

EPICS Database

... in 1 hour?!



**Kay Kasemir,
SNS/ORNL**

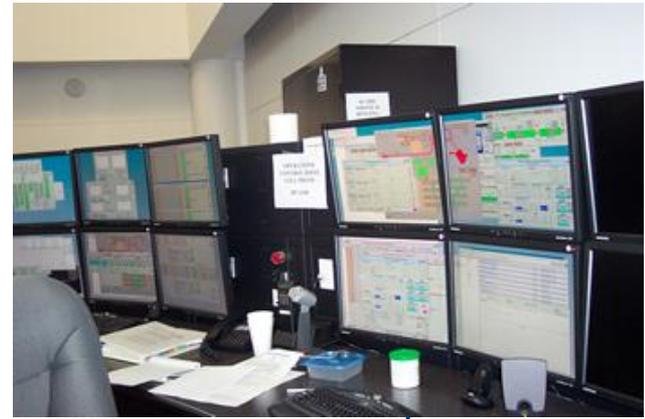
**Many slides from Andrew Johnson,
APS/ANL,
Jan. 2007 USPAS EPICS Course**

kasemirk@ornl.gov

May 2009

Distributed EPICS Setup

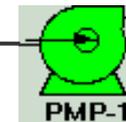
- Operator Interface, Archiver, ...



- EPICS Base:
 - Input/Output Controller (IOC)
 - As “Soft IOC”
 - As “Hard IOCs” at front-end level



