

EPICS Process Database

Michael Davidsaver

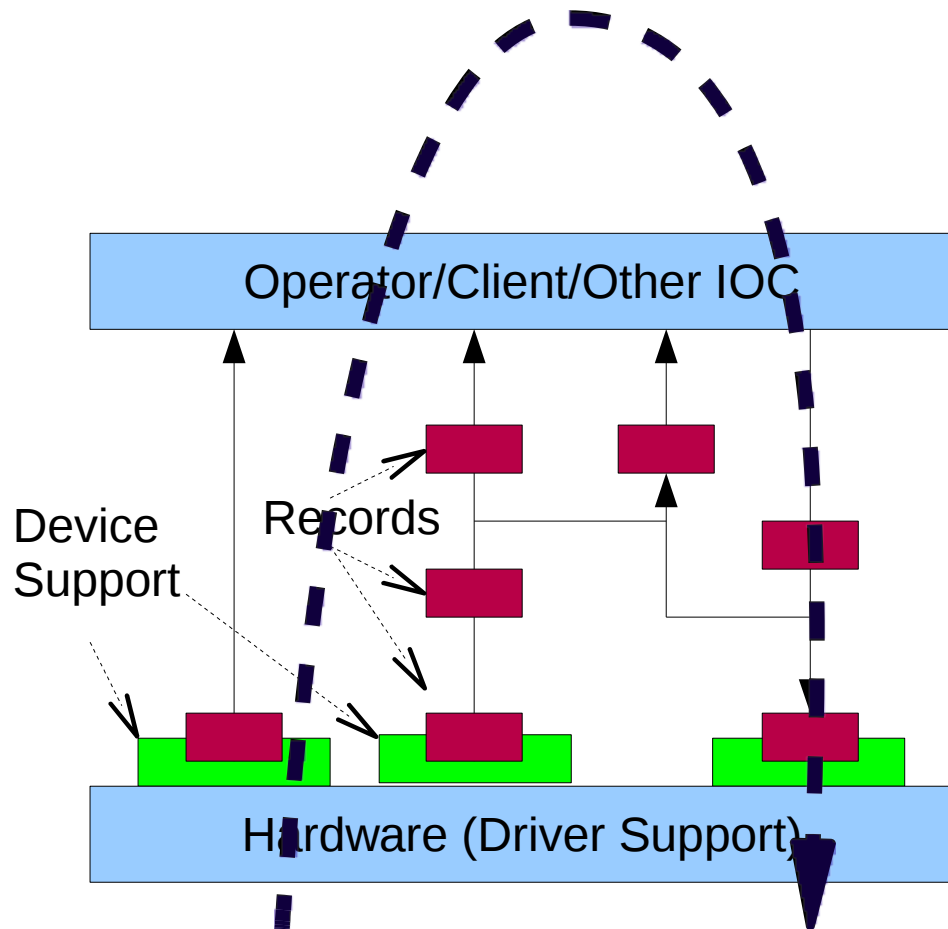
mdavidsaver@bnl.gov

Research Instruments Site Training Fall 2010

Goals

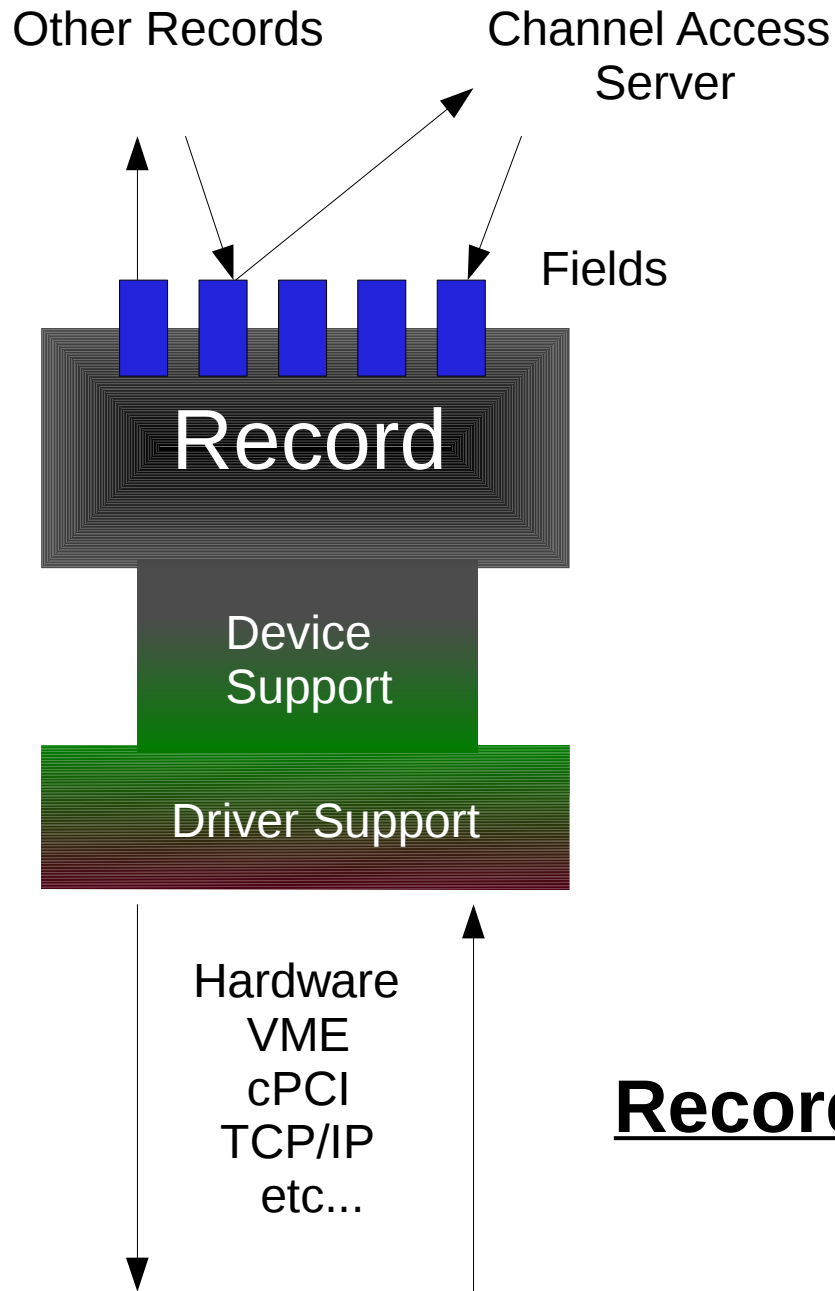
- Understand .db file and IOC shell syntax
- Be able to follow the flow of a database
- Add/modify an alarm condition
- Add a calcout record to modify a value
- Examples
 - <http://epics.nsls2.bnl.gov/training/training-201011.tar.gz>

