

EPICS Database

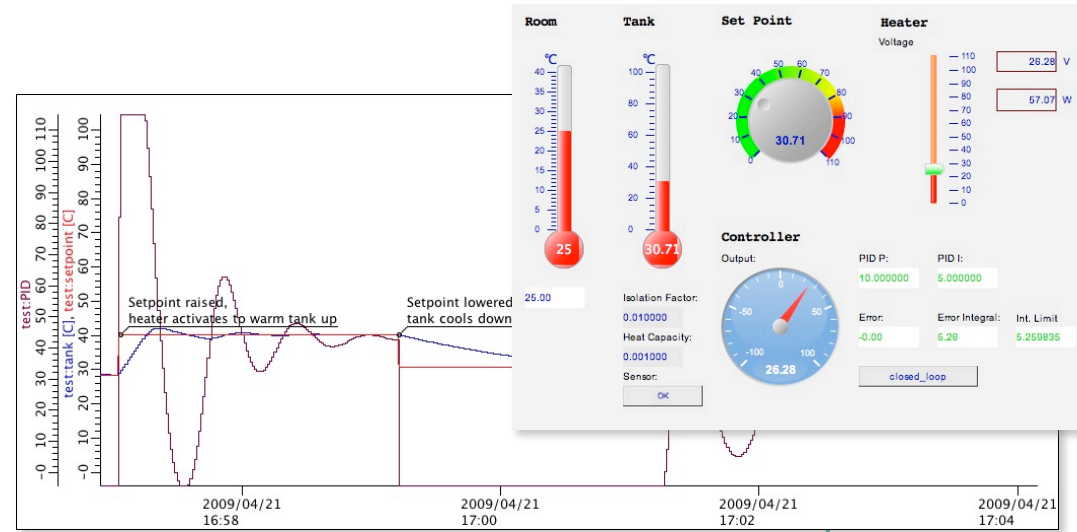
Kay Kasemir

Many slides from Andrew Johnson,
APS/ANL

Jan. 2022

Distributed EPICS Setup

- Operator Interface



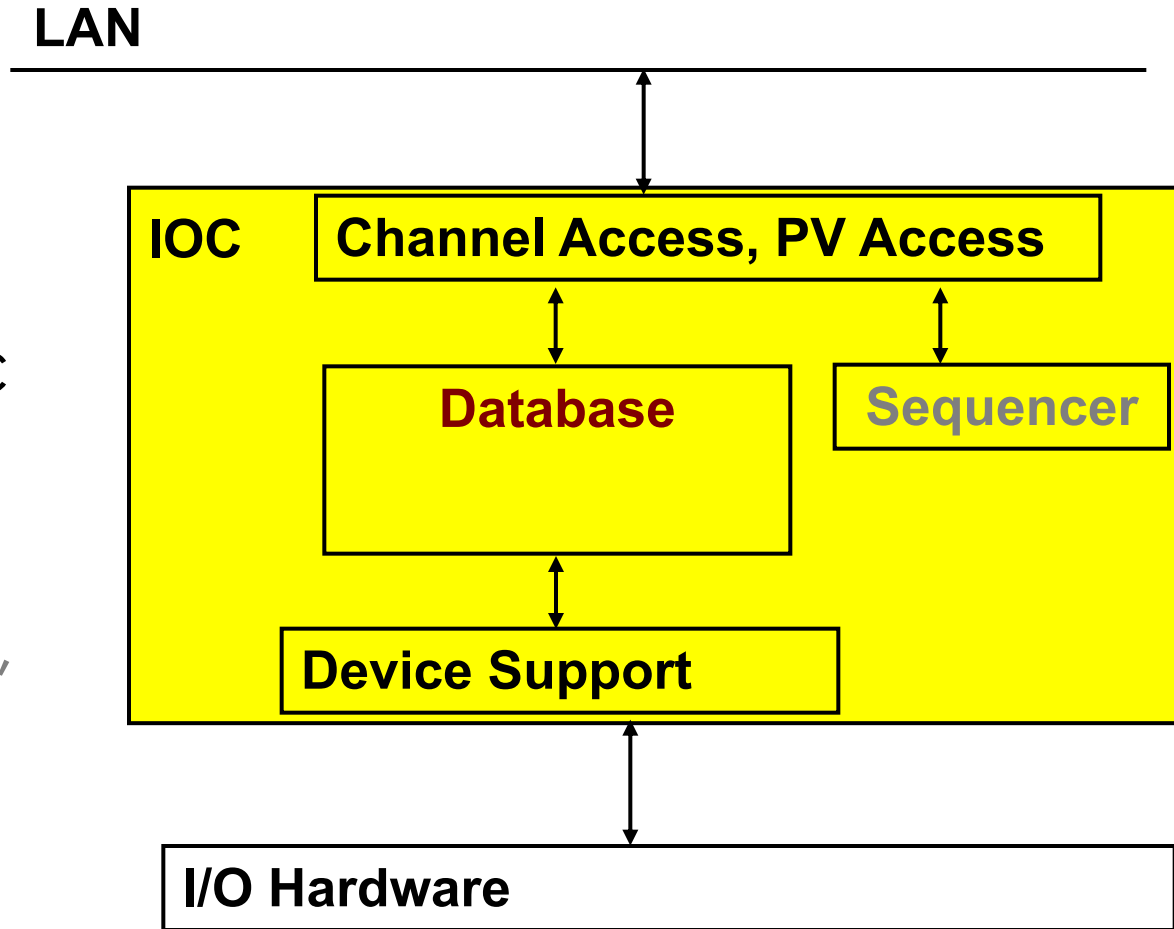
Channel Access
(pvAccess)

- Input/Output Controller (IOC)



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, ...)