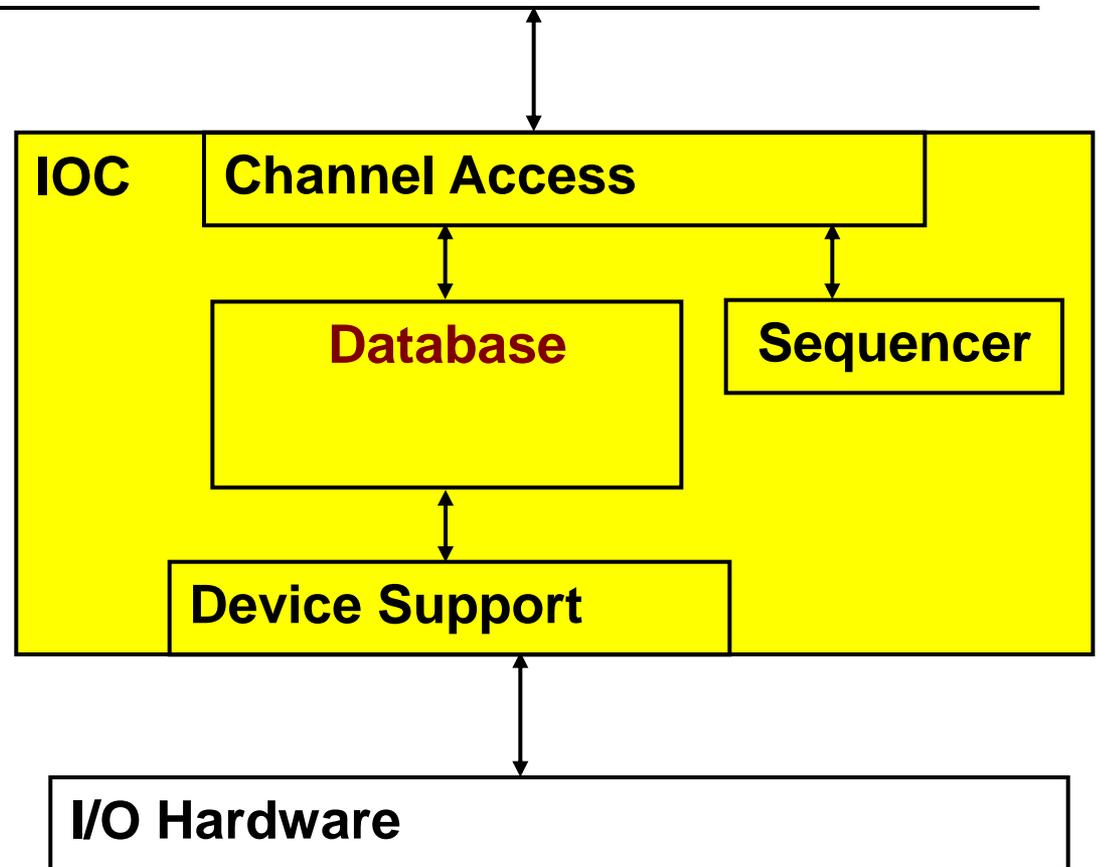
Channel Access



IOC

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

LAN



**“Soft IOCs” have no I/O Hardware.
May use Device Support to contact networked devices
(PLC via Ethernet, ...)**

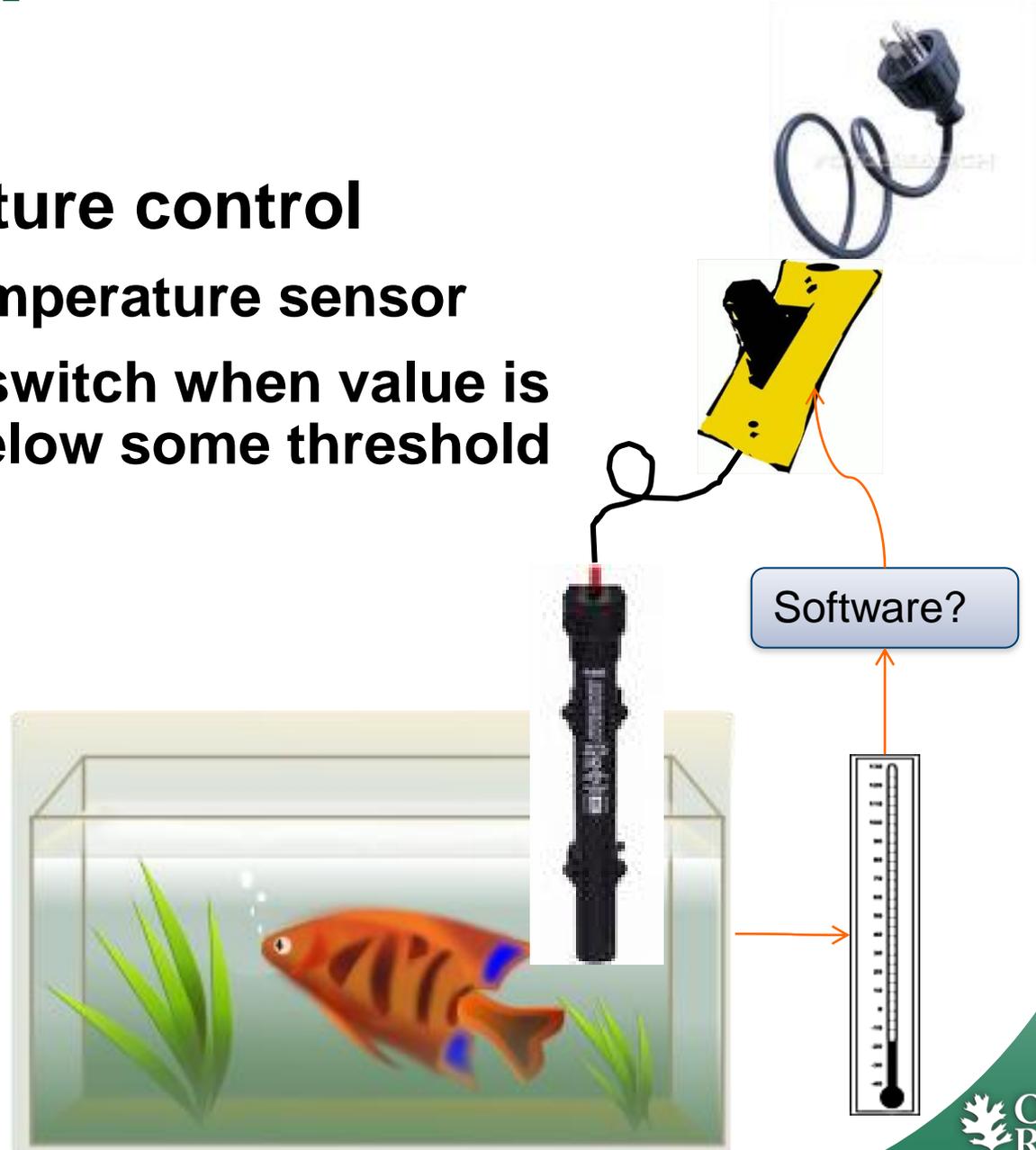
IOC Database

- **'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 Task

- **Basic temperature control**
 - Read some temperature sensor
 - Open/close a switch when value is above resp. below some threshold



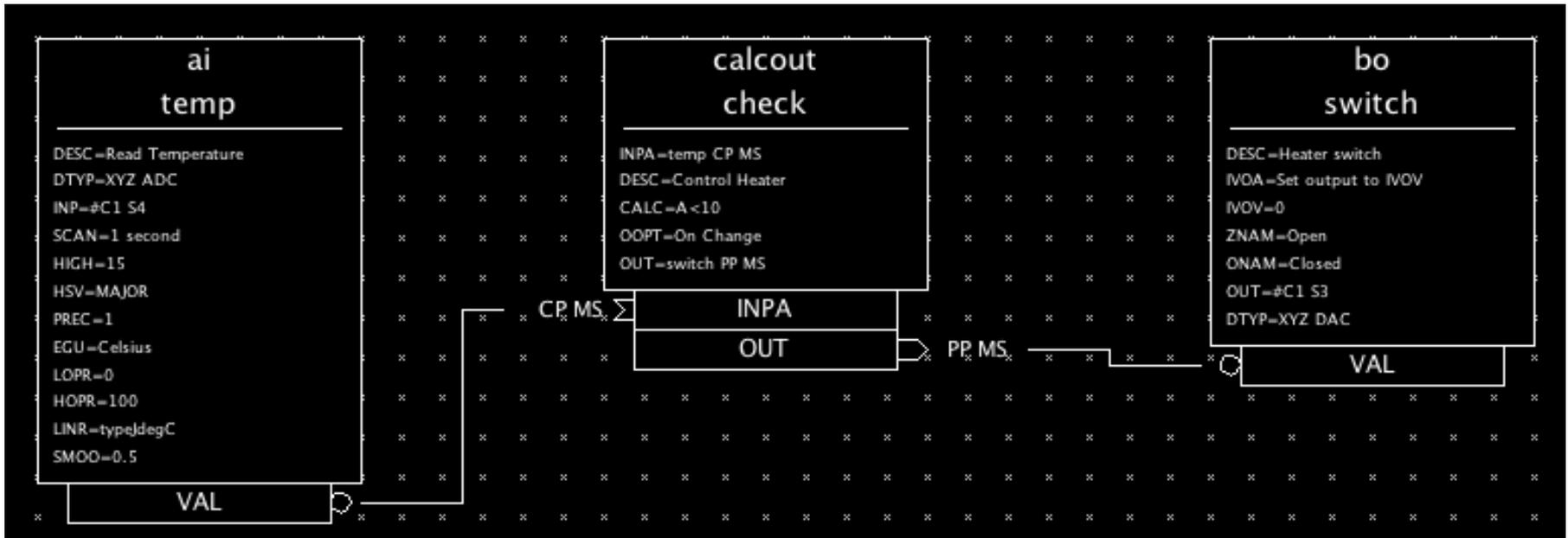
The Example in simplified Code

```
Sensor temp = open_device (...);  
Switch switch = open_device (...);  
  
Loop:  
    if (temp.value() < 10)  
        switch.close();  
    else  
        switch.open();  
  
    delay(1.0);
```