IOC Data Pipeline



- Configuration vs. Compilation
- Modular
- Database
 - Data (Configuration)
- Device Support
 - Operations
 - Modify DB
 - Access HW
 - Stateless (ideally)
- Driver Support
 - Catch-all

In the Database



- Record
 - Collection of fields
 - Recordtype
 - Many have devSup
- Fields
 - Atomic value (int, ...)
 - Scalar, array, or link
 - Attributes (eg. PP)

Record+Field=Process Variable

In the Database

```
record(ai, "myrecord:v17") {  
    field(DESC, "neat")  
    field(DTYP, "mydev")  
    field(INP, "#C1 S7 @v")  
}
```

- Record
 - Collection of fields
 - Recordtype
 - Many have devSup
- Fields
 - Atomic value (int, ...)
 - Scalar, array, or link
 - Attributes (eg. PP)

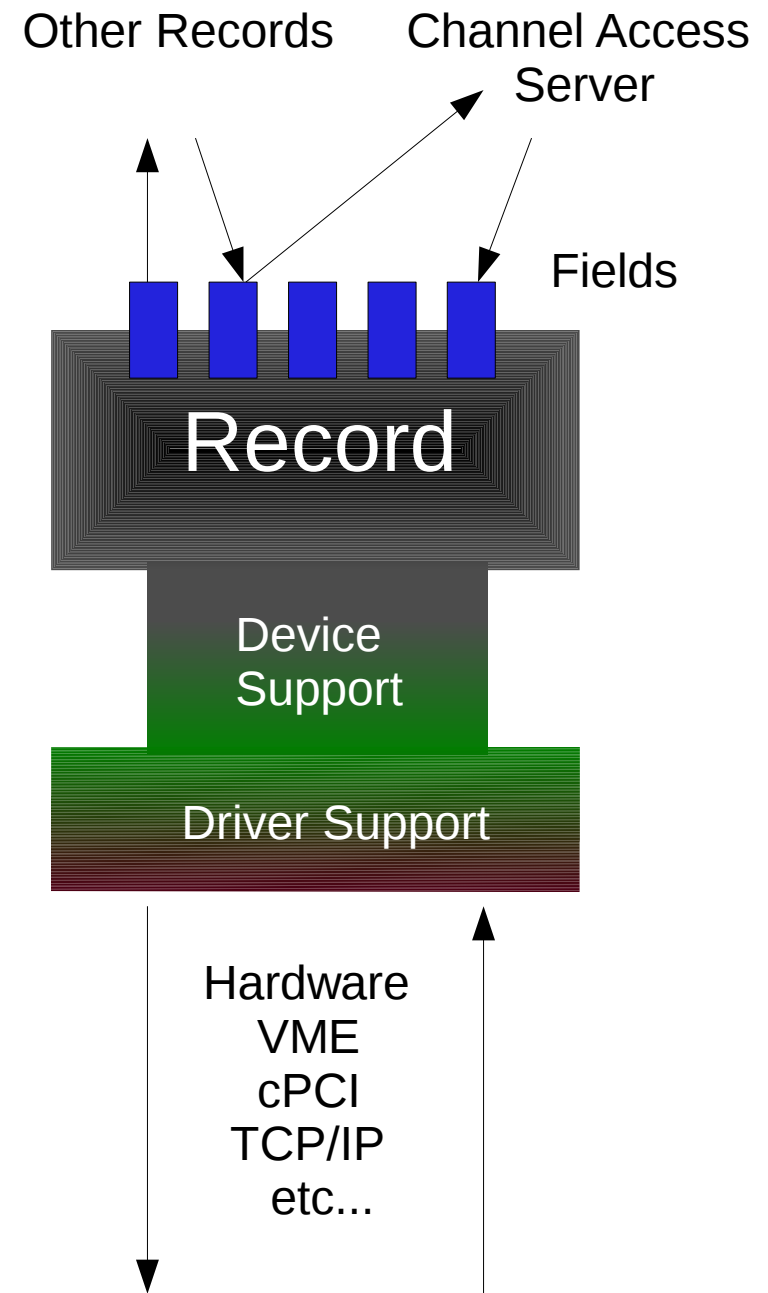
Record+Field=Process Variable

RDB Translation

- Process DB is an object database with an integrated client (device support).
- In terms of a relational database
- recordtype → table
- record → row
- field → column
- link → foreign key (???)
 - FK is a cell (column+row) pointing to a row.
 - A Link is a cell pointing to another cell.

What is a Record?

- A record is the basic unit of “action” in a database
- Field values modify the action
- Records do
- Fields store



Device Support

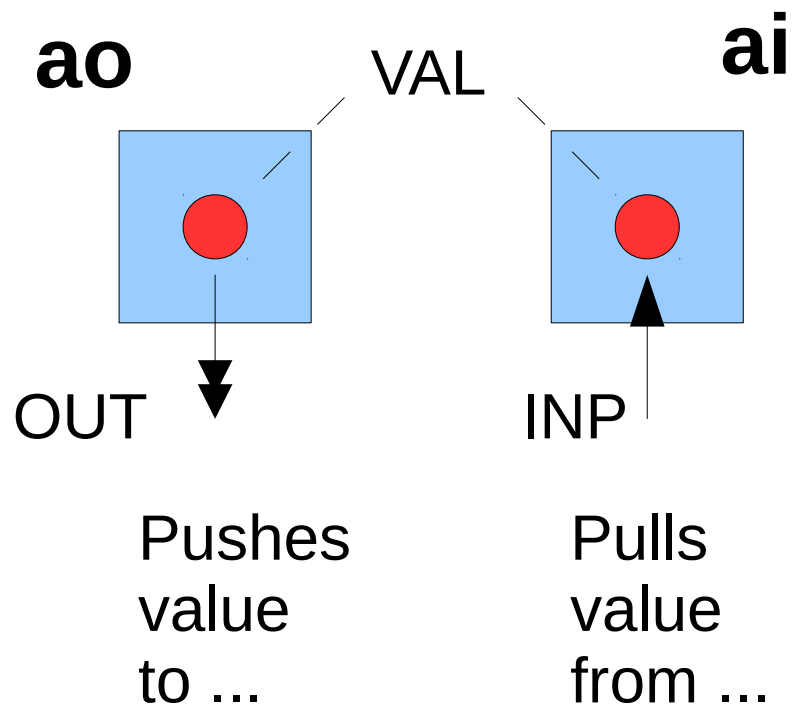
- Compiled (C/C++)
- Associated with a record
- Use pre-defined set of hooks
- API
 - report()
 - init()
 - init_record()
 - get_iointr_info()
 - process()
 - read()
 - write()