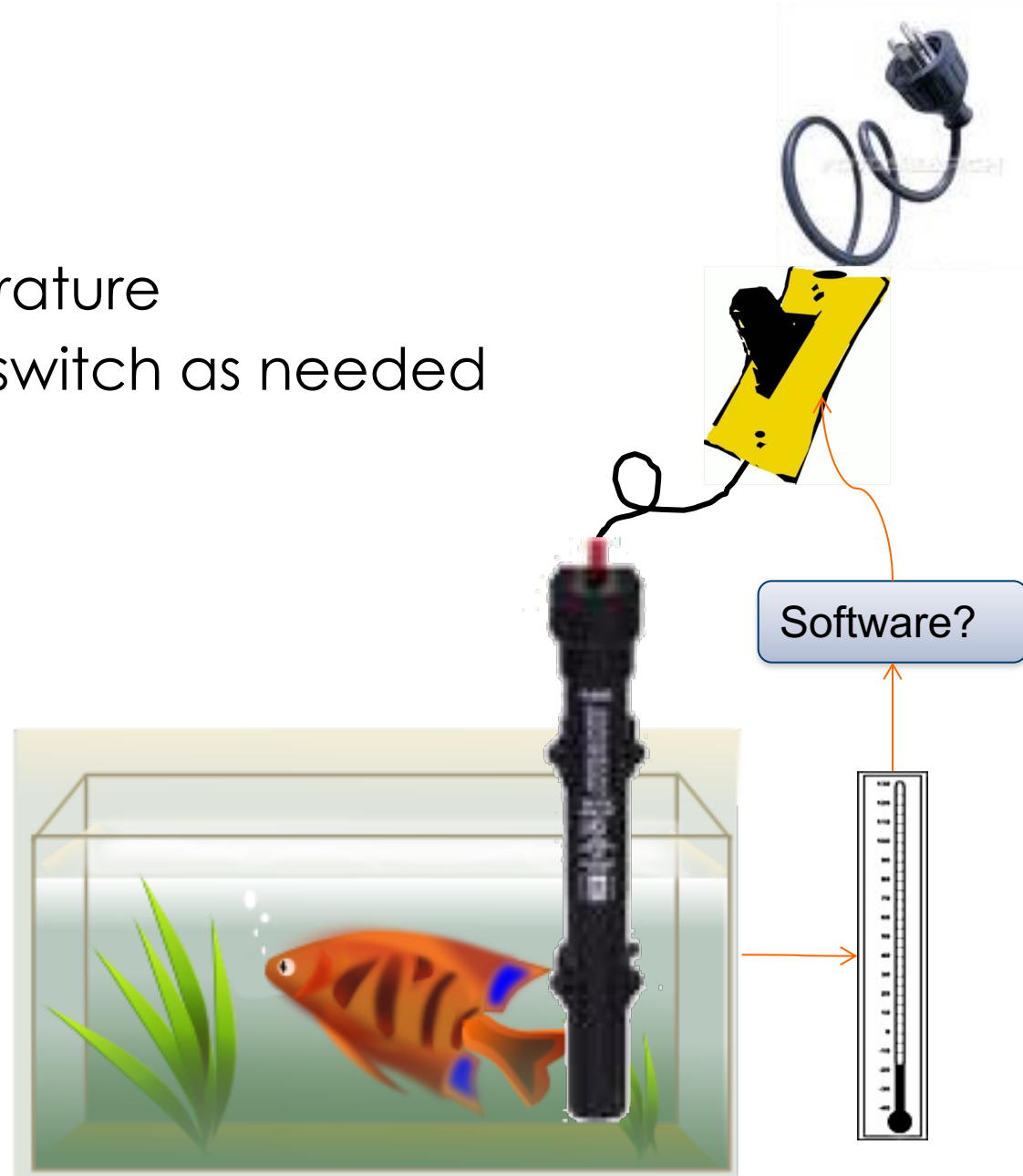
IOC Database

- 'iocCore' software loads and executes 'Records'
 - Configuration of records instead of custom Coding
- All control system toolboxes have (better?)
 - GUI tools
 - Network protocols
 - Hardware driversbut 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 DAC")  
  field(INP, "#C1 S4")  
  field(PREC, "1")  
  field(LINR, "typeJdegC")  
  field(EGU, "Celsius")  
  field(HOPR, "100")  
  field(LOPR, "0")  
  field(SMOD, "0.5")  
  field(HIGH, "15")  
  field(HSV, "MAJOR")  
}
```

Analog Input
Record

SCAN Field

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

Binary Output
Record

```
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")  
}
```

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)
 - Type (determines fields and their functionality)
 - 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, field values, device support
- ... when they are processed
 - Records process periodically or when triggered by events or other records

No action occurs unless a record is processed

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`

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
 - UDF: Undefined? Never processed?
 - PROC: Force processing
- Either
 - TPRO: Trace processing, set to 1 to debug record processing

Record Scanning

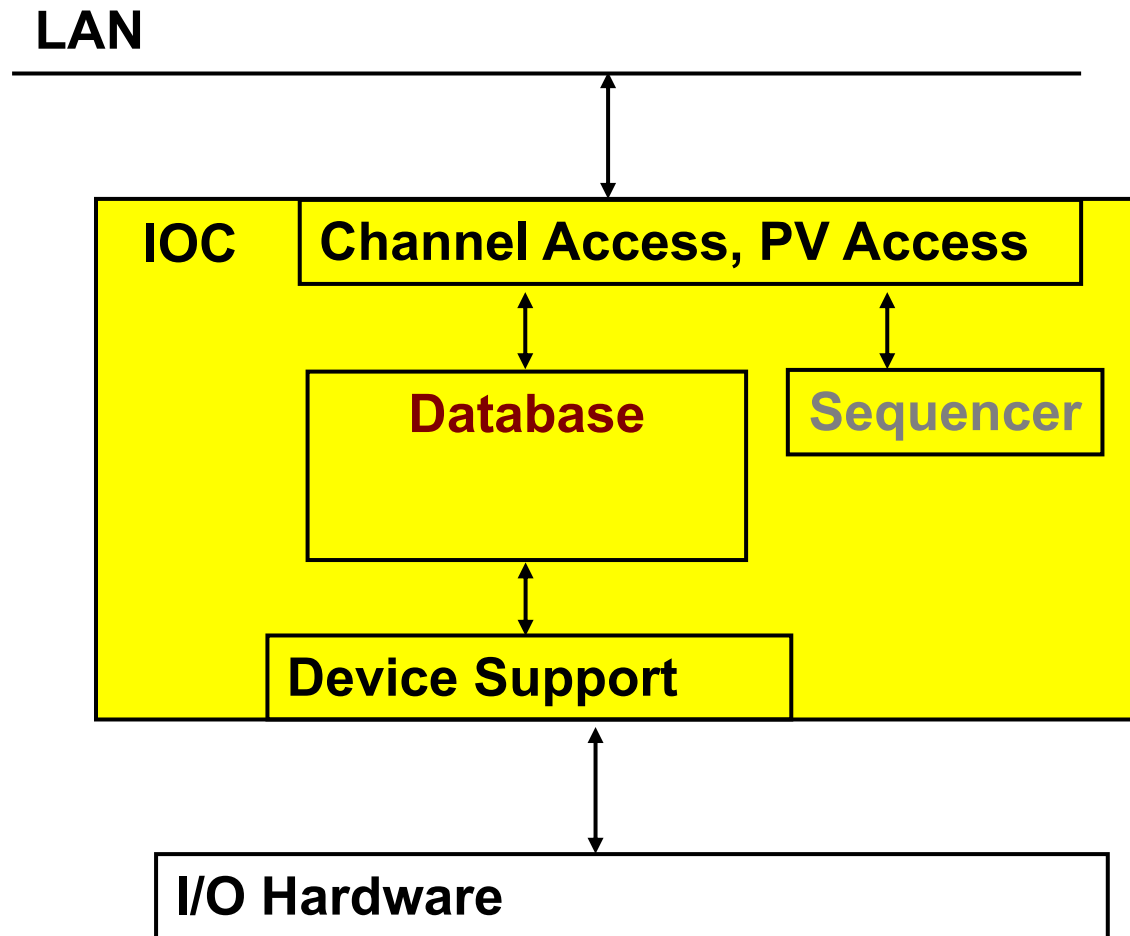
- SCAN field:
 - When processed by other records:
“Passive” (default)
 - Periodically:
“.1 second”, “.2 second”, “.5 second”,
“1 second”, “2 second”, “5 second”, “10 second”
 - On event:
“Event” (EVNT field selects the event),
“I/O Intr” (if device support allows this)
- PHAS field
 - Adds order to records 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

Database vs. the Rest of the IOC

- Record scanning runs at higher priority than Channel Access or PV Access
 - At high CPU load, PVs might not connect while records are still processed
- 0.1 second scan runs at higher priority than 10 second scan
- PRIO field selects priority for async. completion and event-scanned record

Your mileage might vary

- RTOS or plain Linux?



Common Input/Output Record Fields

- DTYP: Device type
- INP/OUT: How to read/write, format depends on DTYP
- RVAL: Raw value (e.g. 16 bit integer)
- VAL: Engineering unit value (e.g. 64bit float)

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

Extending "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
- SMOO: Smoothing
 - $VAL = (1-SMOO)*new_value + SMOO*last_value$
 - SMOO=0: VAL=new_value, default behavior
 - SMOO=1: VAL=last_value, defunct behavior
 - $0 < SMOO < 1$: VAL 'smoothly' follows latest reading
- LINR: Linearization (None, Slope, breakpoint table)
 - EGUL, EGUF, ESLO, EOFF: Parameters for LINR
- LOLO, LOW, HIGH, HIHI: Alarm Limits
 - LLSV, LSV, HSV, HHSV: Associated alarm severities

Binary Record Fields

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

Record Links

- 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"
- Input links may be
 - Constant number: "0", "3.14", "-1.6e-19"
- A record's FLNK field processes another record after current record is 'done'

Database 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 (should be the default?)
 - NMS: non-MS (default)
 - MSS: Maximize Severity and Status
 - MSI: .. when severity = INVALID (should be the default?)
- Example:

