

# EPICS Database Principles

Andrew Johnson  
APS Engineering Support Division

# Outline

- Records
- Fields and field types
- Record Scanning
- Input and Output record types
- Links, link address types
- Connecting records together
- Protection mechanisms
- Alarms, dead-bands, simulation and security

# Database = Records + Fields + Links

- A control system using EPICS will contain one or more IOCs
- Each IOC loads one or more Databases telling it what to do
- A Database is a collection of Records of various types
- A Record is an object with:
  - A unique name
  - A behavior defined by its record type (class)
  - Controllable properties (fields)
  - Optional associated hardware I/O (device support)
  - Links to other records

# Record Activity

- Records are active — they can do things:
  - Get data from other records or from hardware
  - Perform calculations
  - Check values are in range & 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 record type and the settings of its fields
- No action occurs unless a record is processed

# How is a Record implemented?

- A single record definition within an IOC database specifies
  - The record's type
  - One or more unique names for this record
  - Initial values for some of the fields
- Inside the IOC, each record is an instance of a 'C' struct with a data member for each field
  - All records start with a standard set of fields (dbCommon) that the IOC code needs, including pointers to record type information and field metadata
- The record type provides
  - Definitions of all the fields
  - Code which implements the record's unique behavior
- New record types can be added to an application as needed

# A graphical view of a Record (VDCT)

The image displays two windows from a graphical user interface. The left window shows a record definition for 'DemandTemp' with various parameters. The right window is an 'Inspector' for the 'DemandTemp' record, showing a table of parameters grouped into categories like GUI\_COMMON, GUI\_LINKS, GUI\_INPUTS, and GUI\_OUTPUT.

**Record Definition (Left Window):**

ao  
**DemandTemp**  
DESC=Temperature Demand  
SCAN=1 second  
EGU=Celcius  
HOPR=80  
LOPR=20  
DRVH=100  
DRVL=0  
DTYP=Soft Channel  
PINI=NO  
DOL=UserDemand

NPP NMS  $\Sigma$  DOL

**Inspector - DemandTemp (Right Window):**

Inspector - DemandTemp

R DemandTemp (ao)

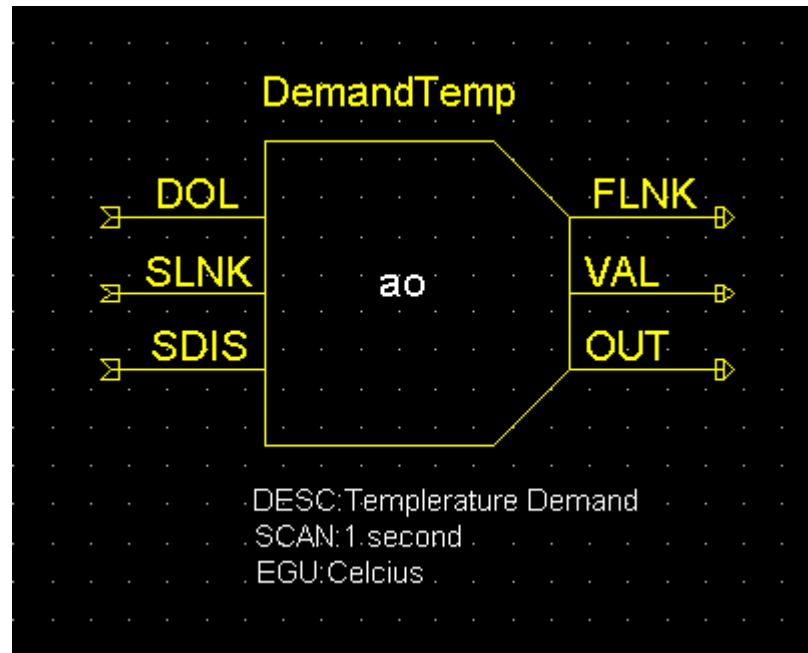
Group Alphabetical DBD Order

GUI_COMMON		GUI_COMMON	
DESC		Temperature Dem...	
ASG			
UDF		1	
GUI_LINKS		GUI_LINKS	
DTYP		Soft Channel	
FLNK			
GUI_INPUTS		GUI_INPUTS	
SIOL			
SIML			
SIMS		<none>	
GUI_OUTPUT		GUI_OUTPUT	
VAL			
OUT			
OROC			
DOL		UserDemand	

Comment

No object selected Frozen

## Another graphical view of a Record (TDCT)



# The IOC's view

## The full .db file entry for an Analogue Output Record

```
record(ao,"DemandTemp") {
  alias("AO:0:0")
  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(DOL,"")
  field(OMSL,"supervisory")
  field(OROC,"0.0e+00")
  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")
}
```

This slide only shows design fields; other fields exist which are only used at run-time



# Fields are for...

- Defining
  - What causes a record to process
  - Where to get/put data from/to
  - How to turn raw I/O data into a numeric engineering value
  - Limits indicating when to report an alarm
  - When to notify value changes to a client monitoring the record
  - A Processing algorithm
  - Anything else which needs to be set for each record of a given type
- Holding run-time data
  - Input or output values
  - Alarm status, severity and operator acknowledgments
  - Processing time-stamp
  - Other data for internal use

# Field types – fields can contain:

- Integers
  - char, short or long
  - signed or unsigned
- Floating-point numbers
  - float or double
- Fixed length strings
  - maximum useful length is 40 characters
- Enumerated/menu choices
  - select one of up to 16 strings
  - stored as a short integer
- Arrays of any of the above types
- Links
  - to other records in this or other IOCs
  - to hardware signals (device support)
  - provide a means of getting or putting a value
- Other private data
  - not accessible remotely

# All Records have these design fields

- NAME*** *60 character unique canonical name for the record (names longer than 28 characters can cause problems)*
- DESC*** *40 character description*
- ASG*** *Access security group*
- SCAN*** *Scan mechanism*
- PHAS*** *Scan order (phase)*
- PINI*** *Process during IOC initialization?*
- PRIO*** *Scheduling priority*
- SDIS*** *Scan disable input link*
- DISV*** *Scan disable value*
- DISS*** *Disabled severity*
- FLNK*** *Forward link*

# All Records have these Run-time fields

*PROC* Force processing

*PACT* Process active

*STAT* Alarm status

*SEVR* Alarm severity

***TPRO*** Trace processing

***UDF*** Non-zero if record value undefined

***TIME*** Time when record was last processed

# Record Scanning

- **SCAN** field is a menu choice from
  - Periodic — 0.1 seconds .. 10 seconds (extensible)
  - I/O Interrupt (if device supports this)
  - Soft event — **EVNT** field
  - Passive (default)
- The number in the **PHAS** field allows the relative order in which records are processed within a scan to be controlled
  - Records with **PHAS=0** are processed first
  - Then those with **PHAS=1** , **PHAS=2** etc.
- The **PINI** field sets if/when records get processed once at startup or when IOC is paused
  - No, Yes, Run, Running, Pause, Paused
- **PRIO** field selects Low/Medium/High priority for Soft event and I/O Interrupts
- A record is also processed whenever any value is written to its **PROC** field

# Input records often have these fields

<i>INP</i>	<i>Input link</i>
<i>DTYP</i>	<i>Device type</i>
<i>RVAL</i>	<i>Raw data value</i>
<i>VAL</i>	<i>Engineering value</i>
<i>LOPR</i>	<i>Low operator range</i>
<i>HOPR</i>	<i>High operator range</i>

## Analogue I/O records have these fields:

<i>EGU</i>	<i>Engineering unit string</i>
<i>LINR</i>	<i>Unit conversion control: No conversion, Linear, Slope, breakpoint table name</i>
<i>EGUL</i>	<i>Engineering unit low value</i>
<i>EGUF</i>	<i>Engineering unit high value (full-scale)</i>
<i>ESLO</i>	<i>Unit conversion slope</i>
<i>EOFF</i>	<i>Unit conversion offset</i>

# Periodically Scanned Analog Input

```
ai
Temperature
-----
DTYP=XY566
SCAN=1 second
PHAS=0
EGU=Celcius
LINR=LINEAR
EGUL=0
EGUF=120
INP=#C0 S0
```

- Analogue Input “Temperature”
- Reads from a Xycom XY566 ADC
  - Card 0 Signal 0
- Gets a new value every second
- Data is converted from ADC range to 0..120 Celsius



# Interrupt Scanned Binary Input

```
bi
VentValve
-----
DTYP=AB-Binary Input
INP=#L0 A0 C3 S5
SCAN=I/O Intr
PHAS=0
ZNAM=Closed
ZSV=NO_ALARM
ONAM=Open
OSV=MAJOR
```