Db Syntax

- Recordtype
- Instance name
 - \$(P) macros expanded when loaded
 - Load same file several times
- Field name
- Field value
 - Always quote

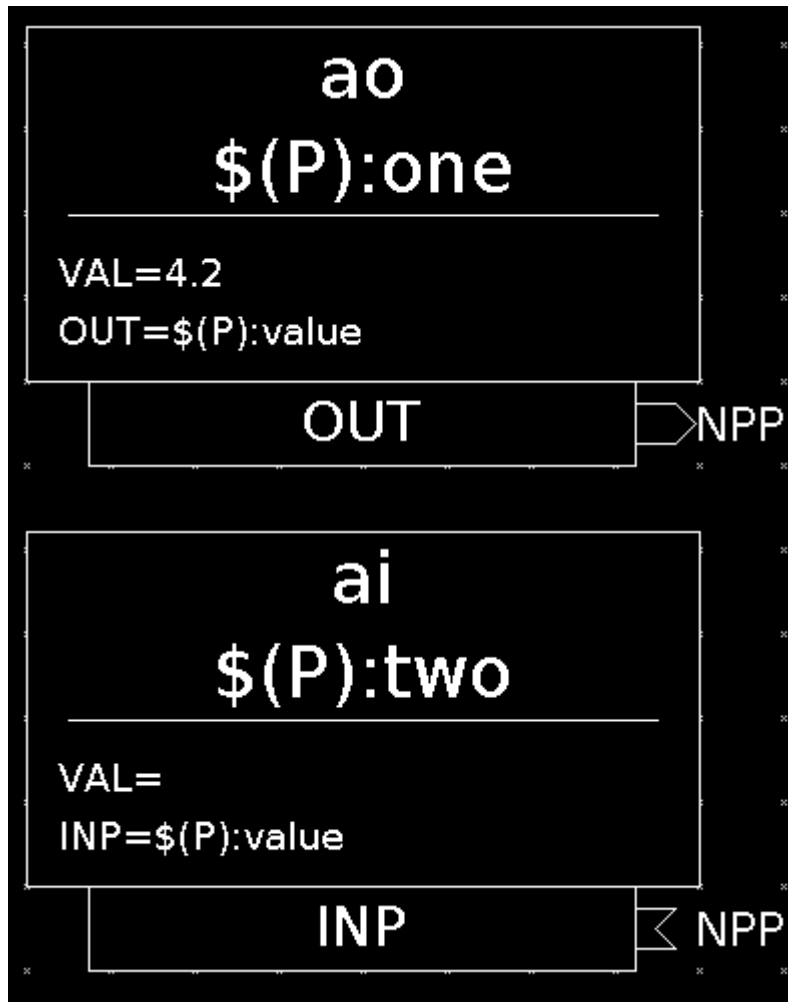
```
record(ao, "$(P)name") {  
    field(DTYP, "mydev")  
    field(OUT, "#C$(C) S0 @")  
    field(LINR, "LINEAR")  
    field(ESLO, "0.1")  
}
```

Simple recordtypes



- Analog (float)
 - ao, ai
- Binary
 - bo, bi, mbbo, mbbi
- Integer (32-bit)
 - longout, longin

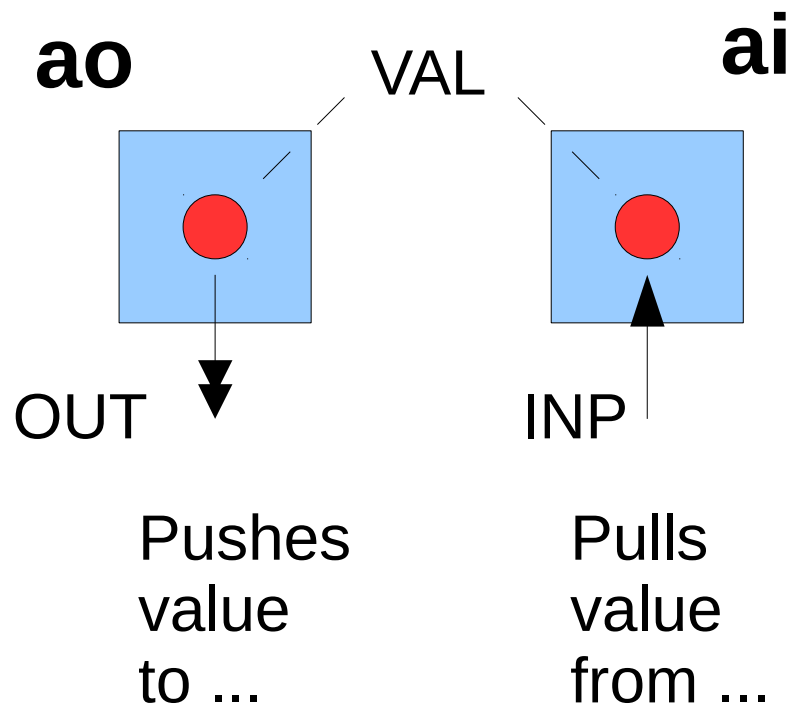
Simple recordtypes



- Analog (float)
 - ao, ai
- Binary
 - bo, bi, mbbo, mbbi
- Integer (32-bit)
 - longout, longin

↖ In VDCT

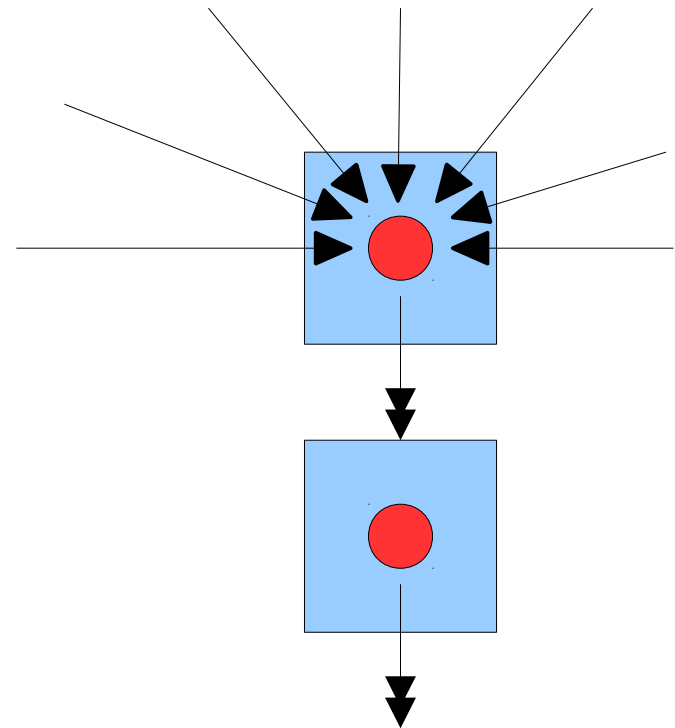
Simple recordtypes (2)