```
field("INP", "other_rec.VAL PP MS")
```

Channel Access Links

When linked record is not in this IOC,
automatically uses Channel Access link

- Flags:
 - PP: Ignored. Not triggering processing on other IOC
 - MS, MSI: Maximize severity (when INVALID)

Channel Access Link Flags

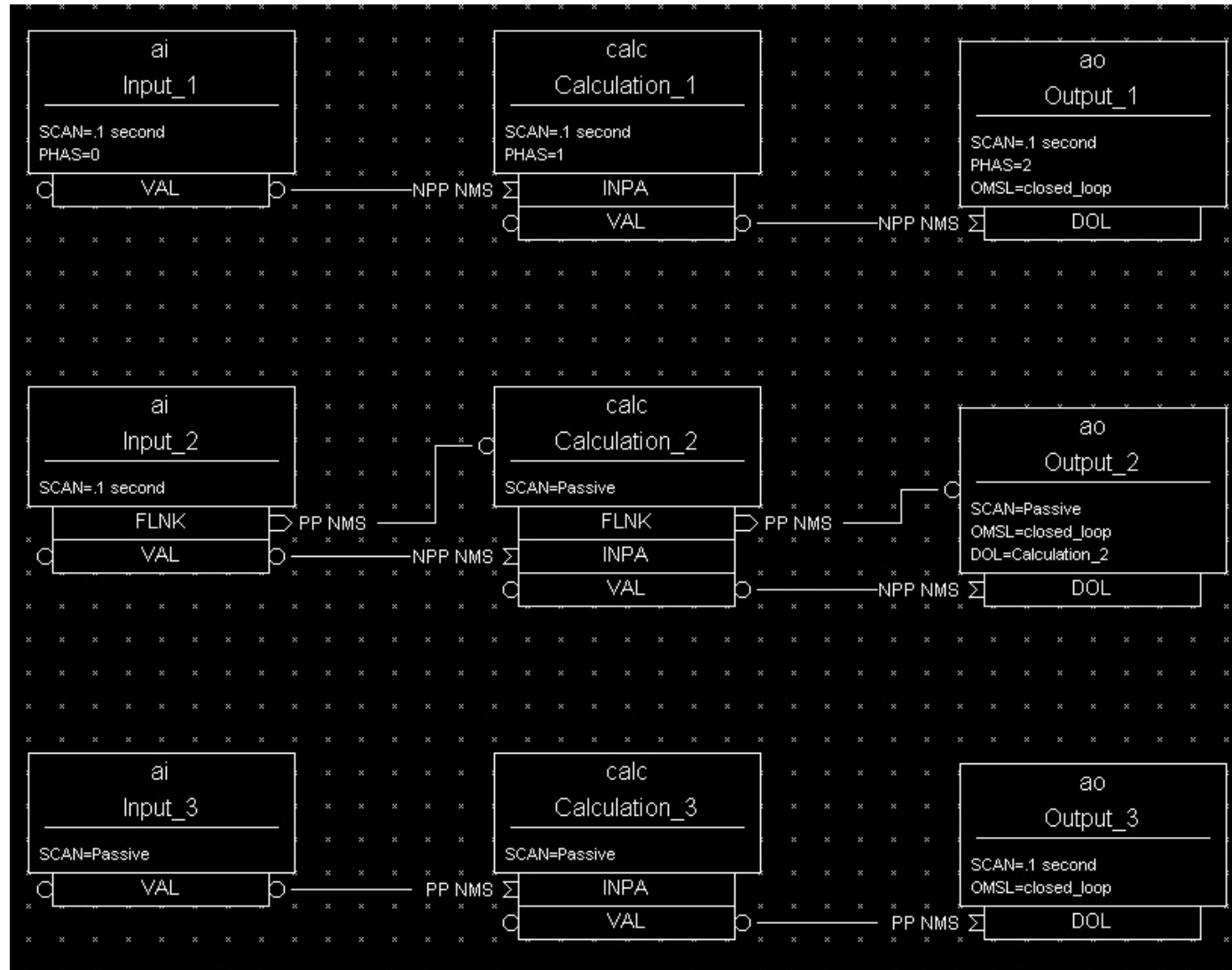
- CA: Force CA link, even though target in same IOC
 - Can be used to break 'lock sets'
- CP: For INP link, process on received CA monitor
 - Typically to cause processing when linked value changes. Details depend on MDEL of source.
- CPP: CP, but only if SCAN=Passive

Forward Links

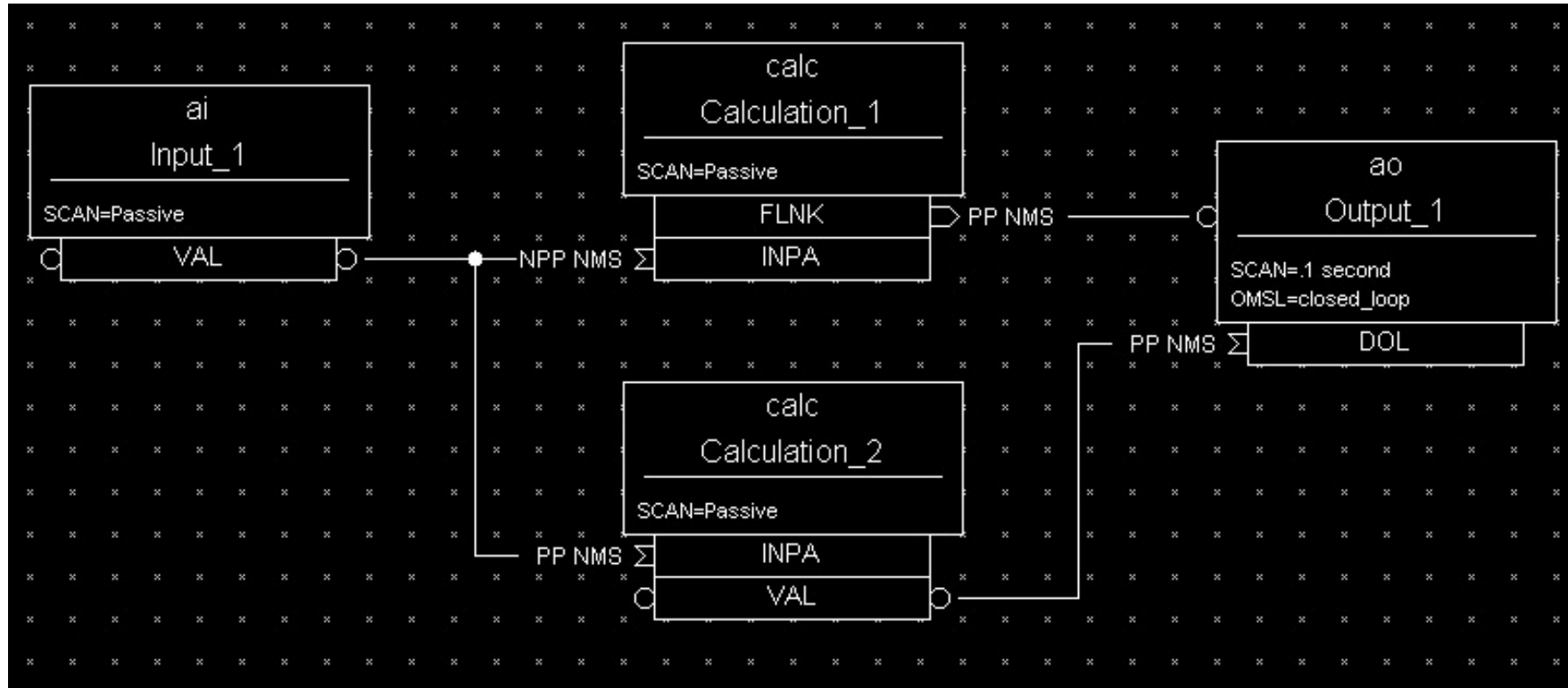
- Trigger processing, not passing data
- Destination processed if SCAN=Passive

- May use CA links
 - Must use FLNK="other.PROC" (other IOC)
or FLNK="other.PROC CA" (same IOC)
 - Always triggers processing even for SCAN!=Passive

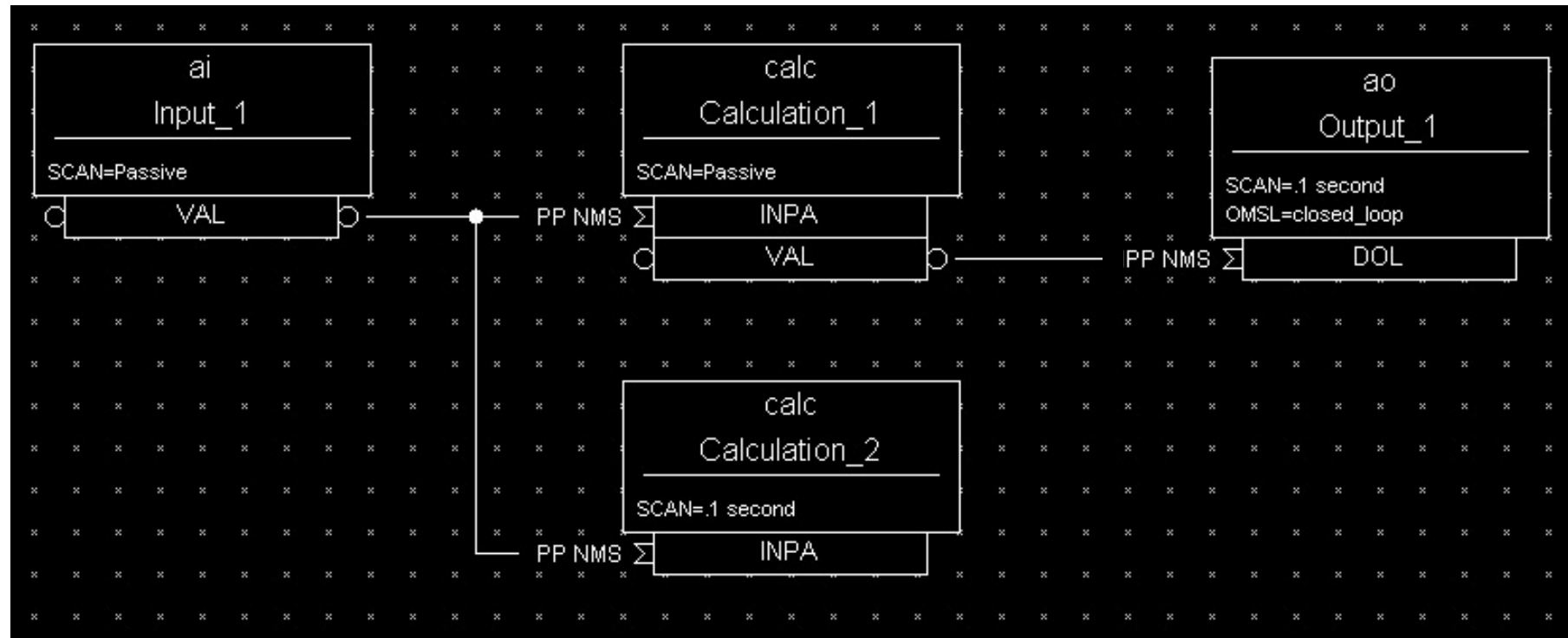