- Binary Input “VentValve”
- Reads from an Allen-Bradley TTL I/O
  - Link 0, Adaptor 0, Card 3, Signal 5
- Processed whenever value changes
- 0 = “Closed”, 1 = “Open”
- Major alarm when valve open

# Most output records have these fields

<i>OUT</i>	<i>Output link</i>
<i>DTYP</i>	<i>Device type</i>
<i>VAL</i>	<i>Engineering value</i>
<i>RVAL</i>	<i>Raw output value</i>
<i>DOL</i>	<i>Input link to fetch output value</i>
<i>OMSL</i>	<i>Output mode select:</i> <b>Supervisory, Closed Loop</b>
<i>LOPR</i>	<i>Low operator range</i>
<i>HOPR</i>	<i>High operator range</i>

## Analogue outputs also have these fields:

<i>OROC</i>	<i>Output rate of change</i>
<i>OIF</i>	<i>Incremental or Full output</i>
<i>OVAL</i>	<i>Output value</i>
<i>DRVH</i>	<i>Drive high limit</i>
<i>DRVL</i>	<i>Drive low limit</i>
<i>IVOA</i>	<i>Invalid output action</i>
<i>IVOV</i>	<i>Invalid output value</i>
<i>RBV</i>	<i>Read-back value</i>

# Passive Binary Output

```
bo
Solenoid
-----
DTYP=XY220
OUT=#C0 S12
SCAN=Passive
PHAS=0
ZNAM=Locked
ONAM=Unlocked
OMSL=supervisory
```

- Binary Output “Solenoid”
- Controls Xycom XY220 digital output
  - Card 0 Signal 12
- Record is only processed by
  - Channel Access ‘put’ to a PP field (e.g. .VAL)
  - Another record Forward Links here
  - Another record writes with PP flag
  - Another record reads with PP flag

Break time...

5 Minute break



# Links

A link is a type of field, and is one of

- Input link
  - Fetches data
- Output link
  - Writes data
- Forward link
  - Points to a record to be processed when this record has finished processing

# Input and Output links may be...

- Constant numeric value, e.g.:
  - 0
  - 3.1415926536
  - 1.6e-19
- Hardware link
  - Address of a hardware I/O signal (fields named **INP** or **OUT** only)
  - The address format depends on the device support layer (**DTYP** field value)
- Process Variable link — names a record, at run-time becomes either
  - Database link
    - Target record must be present in this IOC
  - Channel Access link
    - Target record can be in a different IOC

# Hardware link addresses

<i>VME_IO</i>	<i>#Cn Sn @parm</i> <b><i>Card, Signal</i></b>
<i>INST_IO</i>	<i>@parm</i>
<i>CAMAC_IO</i>	<i>#Bn Cn Nn An Fn @parm</i> <b><i>Branch, Crate, Node, Address, Function</i></b>
<i>AB_IO</i>	<i>#Ln An Cn Sn @parm</i> <b><i>Link, Adapter, Card, Signal</i></b>
<i>GPIB_IO</i>	<i>#Ln An @parm</i> <b><i>Link, Address</i></b>
<i>BITBUS_IO</i>	<i>#Ln Nn Pn Sn @parm</i> <b><i>Link, Node, Port, Signal</i></b>
<i>BBGPIB_IO</i>	<i>#Ln Bn Gn @parm</i> <b><i>Link, Bitbus Address, GPIB Address</i></b>
<i>VXI_IO</i>	<i>#Vn Cn Sn @parm</i>
<i>or</i>	<i>#Vn Sn @parm</i> <b><i>Frame, Slot, Signal</i></b>



# Database links

- These comprise:
  - The name of a record in this IOC  
`myDb:myRecord`
  - An optional field name  
`.VAL` (default)
  - Process Passive flag
    - **NPP** (default)
    - **PP** If the target record has **SCAN**=Passive, process it before reading or after writing the value
  - Maximize Severity flag
    - **NMS** No maximize severity (default)
    - **MS** Maximize severity
    - **MSS** Maximize Status and Severity (new in R3.14.11)
    - **MSI** Maximize Severity when Invalid (new in R3.14.11)
- Example
  - **M1:current.RBV NPP MS**
- NB: Database links with the **PP** flag set do not wait for asynchronous record processing to finish, so an input link that triggers a read from slow hardware will return the previous value from the field

# Channel Access links

- Like database links, but these name a record that may be located in a different IOC
- Use Channel Access to communicate with the target record
  - Just like any other CA client, even for local records
  - Input links set up a CA monitor on the channel
- May include a field name (default **.VAL**)
- **PP** link flags are ignored, get/put behavior is controlled by the target CA server
  - Input links always behave like **NPP**
  - Output links follow the **PP** attribute of target field
- **MS** Link flags apply to Input links
  - Input links honor **NMS** (default)/**MS**/**MSS**/**MSI**
  - Output links are always **NMS**
- Additional flags for CA links
  - **CA** Forces a “local” link to use CA
  - **CP** On input link, process this record on every CA monitor event
  - **CPP** Like **CP** but only process me if my **SCAN** is Passive

## Link flag summary

Type	Input Links	Output Links
<b>DB</b>	PP or NPP NMS, MS, MSS or MSI	PP or NPP NMS, MS, MSS or MSI
<b>CA</b>	Always NPP MS or NMS CA forces link type CP process record on change CPP like CP but only process if SCAN=Passive	PP set by destination field Always NMS CA forces link type

Chapter 5 of the IOC Application Developer's Guide covers record links and scanning in detail, and is worth reading.

# Device Support

- The standard record-types do not access hardware directly
- The Device Support layer performs I/O operations on request from its record type
  - Each device support performs I/O for one record type
- A record's **DTYP** field determines which device support it uses
  - Most record types default to Soft Channel support if you don't set **DTYP**
- The device support selected controls the format of the link (**INP** or **OUT** field) containing the device address
- Adding new device support layers does not require making any changes to or recompilation of the record type code
- Device support often calls other software to do work for it (Driver Support or other libraries)

# Synchronous vs Asynchronous I/O

- IOC rules do not allow device support to busy-wait (i.e. hold up further record processing until the results of a slow I/O operation arrive)
  - Fast I/O should be handled synchronously
  - Slow operations must operate asynchronously
- Register-based VME/PCI cards usually give a fast response ( $<10\mu\text{s}$ ), so should be synchronous
  - A synchronous read or write device support call finishes its I/O operation before returning
- Serial, network or field-bus I/O usually takes some time ( $>10\text{ms}$ ) to return data, so should be asynchronous
  - Asynchronous device support starts an I/O operation when the record asks, flagging it as incomplete by setting **PACT** to true before returning
  - When the results are available (discovered by a CPU interrupt or polling background thread), the device support must call the record's `process()` routine to finish the record processing operations