What we omitted

- **Error checking**
- **Code comments**
- **Apply some smoothing to the temperature reading to filter noise.**
- **Send current temperature and switch state to network clients (operator display).**
- **Attach a time stamp to the data, so that network clients can see for example when the switch was last opened.**
- **Send warnings if the temperature is close to the threshold, or an alarm when way above.**
- **Allow runtime changes of the threshold from the remote operator interface.**
- **Allow runtime changes to the scan rate.**
- **Maybe allow runtime changes to the device address?**
- **What if we have more than one fishtank?**

This IOC 'Database' does all that



- **At first glance, this might look much worse than the code, but...**
 - that was simplified code.
 - there's no way the full code for the above would fit on one screen.
 - after learning more about the database (~2 days), this becomes much more readable than somebody else's custom code for the same functionality.

Some Detail on EPICS 'Records'

```
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")  
}
```

- **Configuration instead of Programming**
- **"SCAN=1 second" instead of starting periodic thread, delaying until next multiple of 1 second, locking required resources, ...**
- **"SMOO=0.5" configures the smoothing algorithm.**
- **Almost any field in any record is accessible via network at runtime**
 - **Change scan rate, smoothing, ...**

IOC Database

- A single analog record often handles the scanning, signal conditioning, alarming of a temperature, pressure, or similar analog reading.
- Combined with binary and computational records, it can express most of the **data flow** logic for a front-end computer
 - Avoiding the pitfalls of real-time, multithreaded and networked programming.
- One can have thousands of records in one IOC.
 - Well suited for systems with high signal counts, like vacuum or water systems with relatively simple logic but many, many sensors and valves.
- kHz-rate processing with record chains is doable
 - Of course limited by CPU. Not 1000nds of kHz rate-records...

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

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**

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 Processing

- **SCAN field is a menu choice from**
 - **Passive (default)**
 - **Periodic — “0.1 second” .. “10 second”**
 - **I/O Interrupt (if device supports this)**
 - **Soft event — EVNT field**
- **The number in the PHAS field allows processing order to be set within a scan**
 - **Records with PHAS=0 are processed first, then PHAS=1 etc.**
- **Records with PINI=YES are processed once at startup**
- **A record is also processed whenever any value is written to its PROC field**
- **A record’s FLNK field processes another record after current record is ‘done’**
- **INP , DOL fields can use “PP” to process a passive record before reading. OUT field can use PP to process after writing**

Example “counter.db”

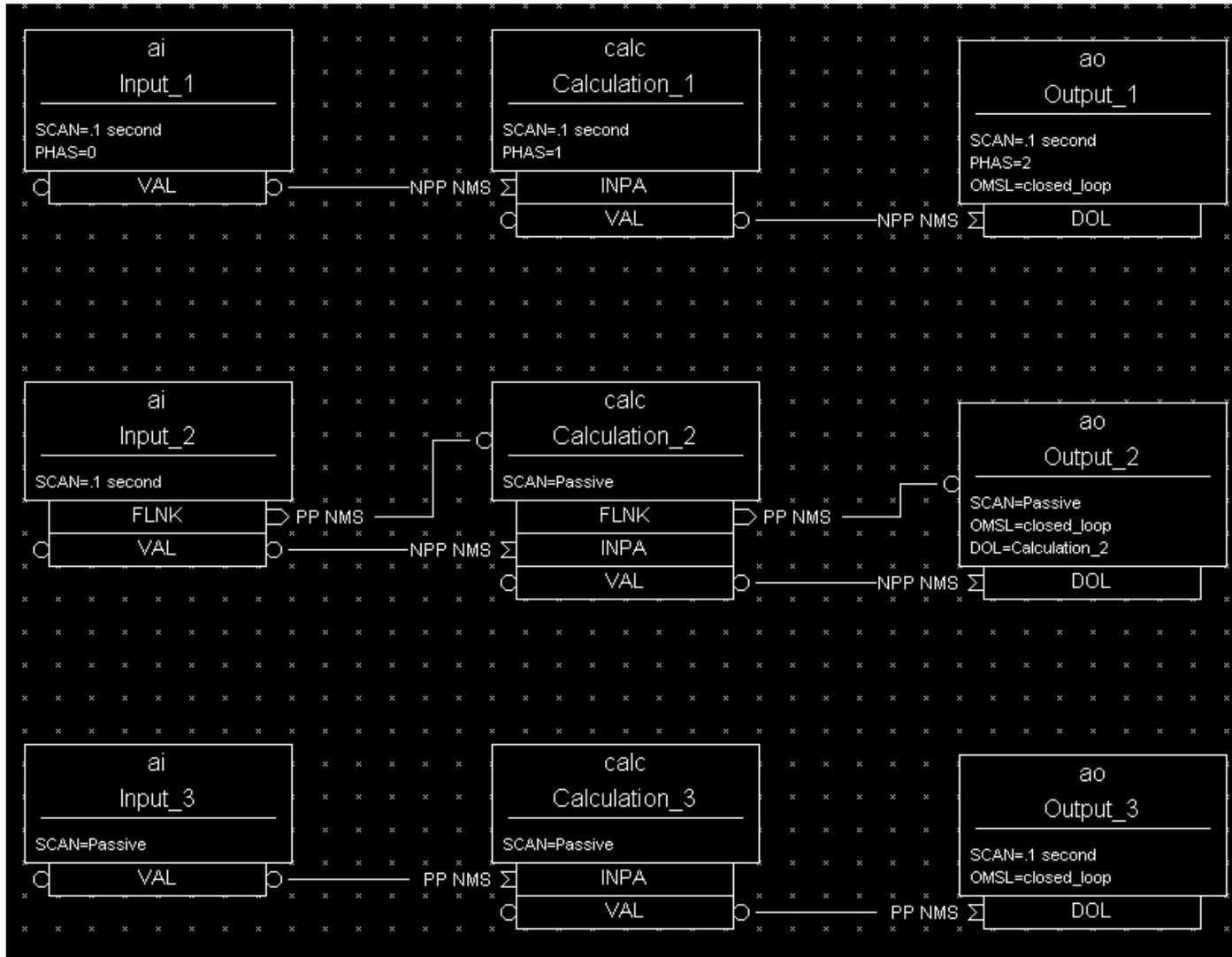
```
record(bi, "enable")
{
  field(DESC, "Enable counter")
  field(ZNAM, "Off")
  field(ONAM, "On")
  field(PINI, "YES")
  field(INP, "1")
}
```

```
record(ao, "limit")
{
  field(DESC, "Counter limit")
  field(PINI, "YES")
  field(DOL, "10")
  field(EGU, "ticks")
}
```