Processing chains



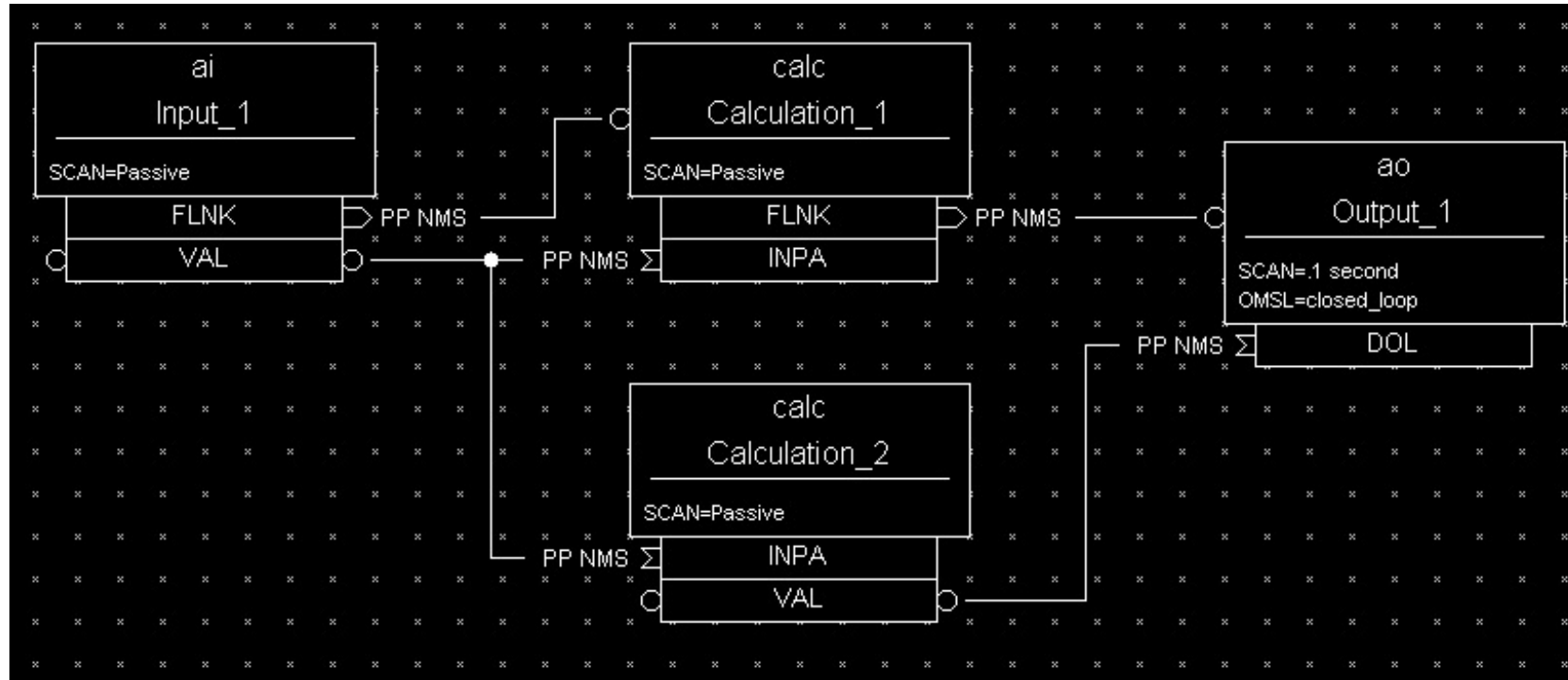
Which record is never processed?



How often is Input_1 processed?



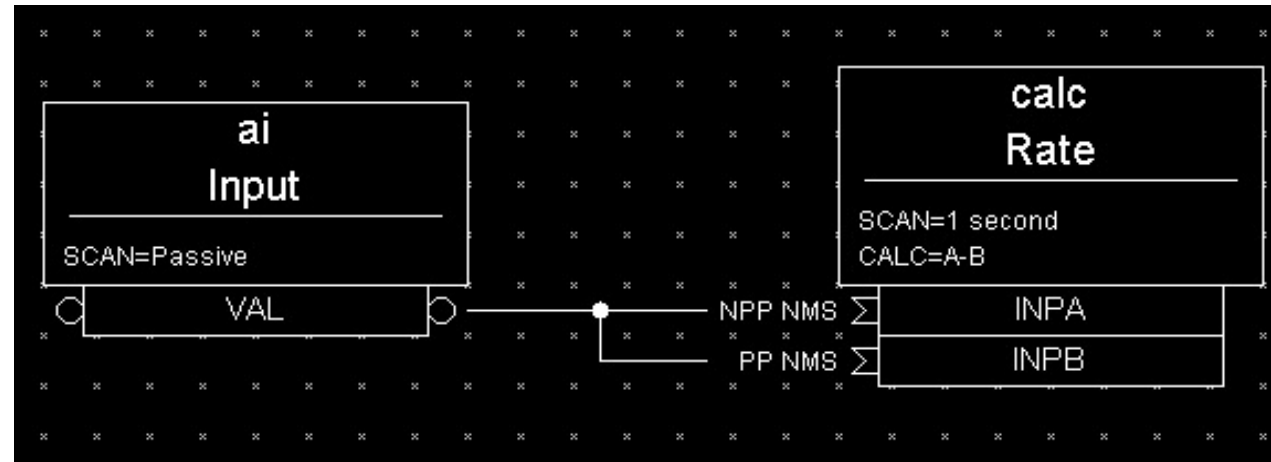
How long will this take?



- PACT: Processing Active

Rate Of Change Example

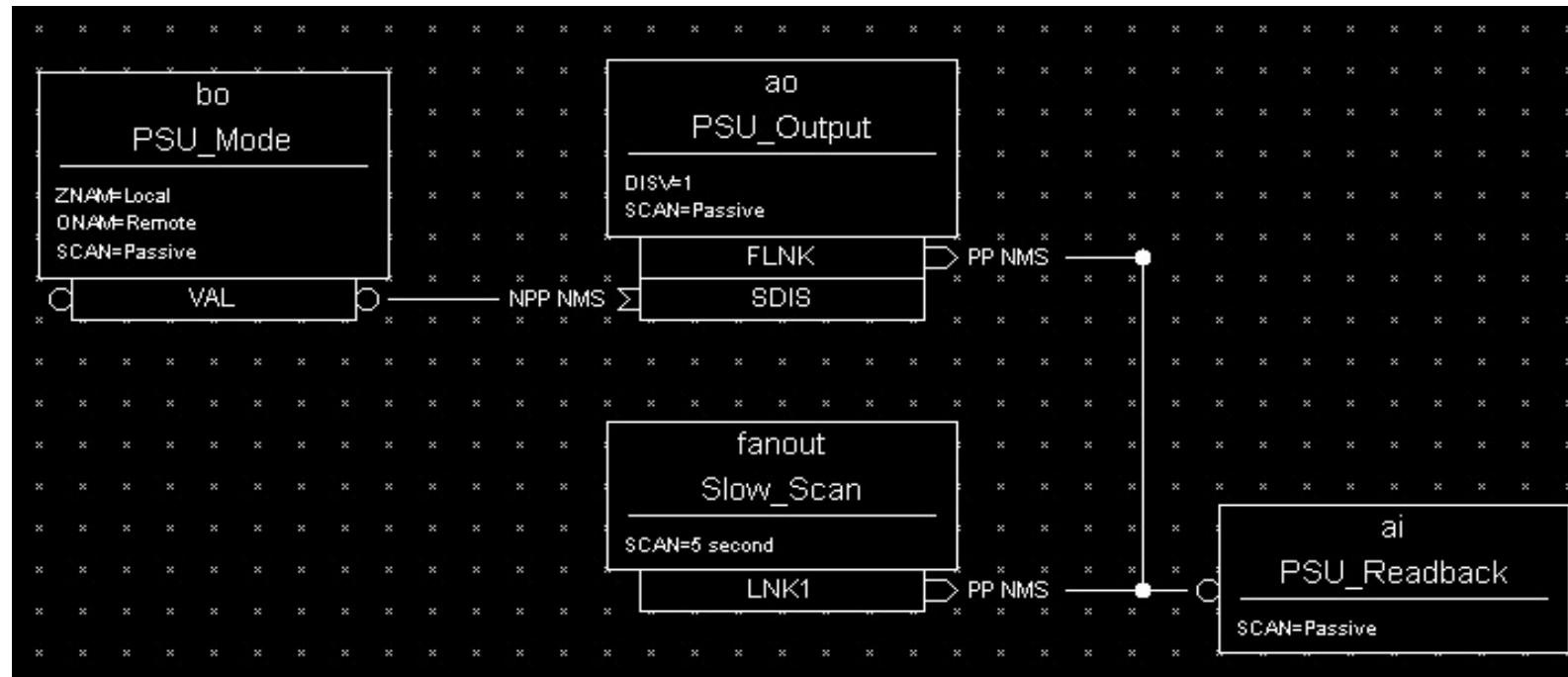
Calculating “Rate-of-Change” of an Input



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.

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.

Device Support

- Records (AI, AO, ..) on their own only read/write from other records
- Device support connects them to hardware
- Hardware Device support is outside of EPICS 'base'. Added to IOC as needed.
- DTYP selects a device support module
- INP/OUT provides detail

Synchronous vs. Asynchronous I/O

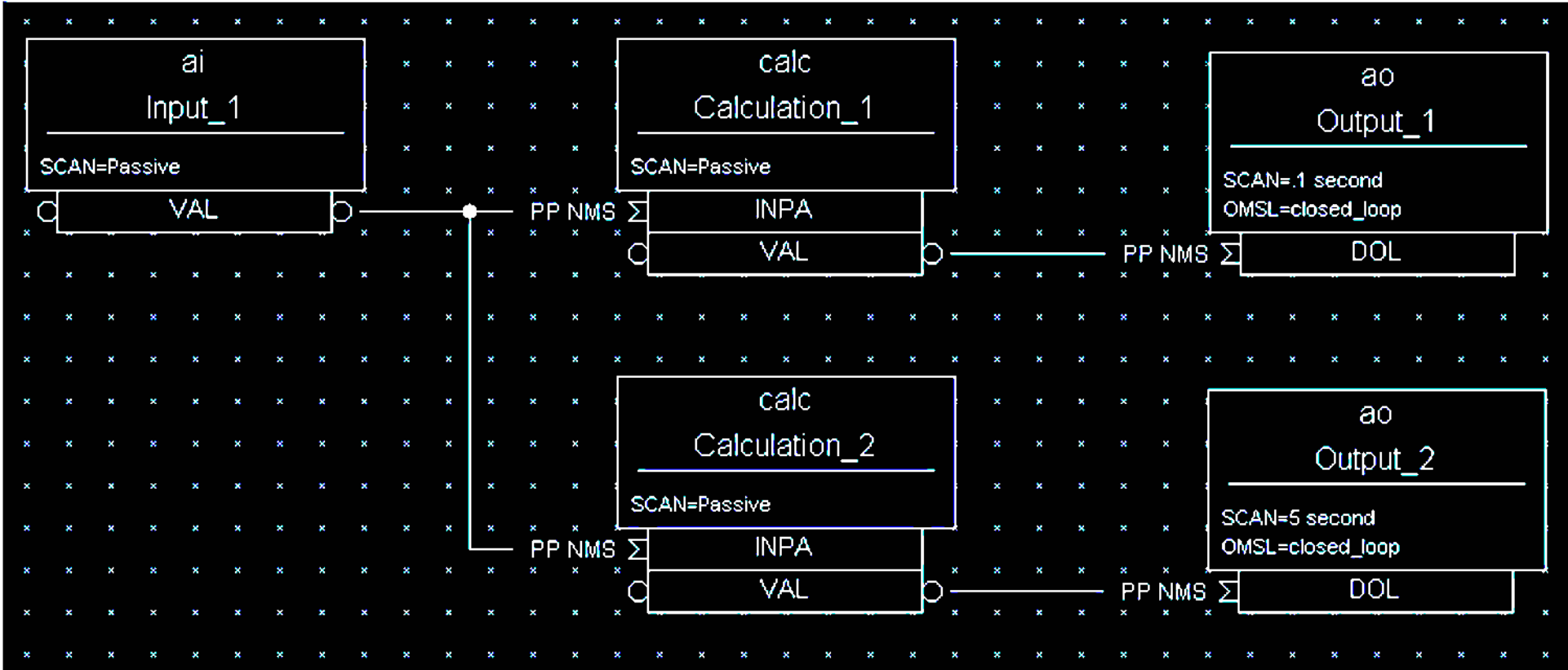
- “Fast”, synchronous device support reads or writes the VAL of a record when the record is processed.