- Input
 - `VAL=read(INP)`
 - Operator reads `VAL`
- Output
 - Operator sets `VAL`
 - `write(OUT,VAL)`
- Address (`INP/OUT`)

Calculator Record

- recordtype: calcout
- 12 Input links (A-L)
- 1 Output link (OUT)
- Result stored in VAL
- CALC field
 - $(A-B)/(A+B)$
 - Length limit (40 char)
- Changeable at runtime
- OUT, OCALC, OVAL will be explained later



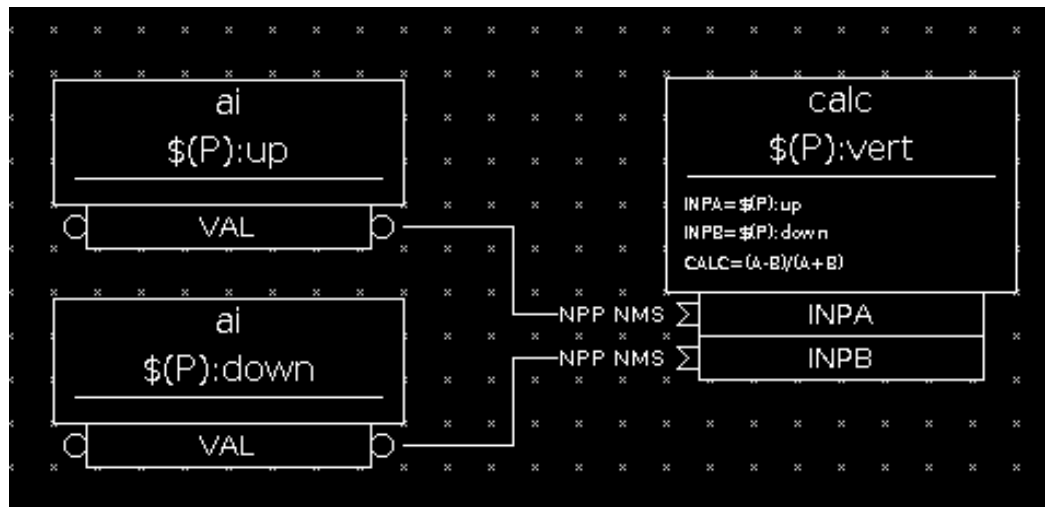
Calculator Record

- recordtype: calcout
- 12 Input links (A-L)
- 1 Output link (OUT)
- Result stored in VAL
- CALC field
 - $(A-B)/(A+B)$
 - Length limit (40 char)
- Changeable at runtime
- OUT, OCALC, OVAL will be explained later

```
record(ai, "bpm:up") {  
}  
record(ai, "bpm:down") {  
}  
record(calcout, "bpm:y") {  
    field(INPA, "bpm:up")  
    field(INPB, "bpm:down")  
    field(CALC, "(A-B)/(A+B)")  
}
```

Full list of CALC expression syntax
in record reference manual

Calculator Record



```
record(ai, "bpm:up") {  
}
```

```
record(ai, "bpm:down") {  
}
```

```
record(calcout, "bpm:y") {  
    field(INPA, "bpm:up")  
    field(INPB, "bpm:down")  
    field(CALC, "(A-B)/(A+B)")  
}
```

Full list of CALC expression syntax
in record reference manual

Exercise 1 – softloc Basics

- Executable: softloc
- Part of EPICS Base
- No device support

Useful utilities from EPICS Base

caget, camonitor, caput, cainfo

The Counter

- Start script: count.cmd

```
dbLoadRecords("count.db", "P=myprefix")
```

```
iocInit()
```



Select your own

- Simple 1 Hz counter

```
softloc count.cmd
```

- Database: count.db

```
record(calcout, "$(P):count") {  
    field(SCAN, "1 second")  
    field(VAL, "7")  
    field(INPA, "$(P):count.VAL NPP")  
    field(INPB, "2")  
    field(CALC, "A+B")  
}
```

- Things to try

```
camonitor myprefix:count
```

```
caput myprefix:count.SCAN
```

```
    ".1 second"
```

```
caput myprefix:count.B -2
```

Full list of CALC expression syntax
in record reference manual

Extending Counter

- Things to try
- Make the counter roll over

Use “cond ? true : false”

Set limits

field(HOPR, “10”)

field(LOPR, “0”)

- Examine CA meta-data
- `caget -d GR_DOUBLE myprefix:count`
- `caget -d GR_ENUM myprefix:count.SCAN`
- `caget -a myprefix:count`

10 Minute Break

Database Scanning

- Cause processing
- When actions happen
 - Inputs read
 - Outputs written
 - Value conversion
 - Alarms checked
- Defined per record
 - SCAN field
- Conditions
 - Periodic timer
 - 1 second, 10 second, ...
 - HW interrupt
 - I/O Intr
 - Passive
 - More in a moment

Scan Examples

```
record(bi, "status..") {  
    field(SCAN, "1 second")  
    field(INP, "some:other")  
}
```

```
record(ao, "gain..") {  
    field(SCAN, "Passive")  
    field(DTYP, "mydac")  
    field(INP, "#C1 S0 @gain")  
}
```