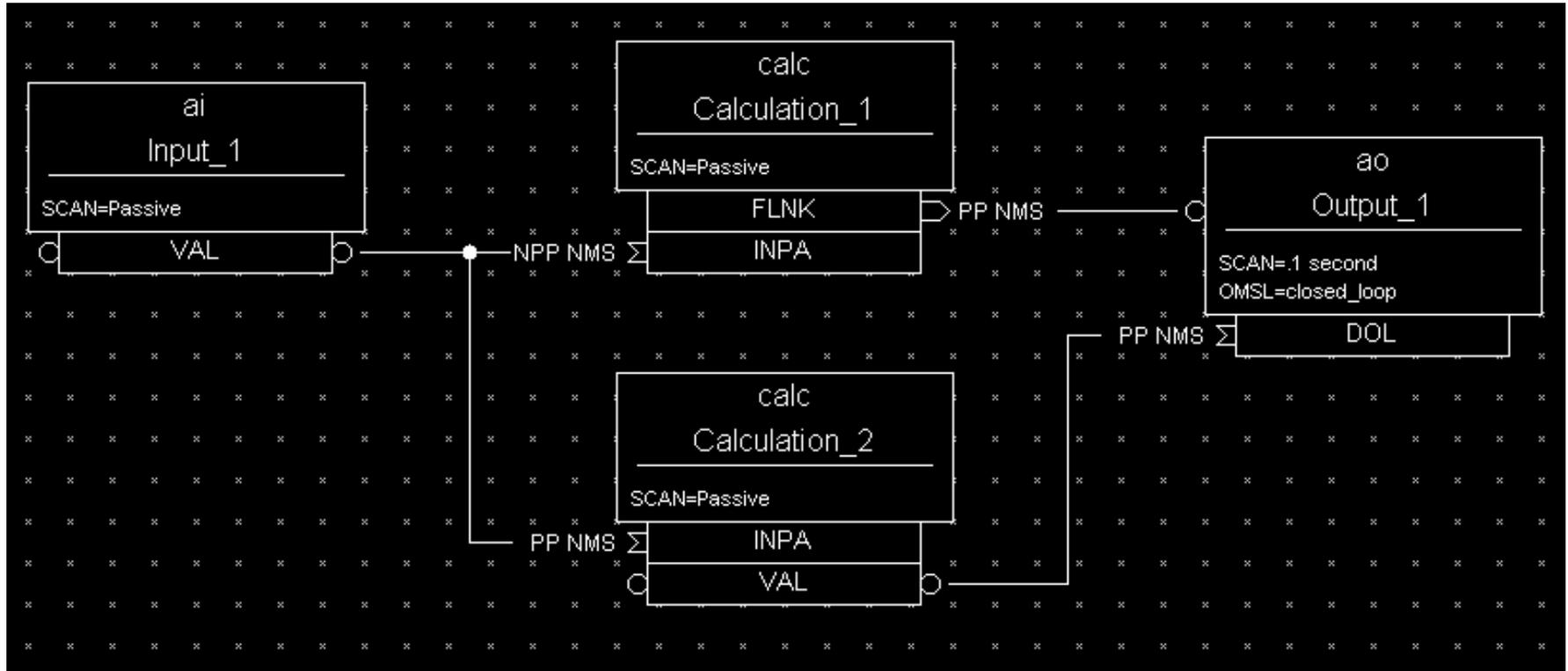
```
record(calc, "counter")
{
  field(DESC, "Counter")
  field(SCAN, "1 second")
  field(INPA, "enable")
  field(INPB, "limit")
  field(INPC, "counter")
  field(CALC, "(A && C<B)?(C+1):0")
  field(EGU, "ticks")
  field(LOW, "3")
  field(LSV, "MINOR")
}
```

- **Execute: `softloc -s -d counter.db`**
- **Try `dbl, dbpf` to `enable.VAL`, `limit.VAL` and `counter.TPRO`
`camonitor counter`**

