESC, "Integrate Error for PID")
    field(PREC, "3")
    field(INPA, "$(user):error PP MS")
    field(INPB, "$(user):integral")
    field(INPC, "20.0")
    field(CALC, "(B+A>C)?C:(B+A<-C)?(-C):(B+A)")
    field(FLNK, "$(user):PID")
}
```

```
# PID (PI) computation of new output
# A - Kp
# B - error
# C - Ki
# D - error integral
record(calc, "$(user):PID")
{
    field(DESC, "Water Tank PID")
    field(PREC, "3")
    field(LOPR, "0")
    field(HOPR, "110")
    field(INPA, "10.0")
    field(INPB, "$(user):error MS")
    field(INPC, "5.0")
    field(INPD, "$(user):integral MS")
    field(CALC, "A*B+C*D")
}
```

Heater Simulation





Alarms

- Common fields:
  - SEVR: Alarm Severity
    - NONE, MINOR, MAJOR, INVALID
  - STAT: Alarm Status
    - UDF, READ, WRITE, CALC, HIGH, …
- Binary records:
  - ZSV, OSV: Severity for ‘zero’ and ‘one’ state
- Analog records:
  - LOLO, LOW, HIGH, HIHI: Thresholds
  - LLSV, LSV, HSV, HHSV: Associated severities
  - HYST: Hysteresis

Alarm Example

```
# Raise MAJOR alarm when temperature near boiling point
record(ai, "$(user):tank")
{
    field(DESC,"Water Temparature")
    field(SCAN,"…
    field(INP, "…
    field(EGU, "C")
    field(HIGH,"90")
    field(HSV, "MAJOR")
}
```

Summary

- Database Records configure the IOC's data flow
  - Fields, links instead of custom code
- There's more
  - Fields MDEL/ADEL, bo.HIGH
  - Access security
- See http://aps.anl.gov/epics for
  - IOC Application Developers' Guide
  - Record Reference Manual

Things to try

- **Build a simple on/off fish tank controller**
  - **Simulate the heater**
    - **'bo' record to turn on/off**
  - **Simulate the water temperature**
    - **'calc' record(s):**
      - **Temperature rises when heater is on**
      - **Temperature drops to room temperature when heater is off**
  - **Add controller**
    - **'ao' record to allow entering the desired temperature**
    - **'calc' record(s) to turn heater on/off, automagically keeping water temp. close to setpoint**

OK to take inspiration from Heater Control Simulation example

- **But don't copy a single line without understanding it**
- **Compare behavior of the P-I controller with on/off controller**