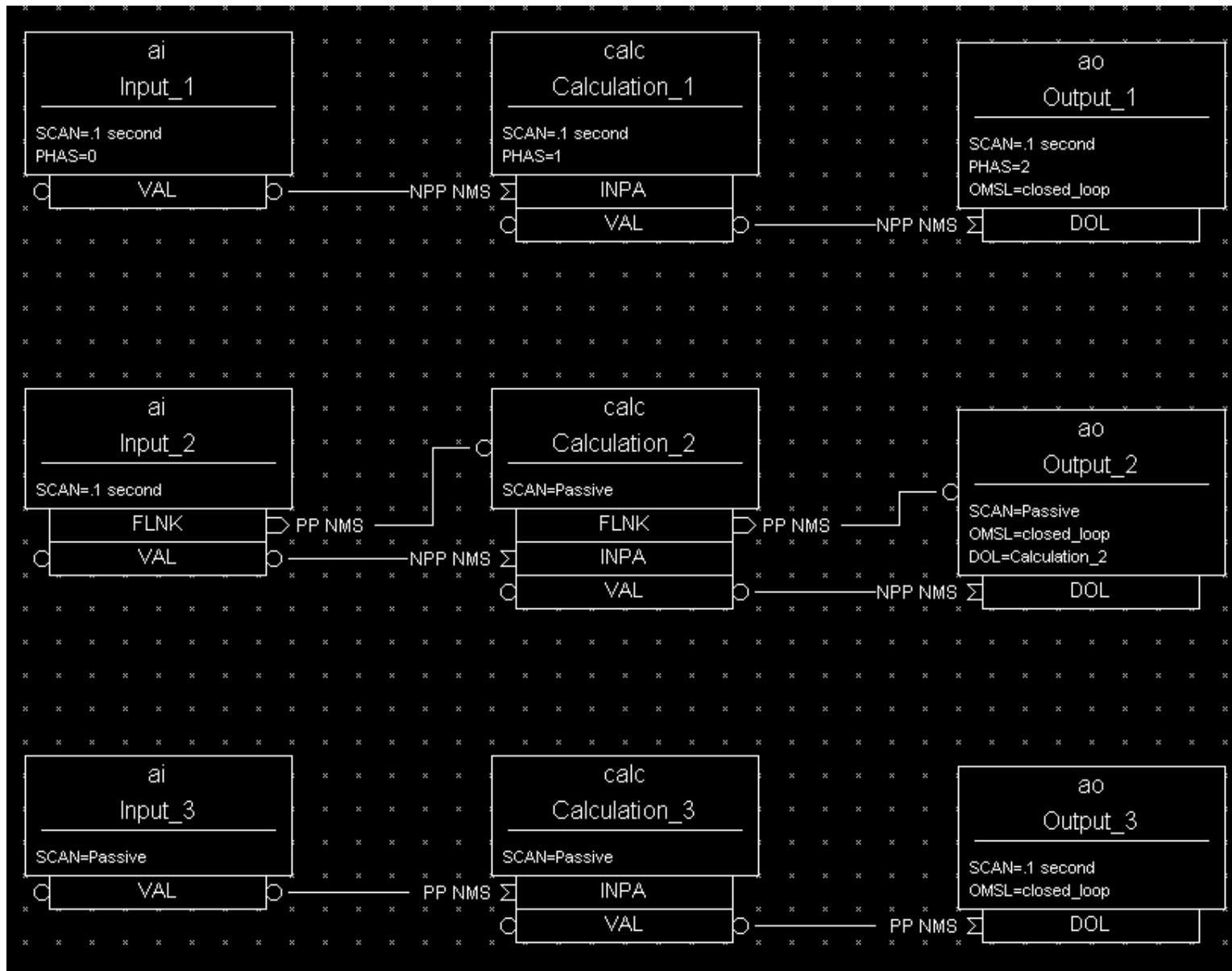
# Soft Device Support

- “Hard” input and output records do hardware I/O via device support
- “Soft” records access data from other records via DB or CA links
- 2 or 3 kinds of support are provided in recent R3.14 releases:
  - Soft Channel
    - Get/Put **VAL** through link, no units conversion performed
  - Async Soft Channel (currently output records only)
    - Put **VAL** through CA link, no conversions, wait for completion
  - Raw Soft Channel
    - Inputs
      - Get **RVAL** via input link
      - Convert **RVAL** to **VAL** (record-type specific)
    - Outputs
      - Convert **VAL** to **RVAL** (record-type specific)
      - Put **RVAL** to output link

# Forward links

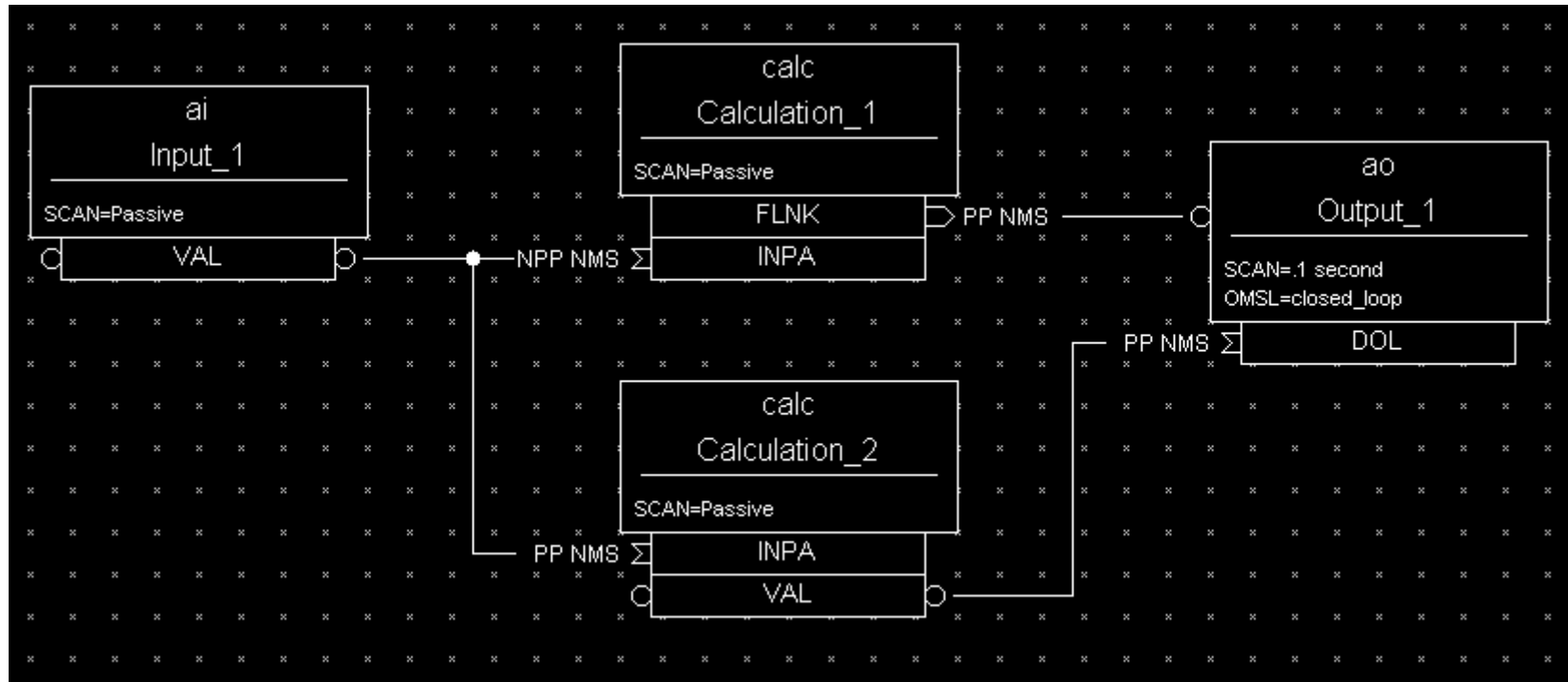
- Do not pass a value, just causes subsequent processing
- Usually a Database link, referring to a record in same IOC
  - No **PP/NPP** or **MS** flags, although VDCT includes them erroneously
  - Destination record is only processed if its **SCAN** field is **Passive**
- Forward linking via Channel Access is possible
  - Use the **CA** flag if the target record is in the same IOC
  - The link must name the **PROC** field of the target record
  - The target record need not have **SCAN** set to **Passive**