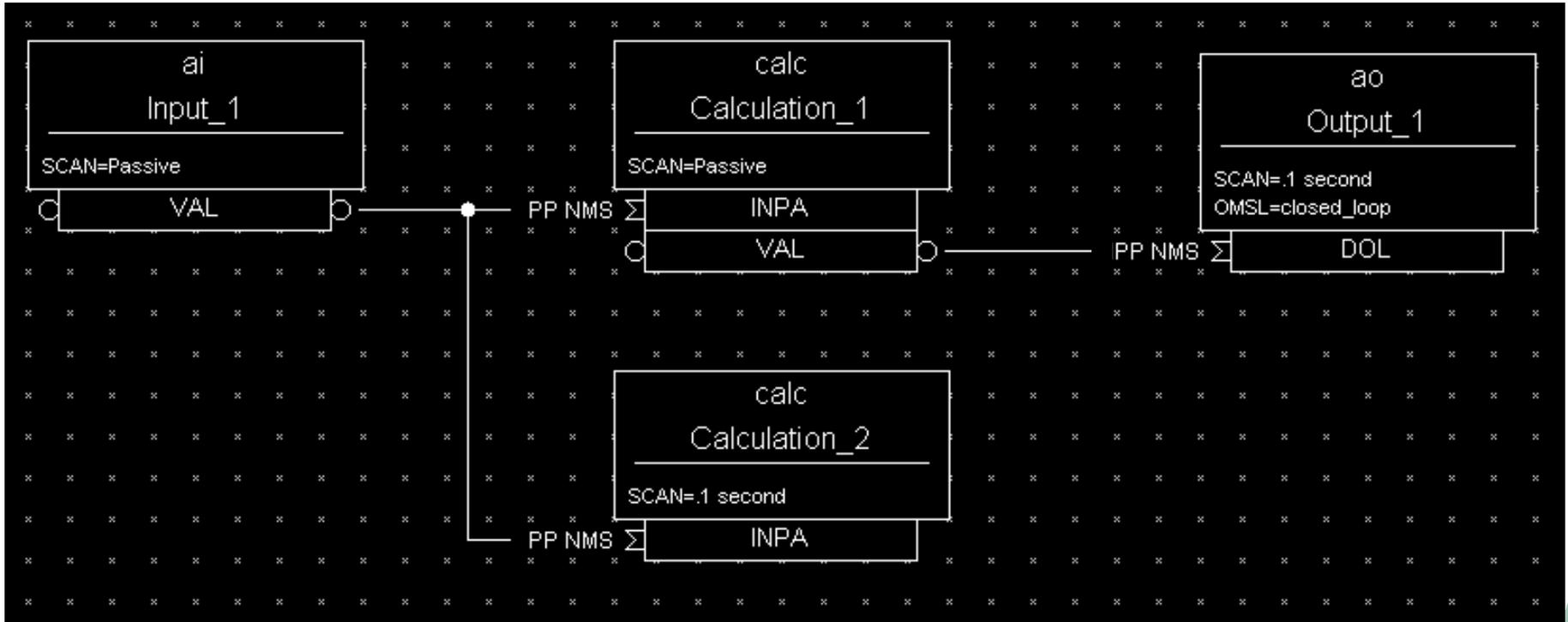
Processing chains



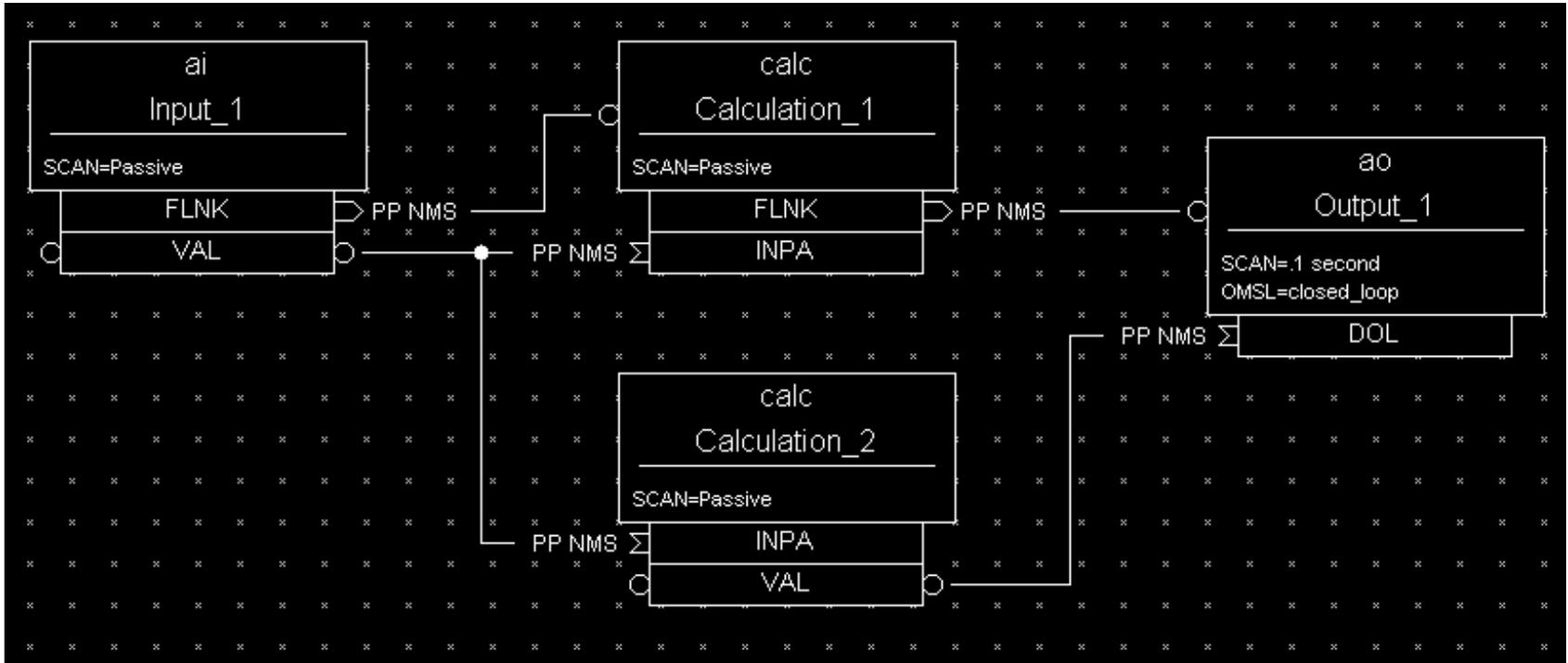
Which record is never processed?