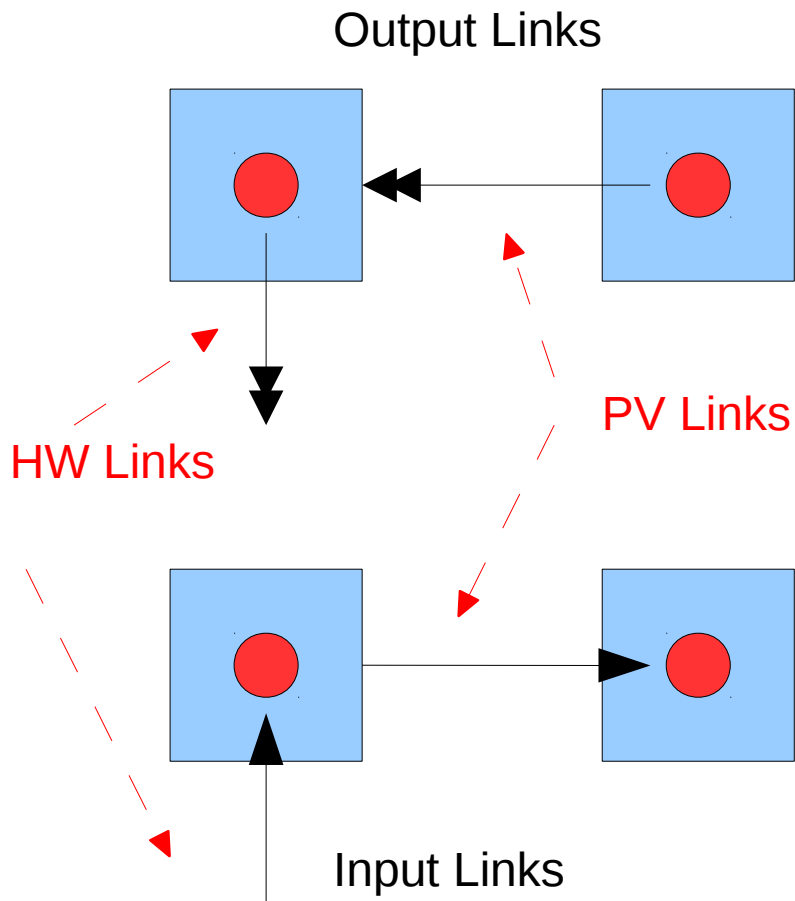
```
record(ai, "level..") {  
    field(SCAN, "I/O Intr")  
    field(DTYP, "myadc")  
    field(INP, "#C1 S0 @adc")  
}
```

Device Support must
know which "interrupt"
source to use

Passive Scan

- Actions which can cause a Passive record to process
 - A 'Get'
 - A 'Put'
 - A forward link (more in moment)
- Things which can cause...
 - A DB link from another record
 - A CA client (another IOC, caget, EDM, BOY, ...)
 - The difference: None

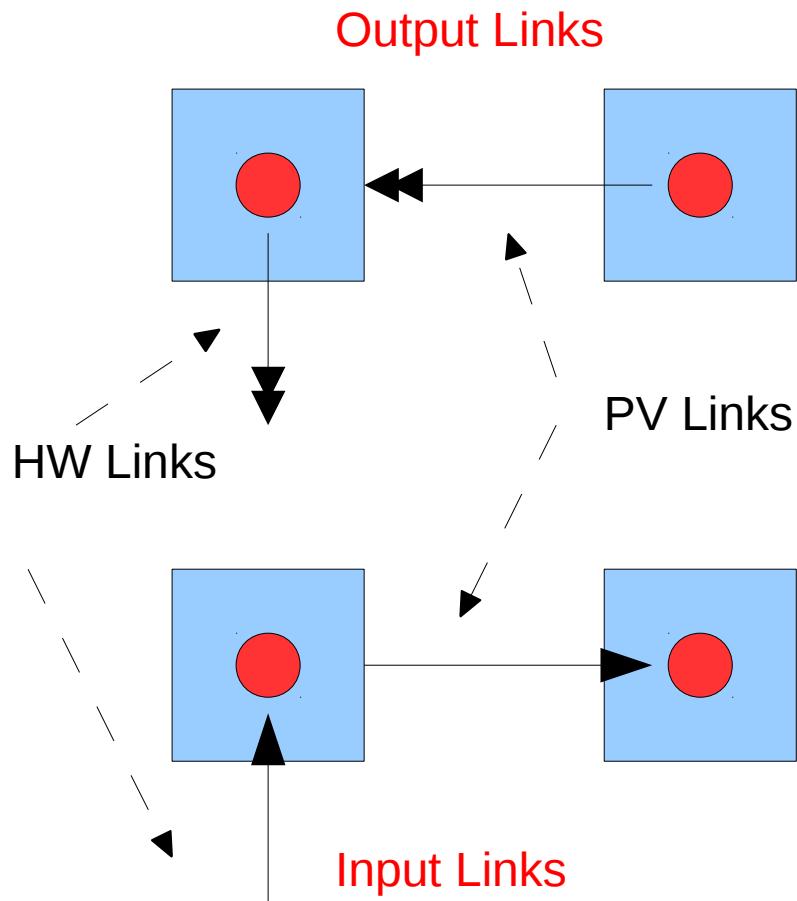
Link Types



- PV Link
 - Connect to another field
 - Any field (almost)
 - Any IOC
 - Local or Remote link
- Hardware Link
 - Token for devSup
 - Identifies (card 7, port 2)

A Link is the least common denominator of a local function call and and CA get/put

Link Types (2)



- Output Link
 - Put data
 - write value to referenced field
- Input Link
 - Get data
 - Read value from referenced field
- Forward Link
 - Cause process()

A Link is the least common denominator of a local function call and and CA get/put

Constant PV Links

- Output

- No action

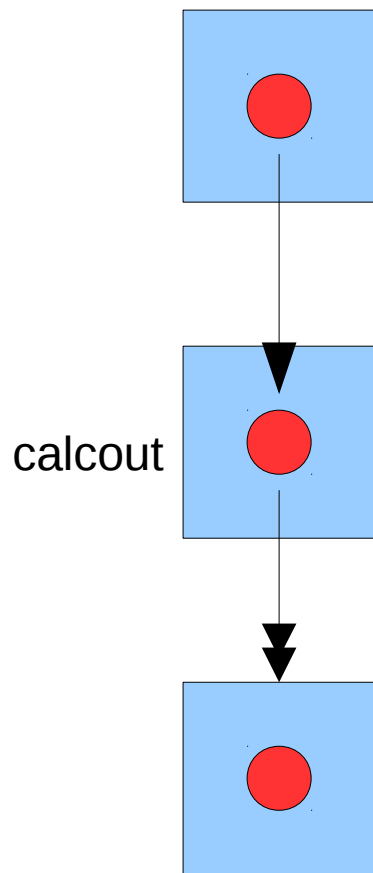
- Input

- Read a constant value (default 0)
 - Store in associated value field (INPA → A)

```
record(calcout, "bpm:y") {  
    field(INPA, "bpm:up")  
    field(INPB, "bpm:down")  
    field(INPC, "14")  
    field(CALC, "(A-B)/(A+B+C)")  
}
```