# Processing chains

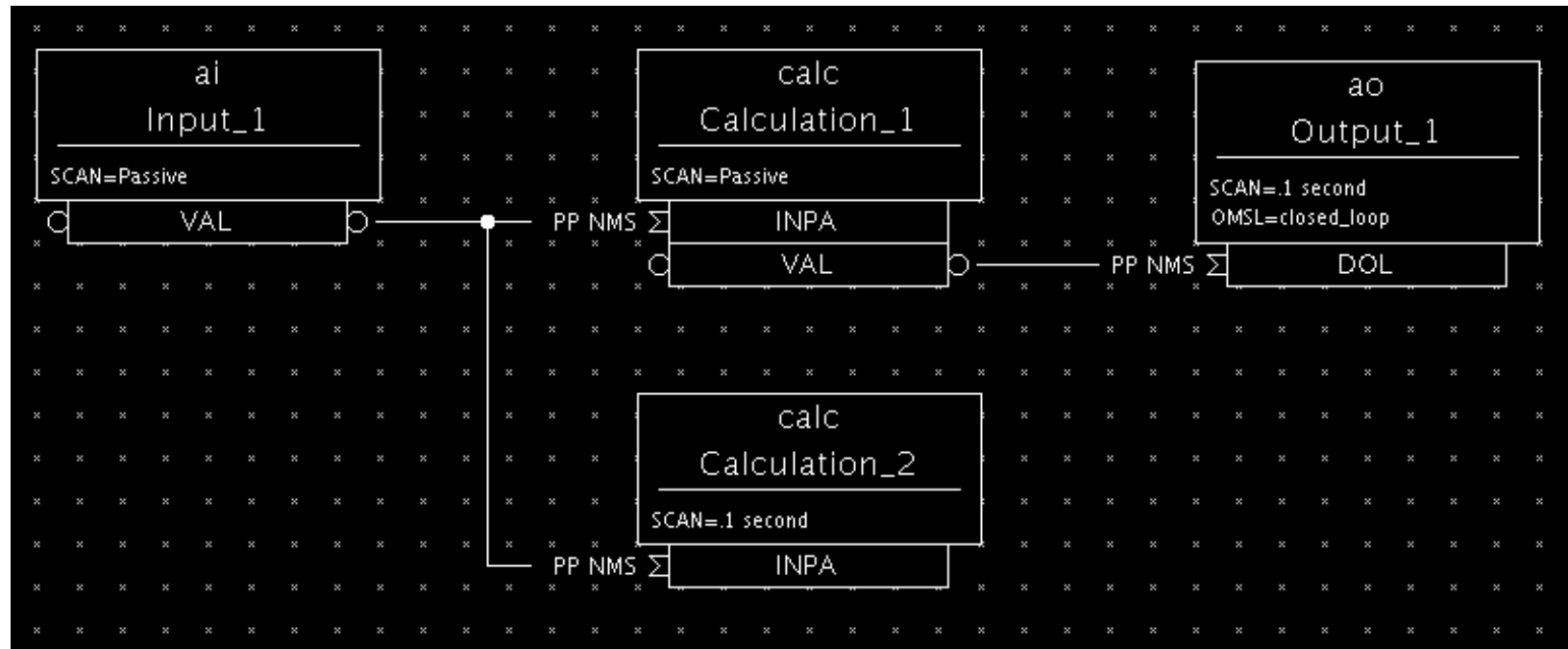




# Quiz: Which record is never processed?



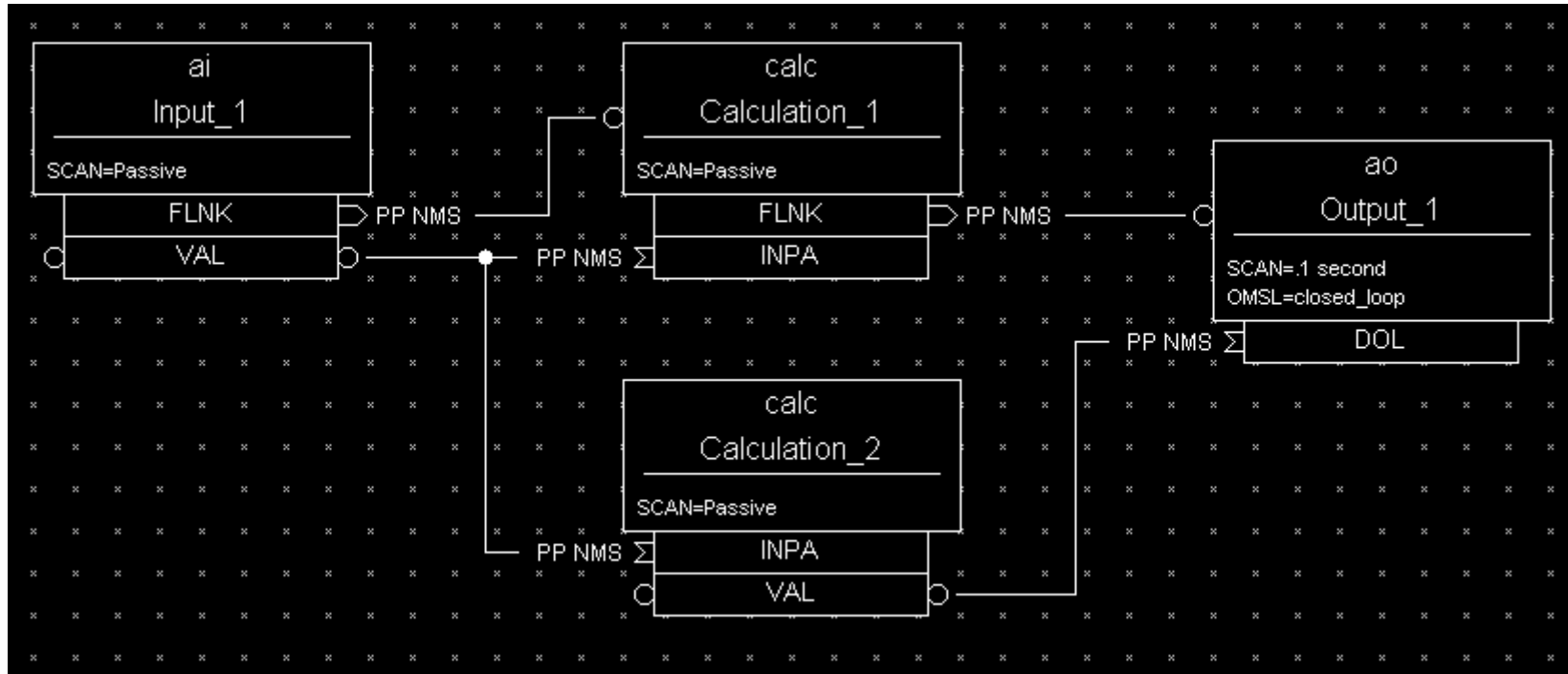
# Quiz: How often is Input\_1 processed?



# The PACT field

- All records have a boolean run-time field **PACT** (Process Active)
- **PACT** breaks loops of linked records
- It is set to true in the early stages of processing a record (but it's not the first thing that the process routine does)
  - **PACT** is true whenever a link from that record is used to get/put data or process **FLNK**
- **PACT** is reset to false after all record I/O and forward link processing have finished
  
- A **PP** link can never process a record that has **PACT** true
  - Input links just take the current value from the target field
  - Output links put their value to the target field
    - In some cases they ask the target record to reprocess itself again later
  - Forward links do nothing

# Quiz: What happens here?



# Preventing records from processing

- It is useful to be able to stop an individual record from processing on some condition
- Before record-specific processing is called, a value is read through the **SDIS** input link into **DISA** (0 if the link is not set)
- If **DISA=DISV**, the record will *not* be processed
- The default value of the **DISV** field is 1
- A disabled record can raise an alarm by putting the desired severity in the **DISS** field
- The **FLNK** of a disabled record is never triggered

Break time...

5 Minute break

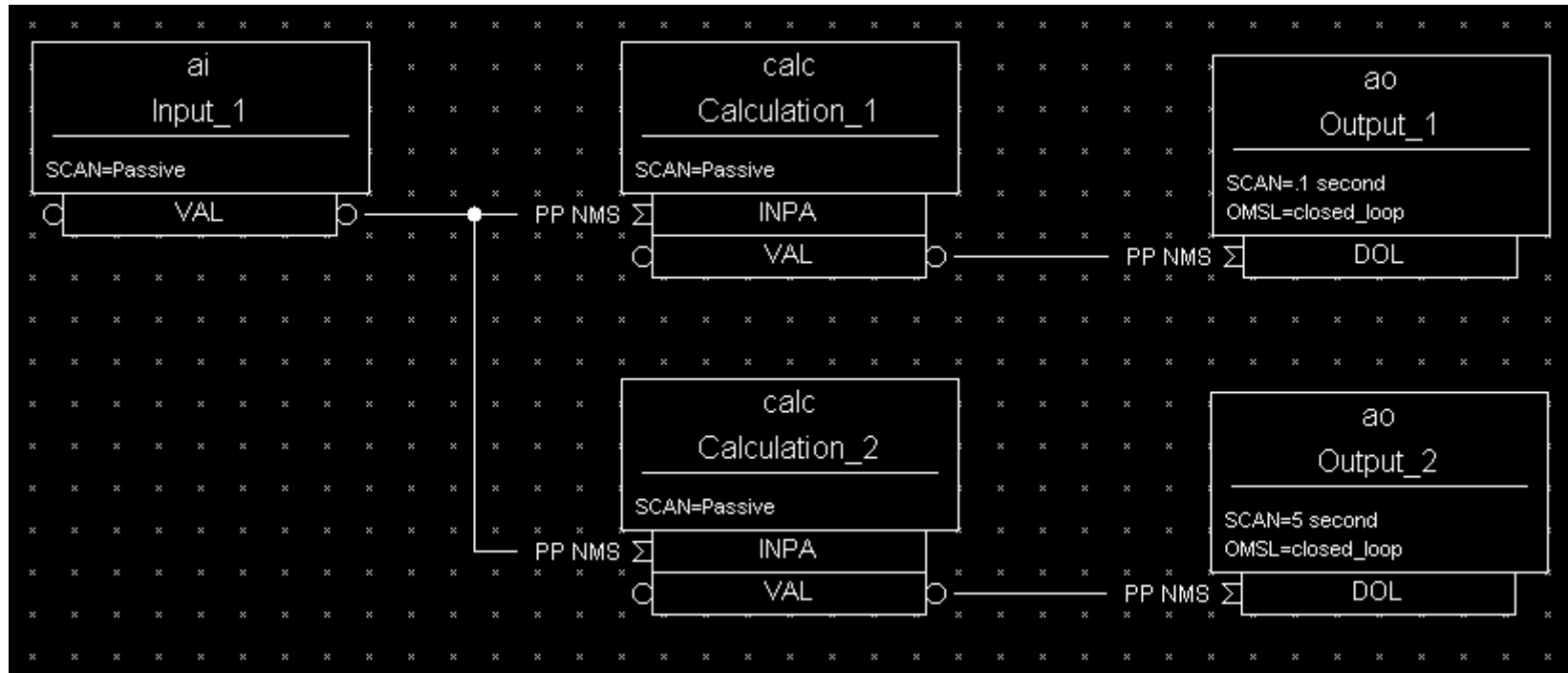


# How are records given CPU time?

Many different IOC threads are used to process records:

- *scanperiod* — Periodic scans
  - Each scan rate gets its own thread
  - Faster scans at higher thread priority (if supported by the Operating System)
- *cbLow*, *cbMedium*, *dbHigh* — Callback facility
  - One thread for each scheduling priority (PRIO field)
  - Used by device support, I/O Interrupts etc.
- *scanOnce*
  - IOC internal use for record reprocessing
- *CAS-client* — CA client-initiated processing
  - One thread for each CA client connected to the server
- Channel Access threads use lower priority than record processing
  - If a CPU spends all its time doing I/O and record processing, you may be unable to control or monitor the IOC via the network

# Quiz: What could go wrong here?





# Lock-sets

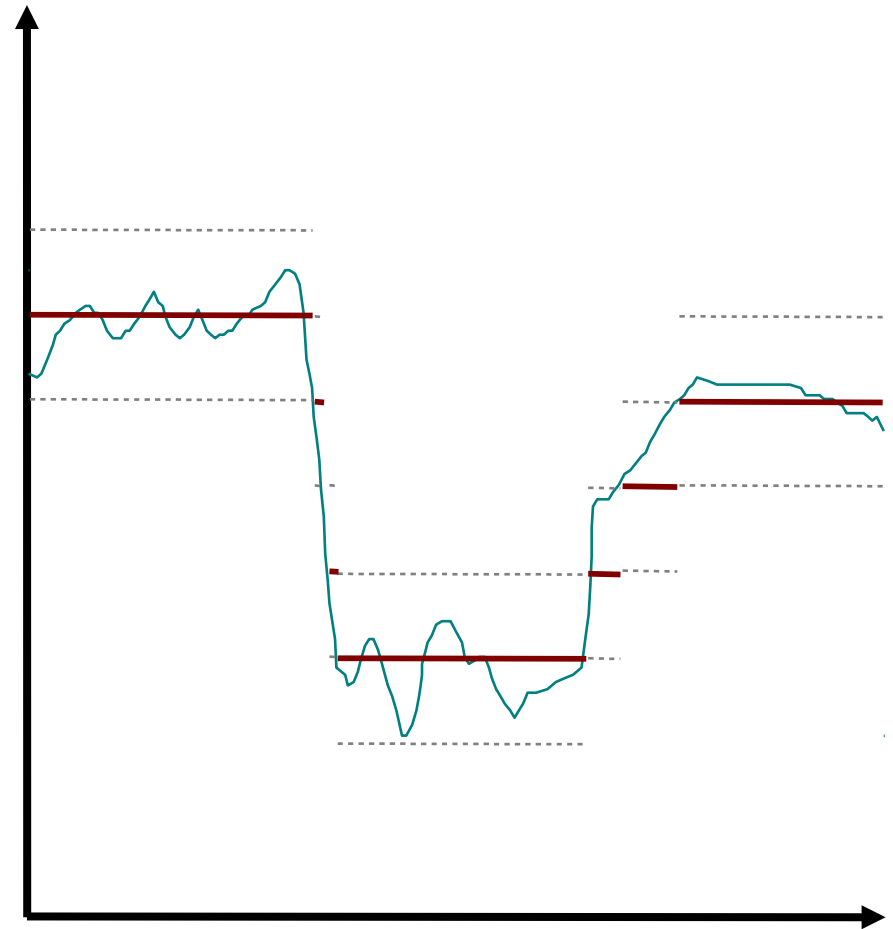
- Prevent records from being processed simultaneously by two different scan tasks
  - **PACT** can't do that, it isn't set early enough and it's not a Mutex
- A lock-set is a group of records interconnected by database links
- Lock-sets are determined automatically by the IOC at start-up, and recalculated whenever a database link is added, deleted or modified
- You can split two linked records into different lock sets by making the link(s) joining them into Channel Access ones, using the **CA** flag
  - Remember that CA links behave slightly differently than DB links, make sure your design still works!

# Alarms

- Every record has the fields
  - **SEVR** Alarm Severity
    - NONE, MINOR, MAJOR, INVALID
  - **STAT** Alarm Status (reason)
    - READ, WRITE, UDF, HIGH, LOW, STATE, COS, CALC, DISABLE, etc.
- Most numeric records compare **VAL** against the **HIHI**, **HIGH**, **LOW** and **LOLO** fields after its value has been determined
- The **HYST** field sets a hysteresis to prevent alarm chattering
- A separate alarm severity can be set for each numeric limit exceeded
  - Fields **HHSV**, **HSV**, **LSV**, and **LLSV**
- Discrete (binary) records can raise alarms on entering a particular state, or on a change of state (COS)

# Change Notification: Monitor Dead-bands

- Channel Access notifies clients that are monitoring a numeric record when
  - **VAL** changes by more than the value in field:
    - MDEL** *Value monitors*
    - ADEL** *Archive monitors*
  - Record's Alarm Status changes
    - HYST** *Alarm hysteresis*
- The Analogue Input record has a smoothing filter to reduce noise on the input signal (**SMOO**)

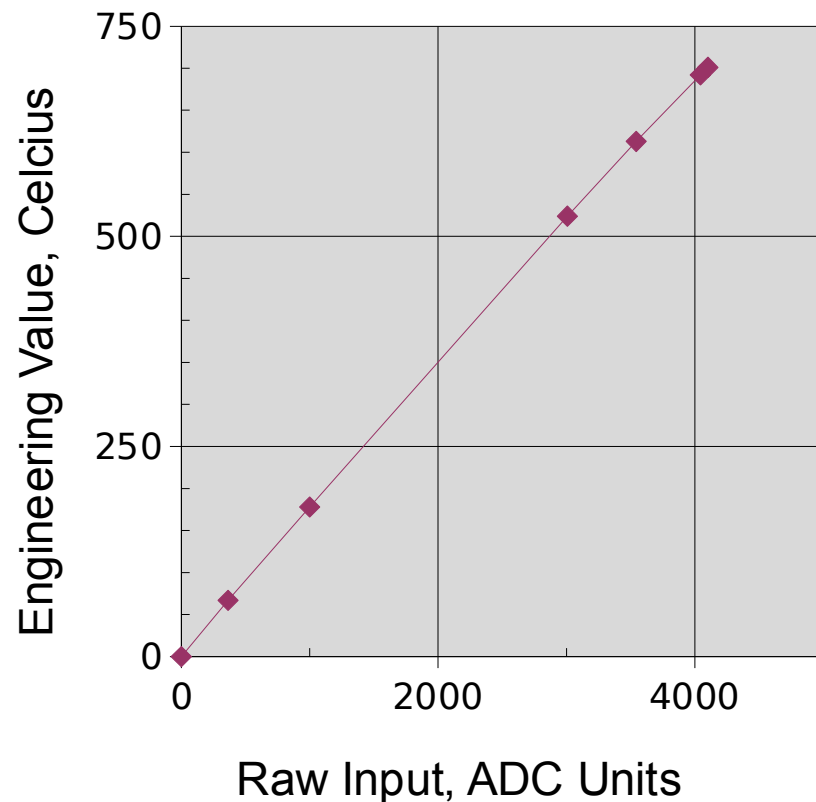


# Breakpoint Tables

- Analogue Input and Output records can do non-linear conversions from/to the raw hardware value
- Breakpoint tables interpolate values from a given table
- To use, set the record's **LINR** field to the name of the breakpoint table you want to use
- Example breakpoint table (in some loaded .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