- “Slow”, async support starts reading or writing when the record is processed. The record remains in PACT=true state, and device support triggers completion of processing when data has been read/written.
- Channel Access ‘get/put’ reads/writes current VAL, no matter if record is processing
- Channel Access ‘get/put callback’ will complete once processing completes

'Soft' Device Support

EPICS base includes DTYP=

- “Soft Channel” for AI, AO, BI, BO, ..
 - Reads/writes the VAL field
- “Raw Soft Channel” for AI, AO, BI, BO, ..
 - Reads/writes the RVAL field, converts to/from VAL
- “Async Soft Channel” for AI, AO, BI, BO, ..
 - Performs a get/put callback, waits for completion

What could go wrong here?



Lock-Sets

- Group of records connected by links
- Processing a record locks its lock-set
 - Prevents processing by multiple threads
 - Similar but technically separate from PACT

Tends to transparently avoid problems

.. Unless you are very unlucky.

Then use “CA” flag to break lock set

Record Locking vs. PACT

Depending on its device support, a record can “process” for a long time

- Processing sets PACT and triggers driver to fetch data.
Some time later driver processes the record again, and clears PACT.

Records in lock-set are locked while

- Processing starts, then again when it completes, but not in the in-between times
- Reading a field
- Writing a field

Alarms

- Common fields:
 - SEVR: Alarm Severity
 - NONE, MINOR, MAJOR, INVALID
 - STAT: Alarm Status
 - UDF, READ, WRITE, CALC, HIGH, STATE, ...
- 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 alarms when temperature near boiling point
record(ai, "$(user):tank")
{
    field(DESC, "Water Temperature")
    field(SCAN, "...")
    field(INP, "...")
    field(EGU, "C")
    field(PREC, "1")
    field(HIGH, "90")
    field(HSV, "MINOR")
    field(HIHI, "100")
    field(HHSV, "MAJOR")
}
```