How often is Input_1 processed?



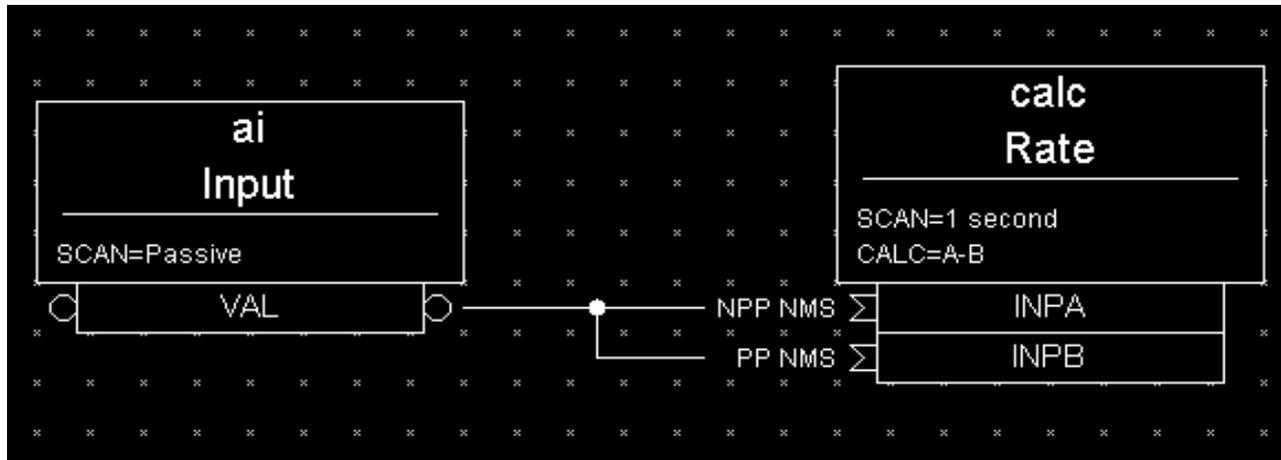
How long will this take?



- **PACT: Processing Active**

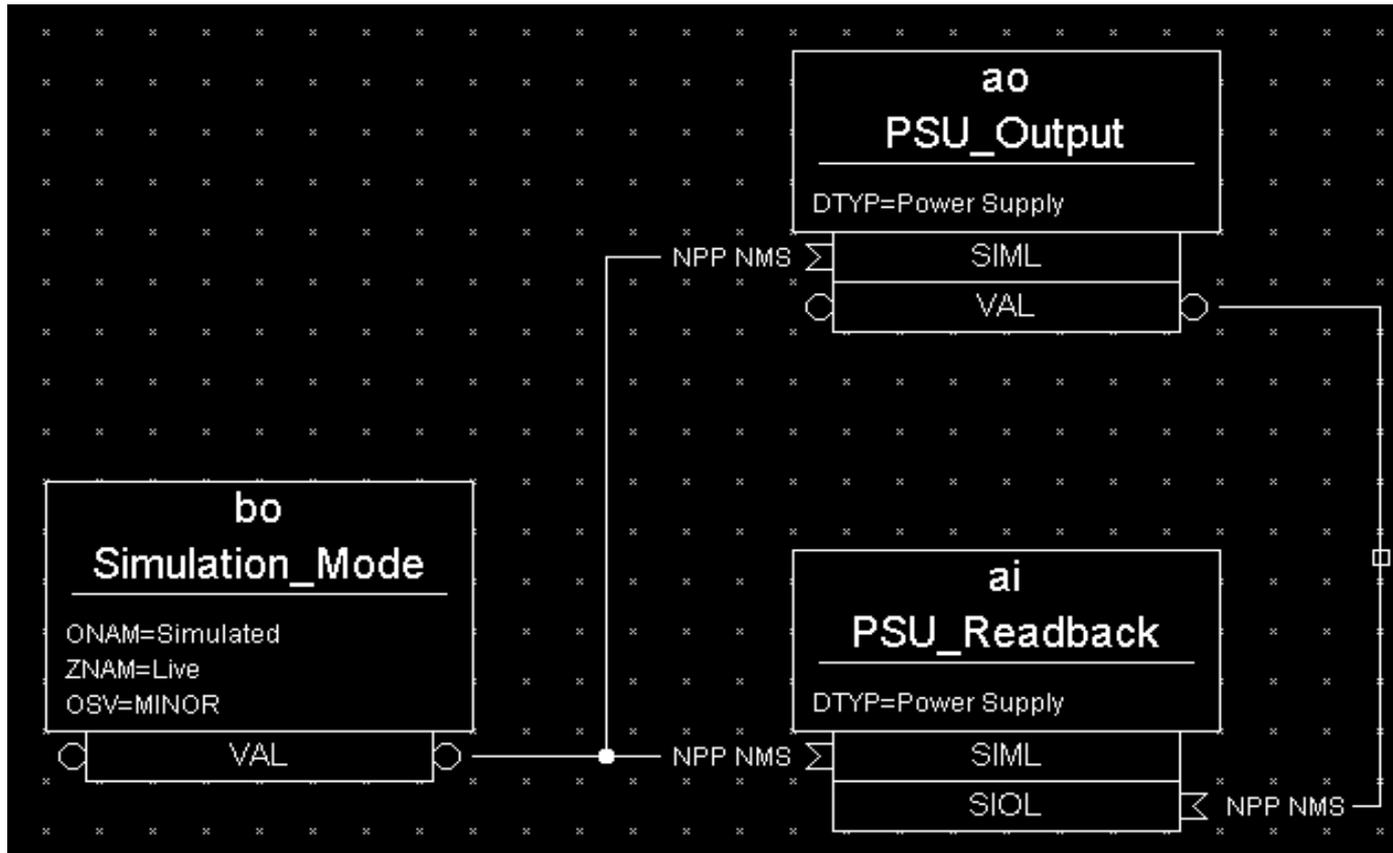
Rate Of Change Example

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.