# Simulation

- Input and output record types often allow simulation of hardware interfaces
  - SIML** Simulation mode link
  - SIMM** Simulation mode value
  - SIOL** Simulation input link
  - SVAL** Simulated value
  - SIMS** Simulation alarm severity
- Before calling device support, records read **SIMM** through the **SIML** link
- If **SIMM=YES** (1) or **SIMM=RAW** (2) the device support is not used; record I/O is done through the **SIOL** link and **SVAL** field instead
- An alarm severity can be set whenever simulating, given by **SIMS** field

# Access Security

- A networked control system must have the ability to enforce security rules
  - Who can do what from where, and when?
- In EPICS, security is enforced by the CA server (the IOC or gateway)
- A record is placed in the Access Security Group named in its ASG field
  - DEFAULT is used if no group name is given
- Rules are specified for each group to determine whether a CA client can read or write to records in that group, based on
  - Client user ID
  - Client host-name or IP address
  - Access Security Level of the field addressed
  - Values read from the database

# Access Security Configuration File

- Security rules are loaded from an Access Security Configuration File, for example:

```
UAG(users) {anj, kuk, wen}
HAG(hosts) {epics.uspas.fnal.gov, kay, eric}
ASG(DEFAULT) {
    RULE(1, READ)
    RULE(1, WRITE) {
        UAG(users)
        HAG(hosts)
    }
}
```

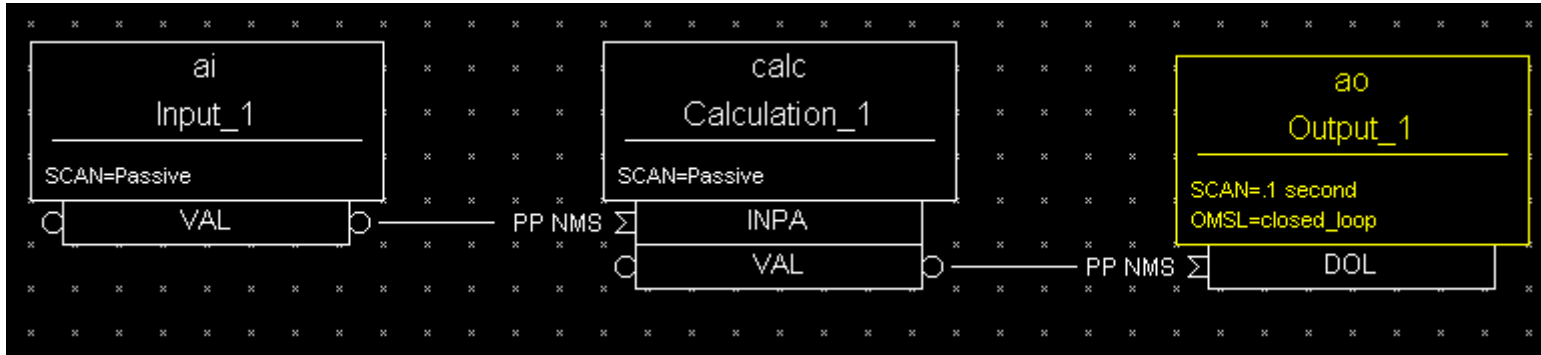
- If no security file is loaded, all Security checks will be turned off
- More details and the security rules file syntax can be found in Chapter 8 of the IOC Application Developers Guide

# Optional Slides

## Detailed Order of Operations

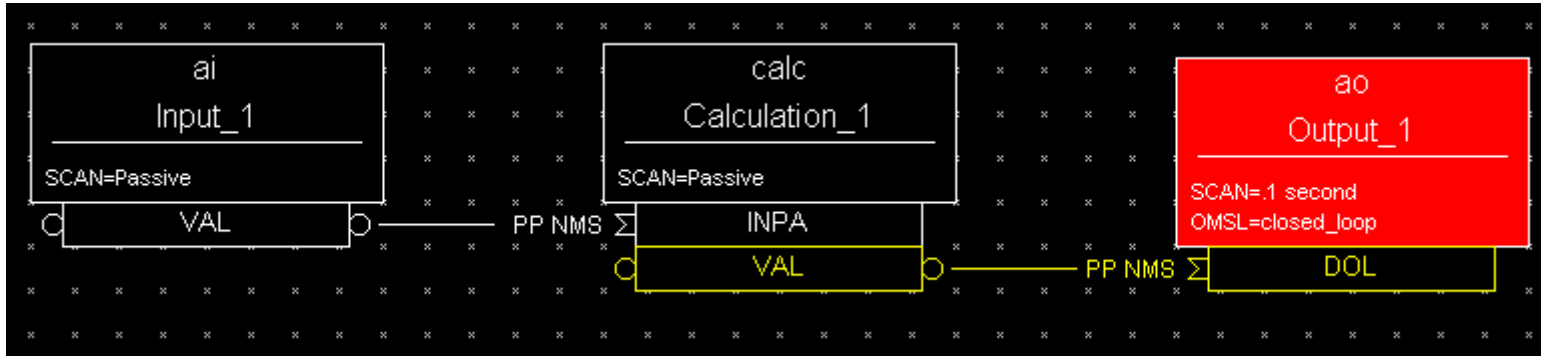


# Order of Operations (Synchronous I/O)



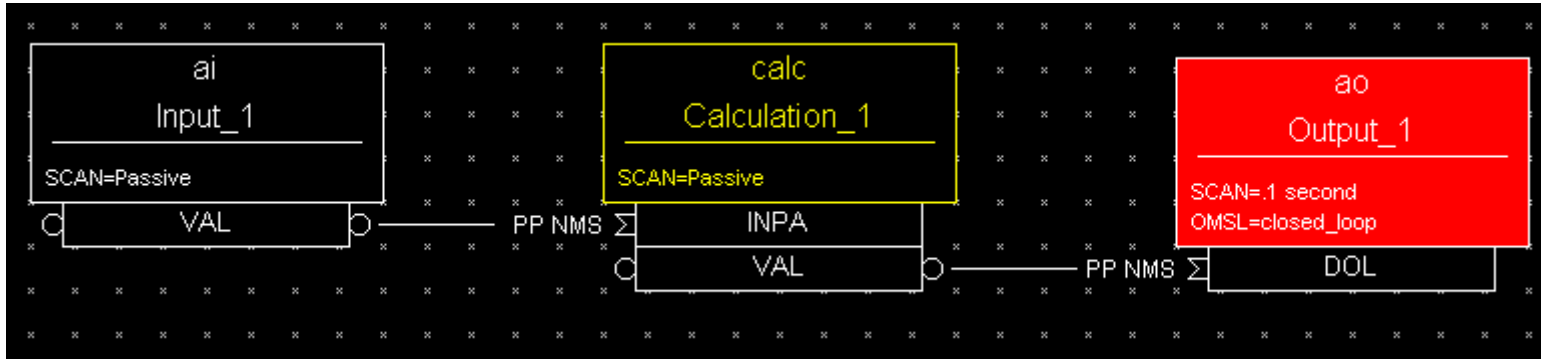
1. Every 0.1 seconds, iocCore will attempt to process the Output\_1 record
2. The Output\_1.PACT field is currently False, so the record is quiescent and can be processed
3. If set, the Output\_1.SDIS link would be read into Output\_1.DISA
4. Since DISA DISV, the ao record type's process() routine is called

# Order of Operations (Synchronous I/O)



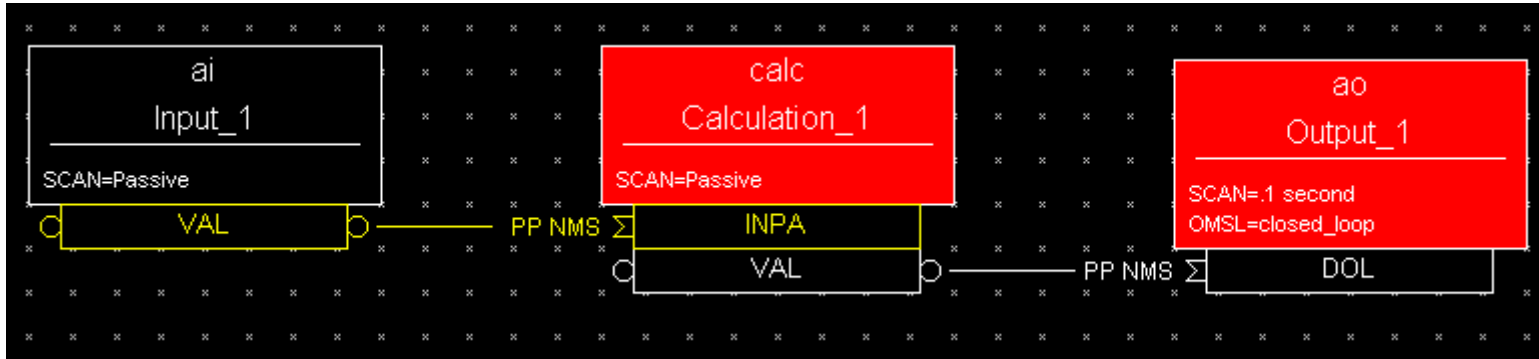
5. The ao's process() routine checks the `Output_1.OMSL` field; it is `closed_loop`, so
6. It sets `Output_1.PACT` to `True`, then
7. Reads a value through the `Output_1.DOL` link
8. The `Output_1.DOL` link contains `Calculation_1.VAL PP` so this first attempts to process the `Calculation_1` record