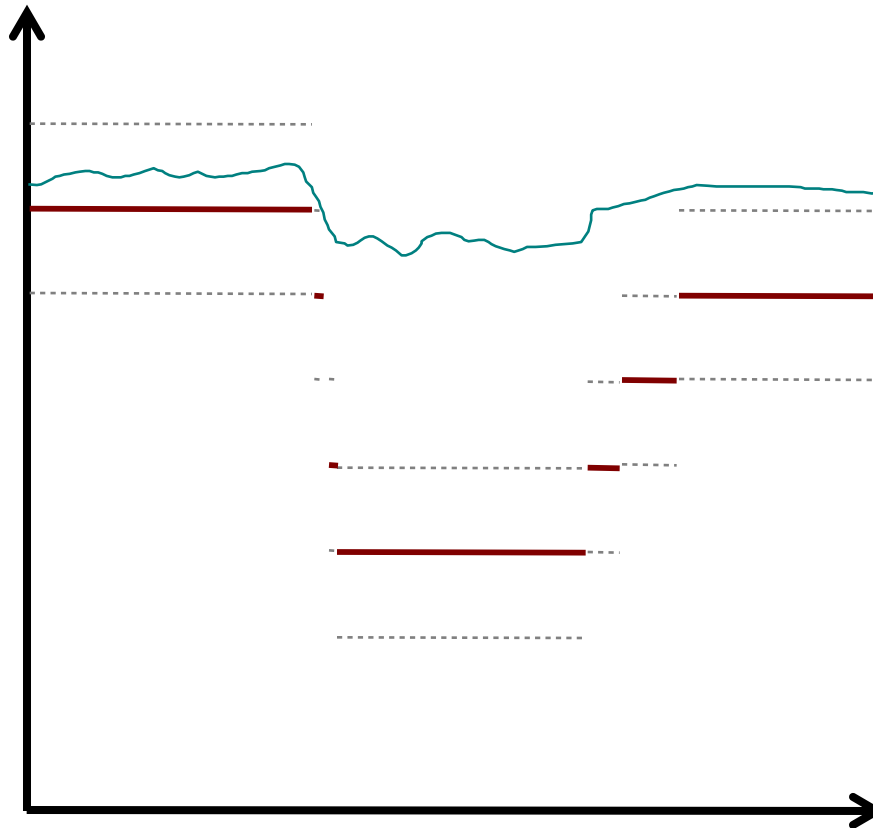
30.0 C

99.3 C

100.0 C

Monitor Dead-Bands

- Analog records send updates to CA clients
 - MDEL Change threshold for most clients
 - ADEL .. for archive client



Record Hints

- Use AO, analog output, for user input
 - DRVL, DRVH can limit the value range
- BO record can be used as timer
 - HIGH field:
When writing VAL=1, remains 1 for HIGH seconds
 - Useful for operator interface buttons:
Button writes 1, record reverts to 0 after HIGH=1

Record Hints

MBBI, MBBO records map states

- Define values and states via ZRVL/ZRST, ONVL/ONST, TWVL/TWST, ...
- Can be used like BI/BO:
ZRVL=0, ONVL=1, ZRST \Leftrightarrow ZNAM, ONST \Leftrightarrow ONAM
- .. For values other than 0, 1:
ZRVL=0, ONVL=255
- Decode Limit Switch state from bits:
ZRVL = 0 ZRST = "Moving"
ONVL= 1 ONST="At Left Limit"
TWVL = 2 TWST="At Right Limit"
THVL = 3 THST = "Broken", THSV=MAJOR

Record Hints

- Use CALCOUT for if-then-else logic
 - INP* and CALC as in CALC record
 - OOPT “On Change”, “When Zero”, “Transition to Zero” etc.
 - Output can either use CALC or a separate OCAL
- SEQ, FANOUT, DFANOUT can process a list of records
 - Simply process or writing values
 - SEL can change from processing all to processing selected records

Record Hints

- COMPRESS record can
 - Keep last N values in circular buffer
 - Compute average, min or max of array
- ASUB record can call C code
 - INAM, SNAM: Name of initial() and sub()
 - Many INP* and VAL* fields
- EVENT record can post database event
 - Trigger records with
SCAN=Event and EVNT=that event

Calcout Record Details

- Combines calc record with analog output functionality
 - `INPA`, `INPB`, ..., `INPL` and `CALC` to compute `VAL`
 - `OUT` to which `VAL` is written
 - `OOPT` determines if/when `OUT` is written
 - “Every Time”, “On Change”, “When Non-zero”, “Transition to Zero”, ...
 - By default, `VAL` is written, but can also configure
 - field(`OCAL`, “A/B+...”) computes alternate `OVAL`
 - field(`DOPT`, “Use OCAL”) selects writing `OVAL` instead of `VAL` to `OUT`
- ➔ Two calculations and an ‘if’ type selector within one record

Calcout Example

```
• record(calcout, "Corrector")
{
    field(SCAN, ".1 second")
    field(INPA, "Enable")
    field(INPB, "Setpoint")
    field(INPC, "Readback")
    field(INPD, "17.54")
    field(CALC, "A")
    field(OOPT, "When Non-zero")
    field(DOPT, "Use OCAL")
    field(OCAL, "D*(B-C)")
    field(OUT, "SteeringMagnet PP")
}
```