Helps with length limit of CALC field

Scanning and Links



- Link attributes (PP or NPP)
- Input Links
 - NPP – Read current field value.
 - PP – Process then read new field value
- Output Links
 - NPP – Write field.
 - PP – Write field then process using new value.
- Default is NPP

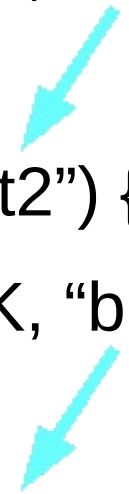
Process Passive Links

- field(INP, “name NPP”)• Gets current value
- field(OUT, “name NPP”)• Sets new value
- field(INP, “name PP”)• If 'name' is SCAN Passive then process()
• Gets new value
- field(OUT, “name PP”)• Sets new value• If Passive then process()

Forward Links

- NPP Links transfer data
- PP Links transfer data and causes processing
- Forward links only causes processing
- Every record has 1 (FLNK)
 - fanout has 8

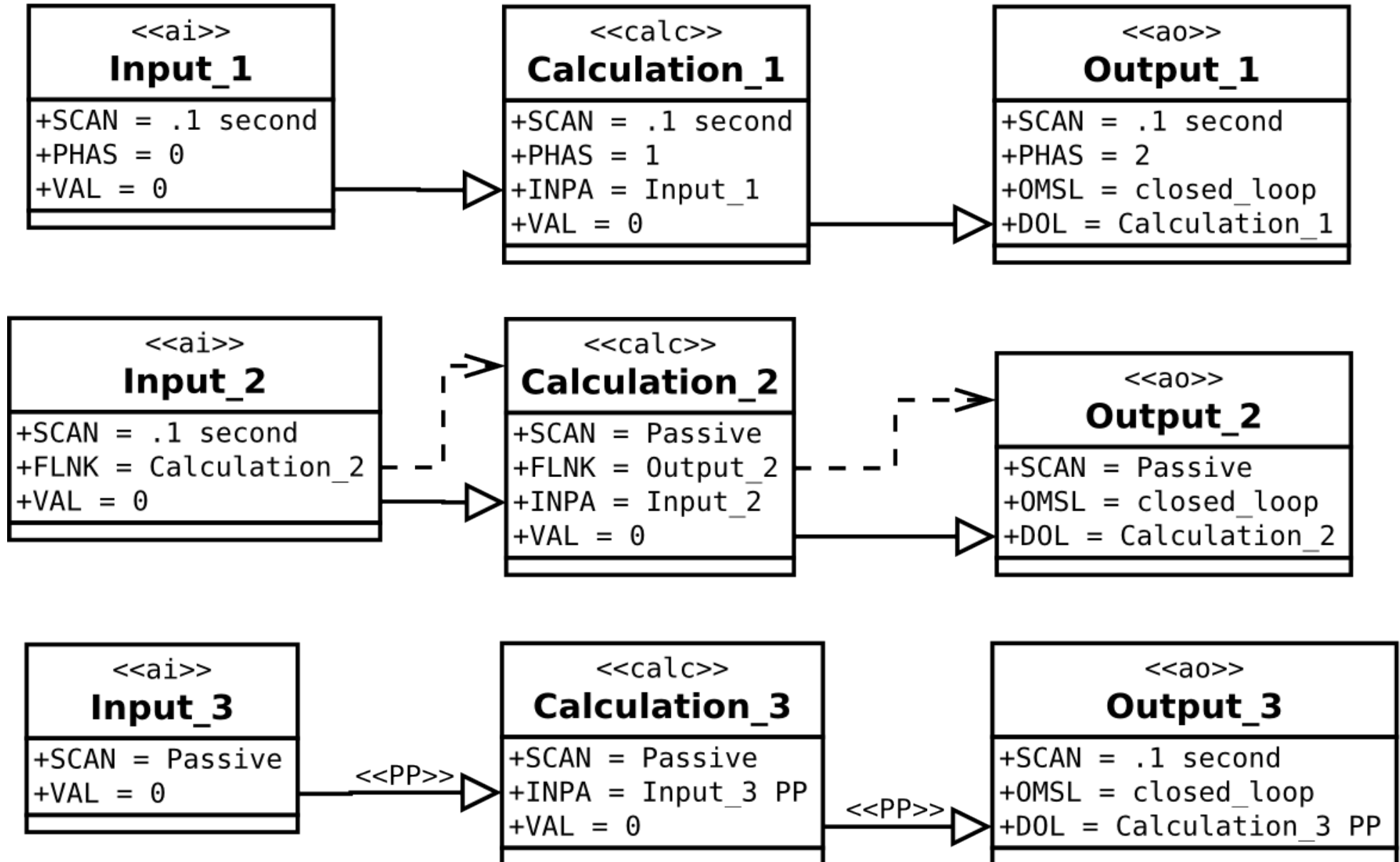
```
record(bi, "bit1") {  
    field(SCAN, "1 second")  
    field(FLNK, "bit2")  
}  
record(bi, "bit2") {  
    field(FLNK, "bit3")  
}  
record(bi, "bit3") {  
}
```