# Order of Operations (Synchronous I/O)



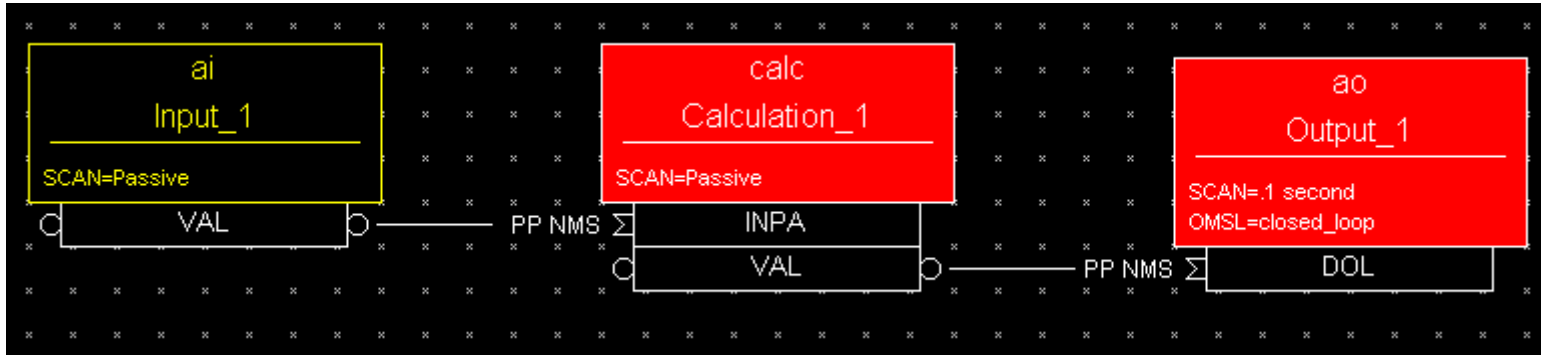
9. The `Calculation_1.SCAN` field is `Passive` and `Calculation_1.PACT` is `False`, so processing is possible
10. If set, the `Calculation_1.SDIS` link would be read into `DISA`
11. Since `DISA DISV`, the `calc` record type's `process()` routine is called

# Order of Operations (Synchronous I/O)



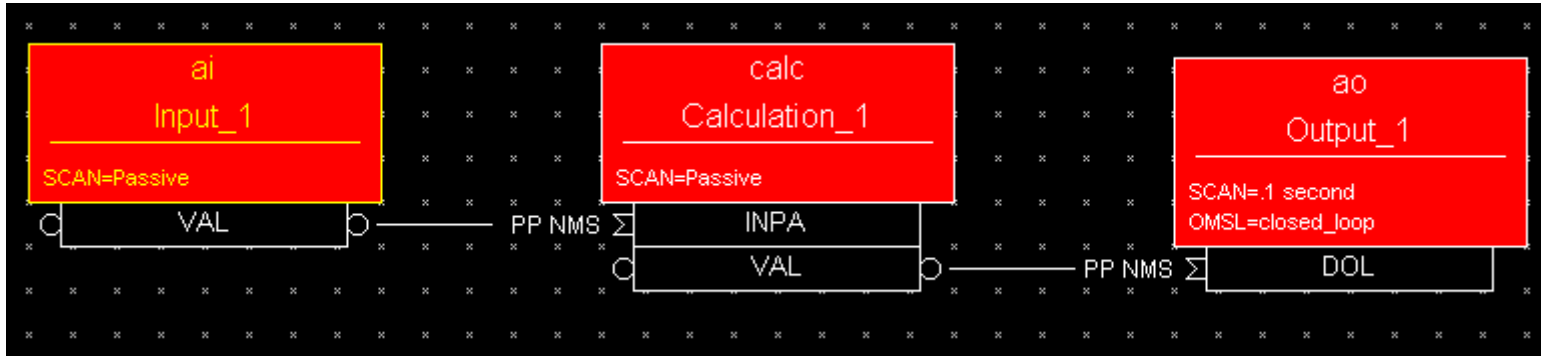
12. The calc's process() routine sets `Calculation_1.PACT` to True, then
13. Starts a loop to read values from the links `INPA` through `INPL`
14. The `Calculation_1.INPA` link is set to `Input_1.VAL PP` so this first attempts to process the `Input_1` record

# Order of Operations (Synchronous I/O)



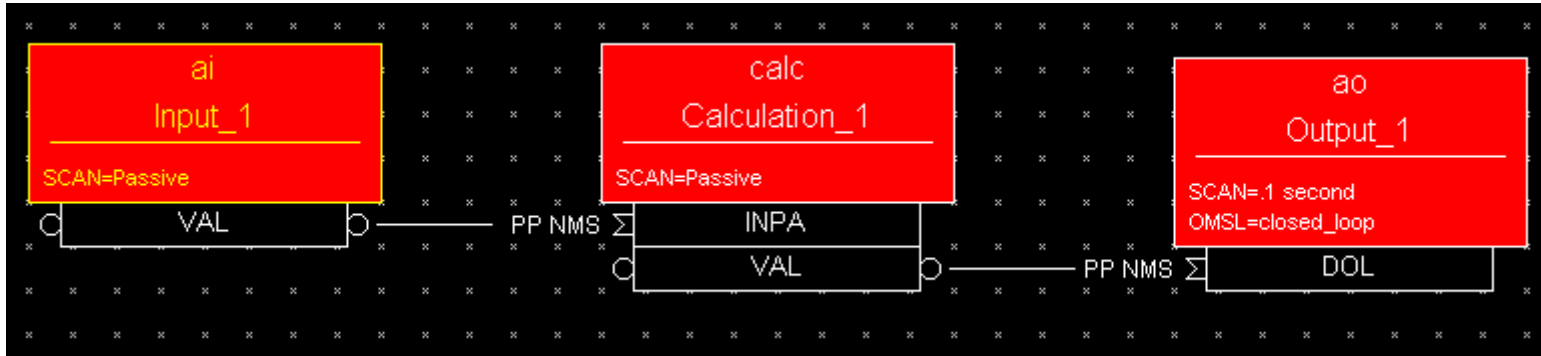
15. The `Input_1 . SCAN` field is `Passive` and `Input_1 . PACT` is `False`, so processing is possible
16. If set, the `Input_1 . SDIS` link is read into the `Input_1 . DISA` field
17. Since `DISA DISV`, the `ai` record type's `process()` routine is called
18. The `ai`'s `process()` routine calls the associated device support to read a value from the hardware it's attached to

# Order of Operations (Synchronous I/O)



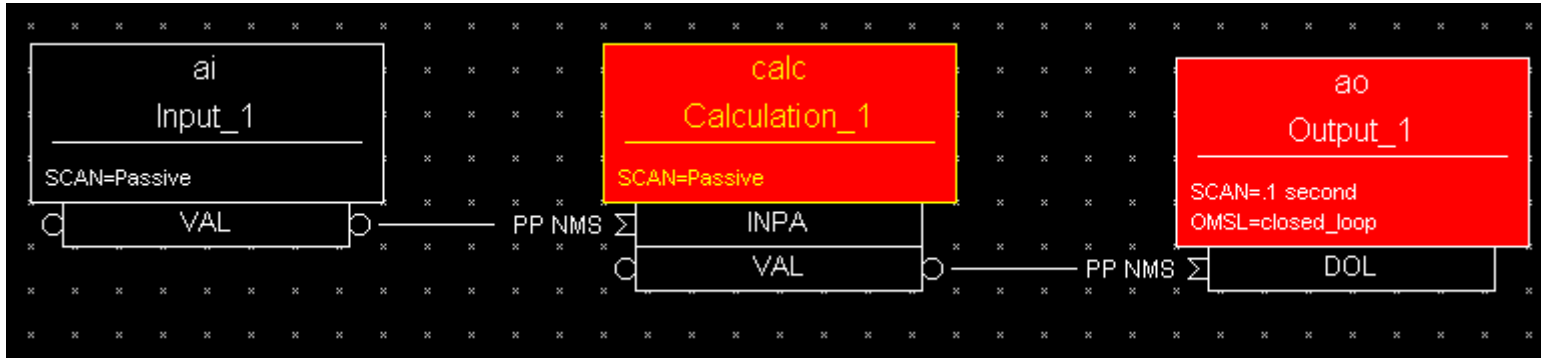
19. The device support is synchronous, so it puts the hardware input value into the `Input_1.RVAL` field and returns to the `ai` record's `process()` code
20. The `Input_1.PACT` field is set to `True`
21. The record's timestamp field `Input_1.TIME` is set to the current time
22. The raw value in `Input_1.RVAL` is converted to engineering units, smoothed, and the result put into the `Input_1.VAL` field