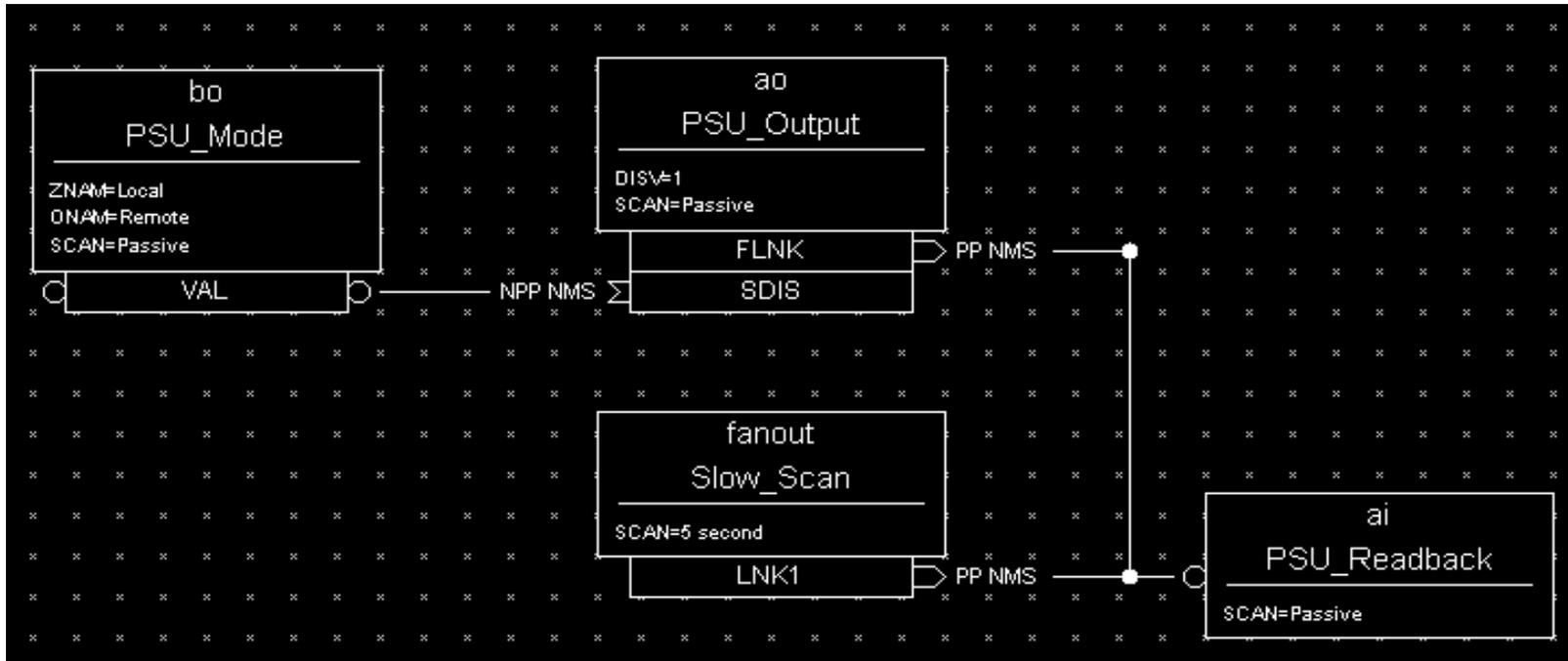
Simulation Mode



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

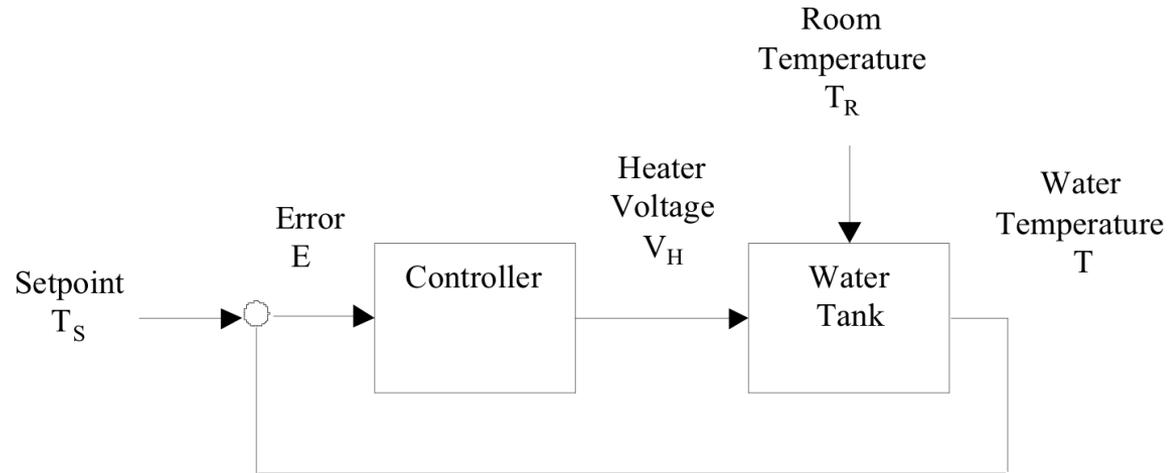
Slow Periodic Scan with Fast Change Response