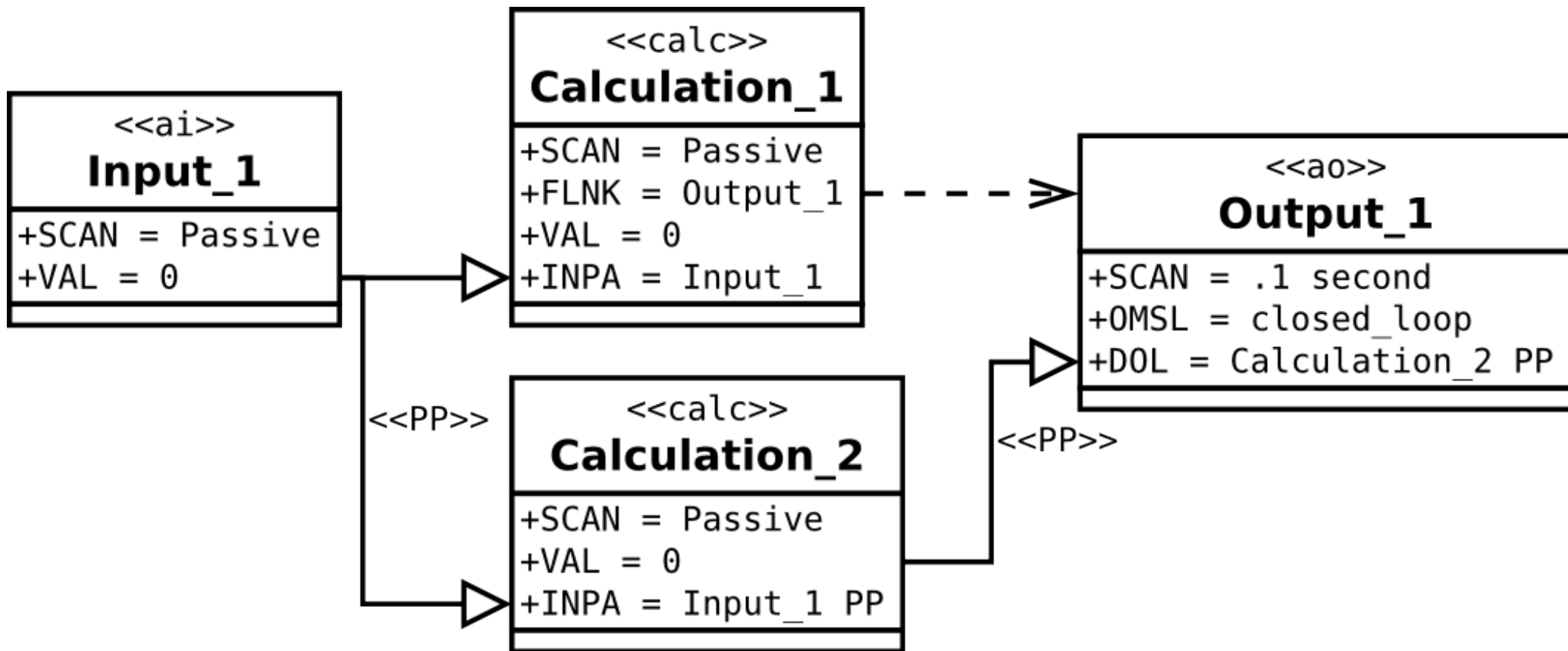
Record Processing Chains

- In most databases most records (~50%) don't belong to a processing chain.
- Processing chains are usually short (2-3 records).
- Most databases have a few longer chains.
 - Eg. an ADC might have a long chain of processing triggered when it samples

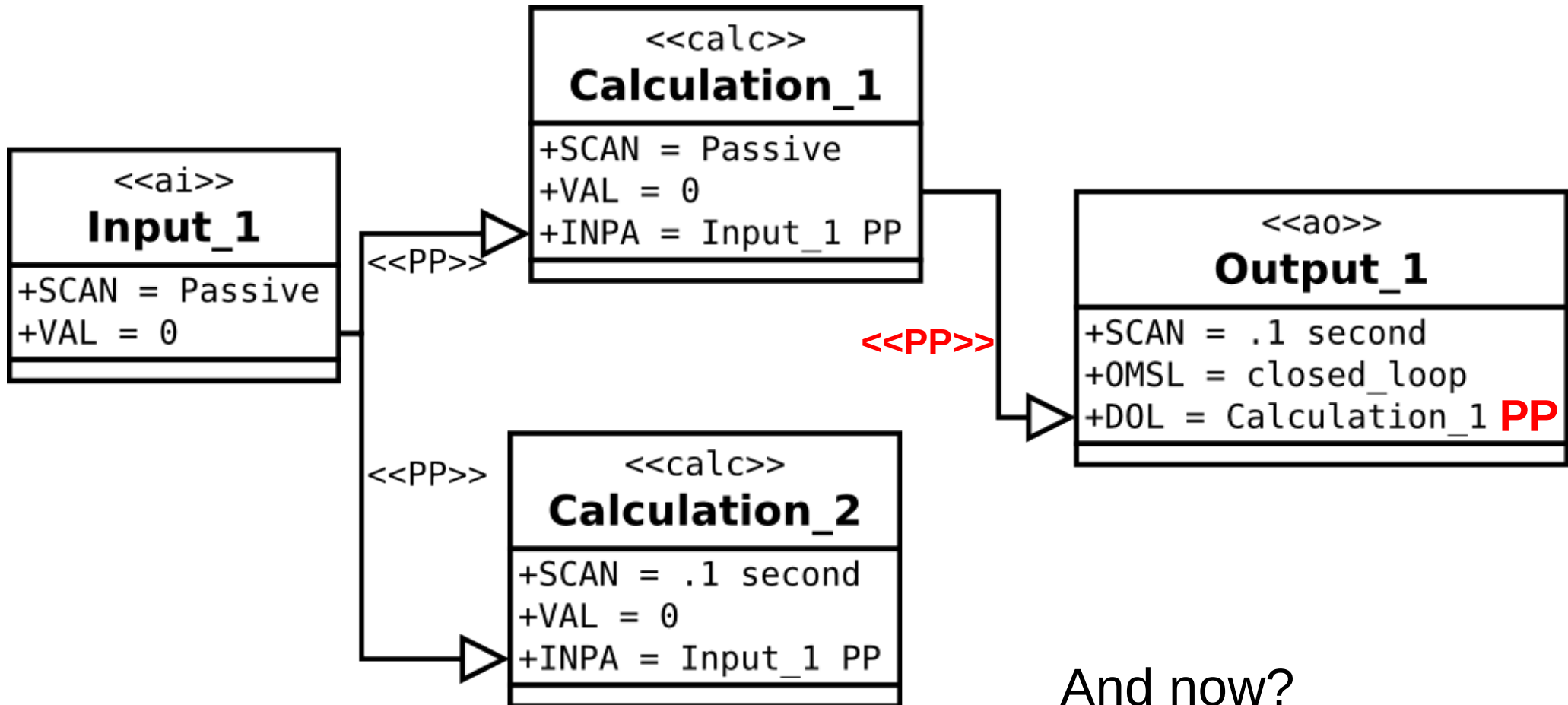
Processing Chains



Which record is never processed?