# Order of Operations (Synchronous I/O)



23. The `Input_1.VAL` is checked against alarm limits and monitor dead-bands, and appropriate actions are taken if these are exceeded
24. If the Forward Link field `Input_1.FLNK` is set, an attempt is made to process the record it points to
25. The `Input_1.PACT` field is set to False, and the `process()` routine returns control to the `Calculation_1` record

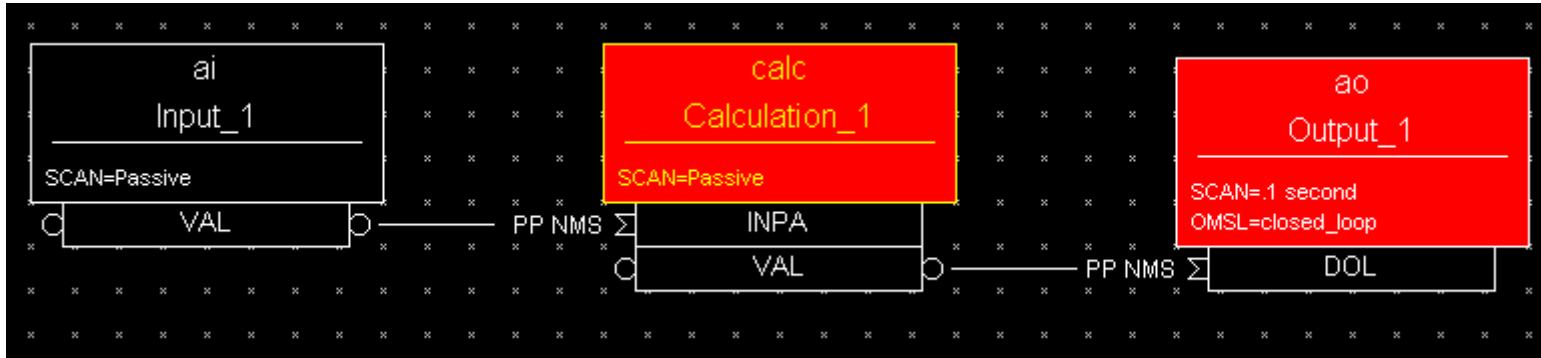
# Order of Operations (Synchronous I/O)



26. The value read through the `Calculation_1.INPA` link is copied into the `Calculation_1.A` field
27. The Calculation record type's `process()` routine continues to loop, reading its input links
28. In this example only the `INPA` link is set, so the routine finishes the loop and evaluates the `Calculation_1.CALC` expression (not shown)
29. The result of the expression is put in the `Calculation_1.VAL` field

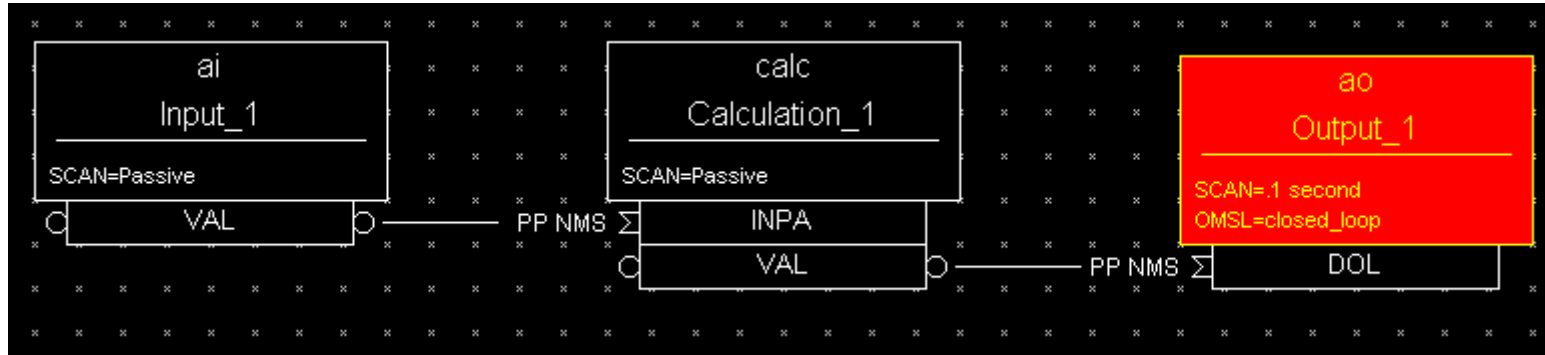


# Order of Operations (Synchronous I/O)



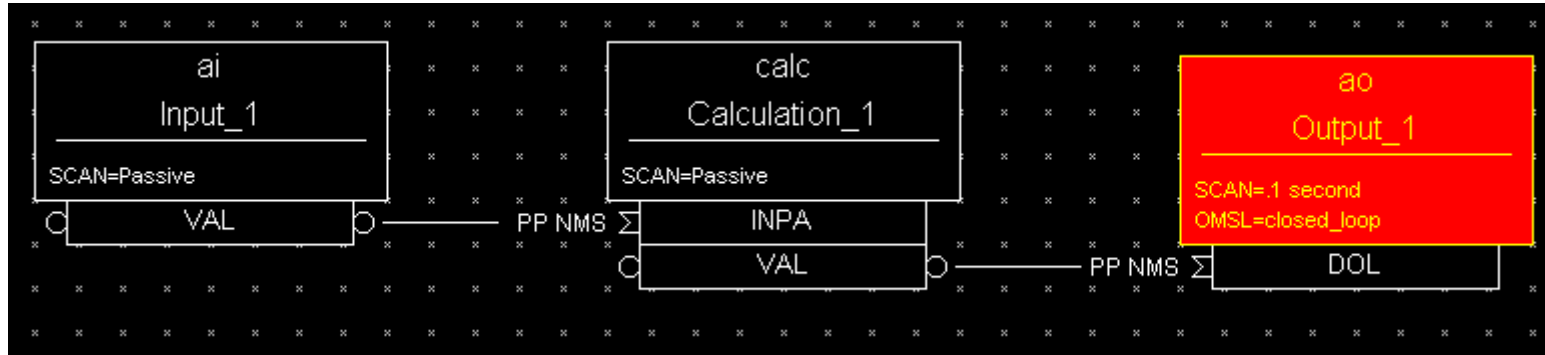
30. The record's timestamp field `Calculation_1.TIME` is set to the current time
31. `Calculation_1.VAL` is checked against alarm limits and monitor dead-bands, and appropriate action is taken if these are exceeded
32. If the Forward Link field `Calculation_1.FLNK` is set, an attempt is made to process the record it points to
33. The `Calculation_1.PACT` field is set to False, and the `process()` routine returns control to the `Output_1` record

# Order of Operations (Synchronous I/O)



34. The value read through the `Output_1.DOL` link would now be forced into the range `DRVL..DRVH` if those fields were set, but they aren't so it's copied to the `Output_1.VAL` field unchanged
35. The `Output_1.VAL` value is converted from engineering to raw units and placed in `Output_1.RVAL`
36. `Output_1.VAL` is checked against alarm limits and monitor dead-bands, and appropriate action is taken if these are exceeded
37. The associated device support is called to write the value to the hardware

# Order of Operations (Synchronous I/O)



- 38. The device support is synchronous, so it outputs the value to the attached hardware and returns
- 39. The record's timestamp field `Output_1.TIME` is set to the current time
- 40. If the Forward Link field `Output_1.FLNK` is set, an attempt is made to process the record it points to
- 41. The `Output_1.PACT` field is set to False, and the `process()` routine returns