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 |



- 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/DRVH**

```
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:
#  $P = UI = U^2 / R$ ,  $R \sim 12$  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
#  $T(n+1) = T(n) + [T_{room} - T(n)] * Isolation\_factor$ 
#   + 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

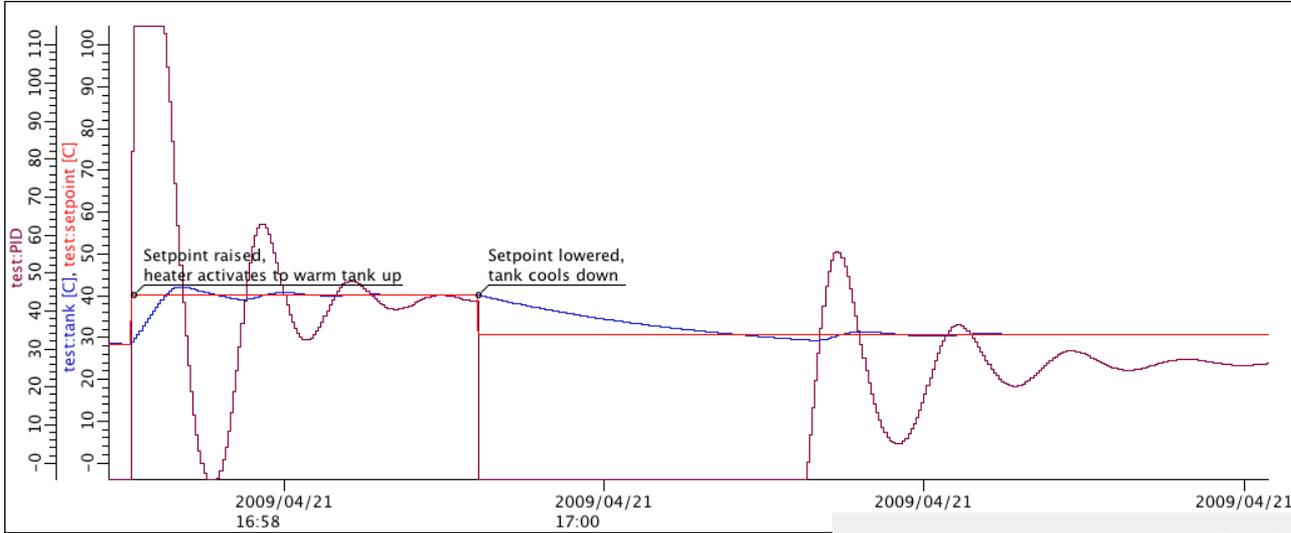
B - error

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



Room

°C

40
35
30
25
20
15
10
5
0

25

25.00

Tank

°C

100
80
60
40
20
0

30.71

Isolation Factor:
0.010000

Heat Capacity:
0.001000

Sensor:

Set Point

50 60 70 80 90 100 110

30.71

Heater

Voltage

110
100
90
80
70
60
50
40
30
20
10
0

26.28 V

57.07 W

Controller

Output:

0 50 100

26.28

PID P: 10.000000

PID I: 5.000000

Error: -0.00

Error Integral: 5.26

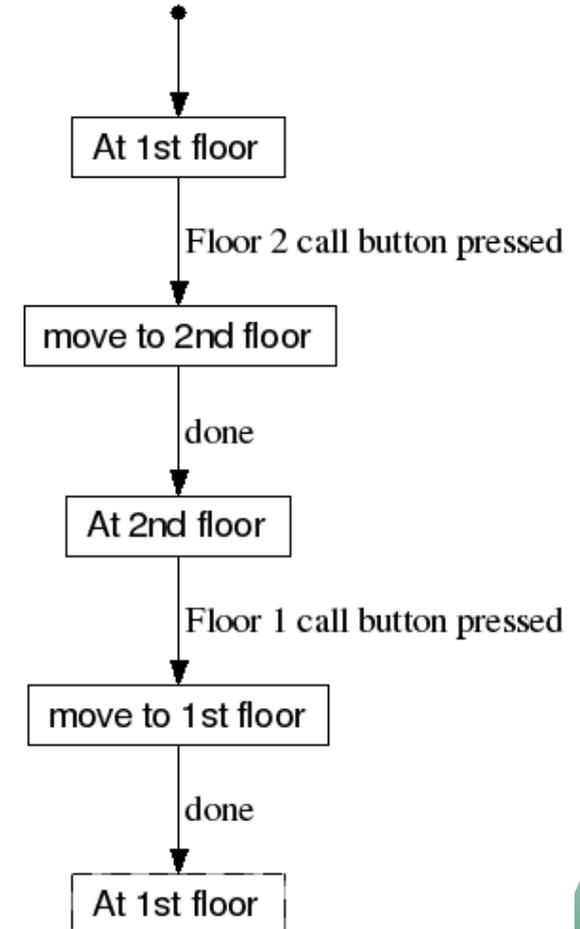
Int. Limit: 5.259835

EPICS Sequencer

- Adds **state-machine** behavior to the IOC

```
program Elevator_Simulation
ss Elevator
{
  state floor1
  {
    when (floor2_call)
    {
      // Almost any C code here...
    } state goto2
  }

  state goto2
  {
    entry
    {
      // Almost any C code here...
    }
  }
  ...
}
```



Summary

- **Database ‘records’ configure the IOC’s data flow**
 - Configuration instead of Code
 - For things that can’t be done in the database, try the sequencer
- **There’s more**
 - Fields MDEL/ADEL/HYST, bo.HIGH
 - Access security
- **See <http://aps.anl.gov/epics> for**
 - IOC Application Developers’ Guide
 - Record Reference Manual
 - VDCT, “Visual” Database config. tool
 - Better (longer) Database training slides