→ CALC to determine if we should do anything,
OCAL for actual computation

'synApps' Records outside of EPICS base

- MOTOR record
 - A whole ecosystem for controlling motors
- BUSY record can be used to support put-callback
 1. Some Setpoint record FLNKs to logic that sets BUSY record = 1
 2. When device reaches setpoint, set BUSY record VAL=0

➔ CA 'put-callback' to setpoint will complete after device reached the setpoint

Motor Record Details

- Record of 100+ fields

<https://epics.anl.gov/bcda/synApps/motor/motorRecord.html>

- Basic control

- VAL Desired motor position. Supports put-callback
- RBV Readback value, actual motor position
- DONE Are we done moving the motor?
- HLS, LLS Are we at the high or low limit?
- STOP

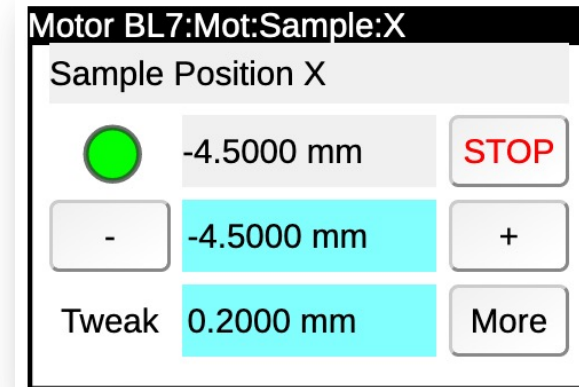
- Resolution

- MRES, DIR, OFF, EGU... Turn motor ticks into “mm” or “degrees”
- Is it a stepper motor or servo? Do we have an encoder, how is it calibrated?

- Motion

- VBAS, VMAX, ACCL, HVEL, JVEL, ... Speed, acceleration
- HLM, LLM, ... Limits of motion range
- BDST, BACC, RTRY, .. Backlash compensation, retries

- More: Homing, “jog”, “tweak”, status,



Max Velocity (VMAX)	5.0000 mm/sec
Velocity (VELO)	5.0000 mm/sec
Base Velocity (VBAS)	0.0000 mm/sec
Acceleration Time (ACCL)	2.0000 sec
Home Velocity (HVEL)	0.2000 mm/sec
Jog Velocity (JVEL)	0.2000 mm/sec
Jog Acceleration (JAR)	0.5000 mm/s/s
Backlash Distance (BDST)	0.0000 mm
Backlash Velocity (BVEL)	0.0000 mm/sec
Backlash Acceleration (BACC)	0.5000 sec
Move Fraction (FRAC)	1.0000 mm
Retry Deadband (RDBD)	0.0010 mm
Max Retry Count (RTRY)	0
Retry Mode (RMOD)	Default ▾
Retry Count (RCNT)	0
Readback Settle Time (DLY)	0.0000

Large or small Records?

One Motor Record?

motor.VAL	Setpoint
motor.RBV	Readback
motor.DONE	Done?
motor.HLS	At high limit?
motor.MRES	Steps per unit
...	

Collection of Records?

```
record(ao, "$(M)_Set")
record(ai, "$(M)_Pos")
record(bi, "$(M)_Done")
record(bi, "$(M)_AtHighLim")
record(ao, "$(M)_StepsPerUnit")
record(waveform, "$(M)_Profile")
```

*A servo motor doesn't use **steps per unit**.*

*A stepper might support a **motion profile***

→ There is no commonly shared "Power Supply", "Camera", "PID", .. record.
Preferred approach is collection-of-records.

Time Stamps

- TIME is generally set to the time when the record last processed
- TSE=-2
 - Device support already set the TIME, for example to the exact trigger time obtained from hardware
- TSE=1 ...255
 - Set TIME to the last occurrence of event 1..255, obtained from timing system
- TSEL
 - Allows fetching the time stamp from another record

Linear Conversion

Analog records convert RVAL \Leftrightarrow VAL

LINR=NO CONVERSION

$$\text{VAL} = \text{RVAL}$$

LINR=SLOPE

$$\text{VAL} = (\text{RVAL}) * \text{ESLO} + \text{EOFF}$$

This assumes device support populates the (integer) RVAL field.
If device support already has a floating-point value, it places that in the (double) VAL field.
For additional conversions, then use a CALC record.

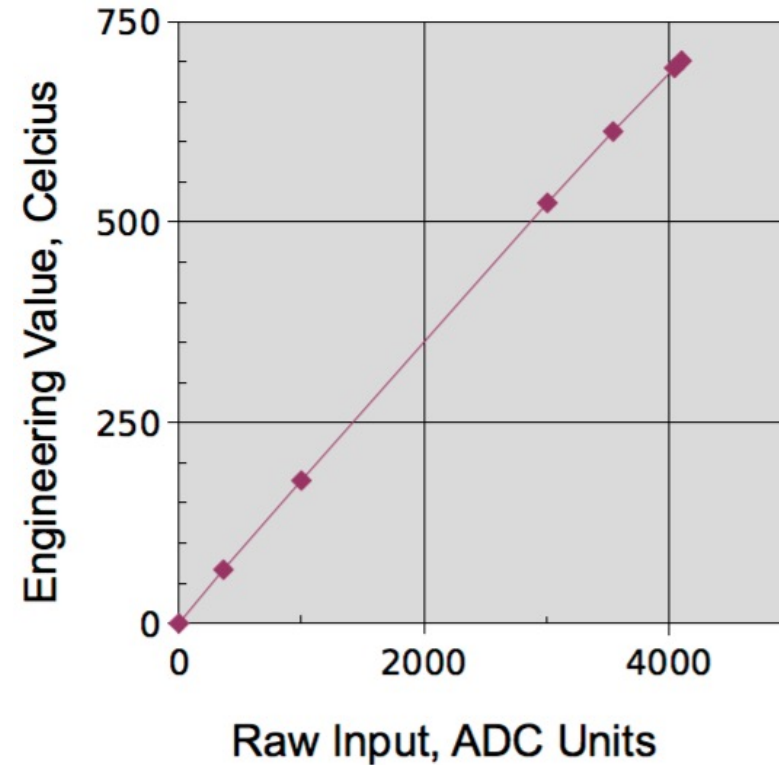
Breakpoint Tables

Analog records can set
LINR= typeKdegC

with this in a *.dbd file:

```
breaktable (typeKdegC)
{
    0.000000    0.000000
    299.268700  74.000000
    660.752744 163.000000
    1104.793671 274.000000
    1702.338802 418.000000
    2902.787322 703.000000
    3427.599045 831.000000
    ...
}
```

Type J Thermocouple



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 <https://epics-controls.org> for
 - IOC Application Developers' Guide
 - Record Reference Manual,
<https://epics.anl.gov/base/R7-0/6-docs/RecordReference.html>