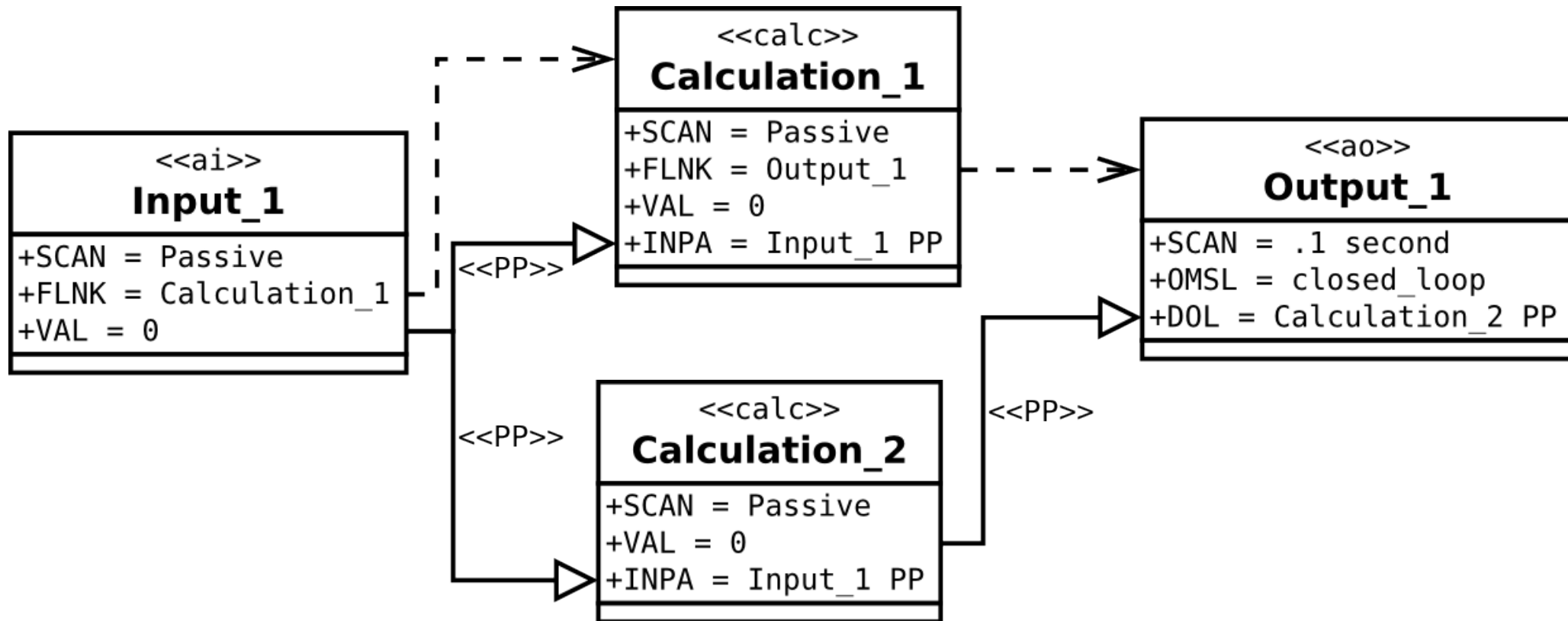


How often is Input_1 scanned?

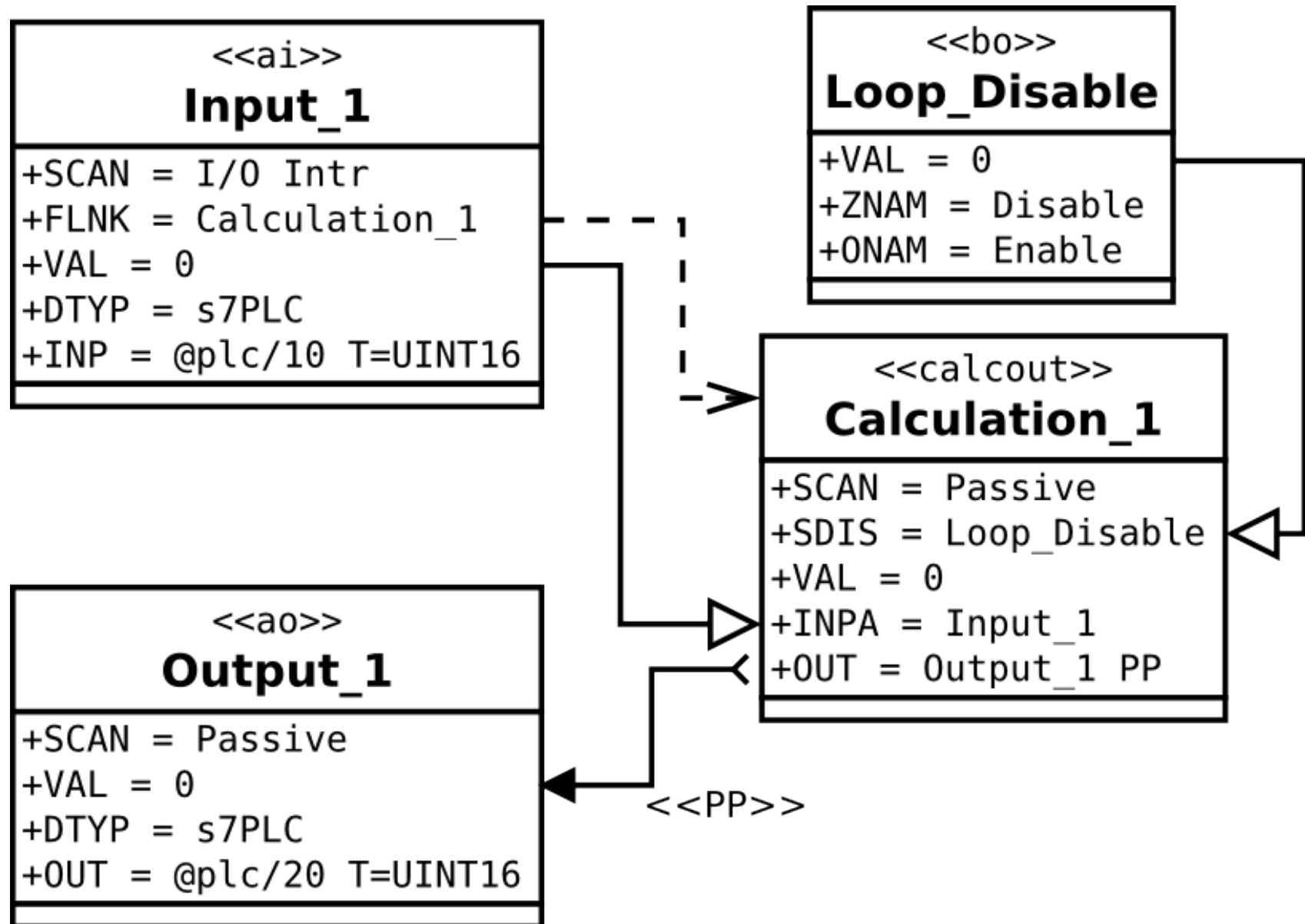


And now?

What happens here?



Output Links and SDIS



Link Types (3)



Input Link



Output Link



"Process" Link

PV Link

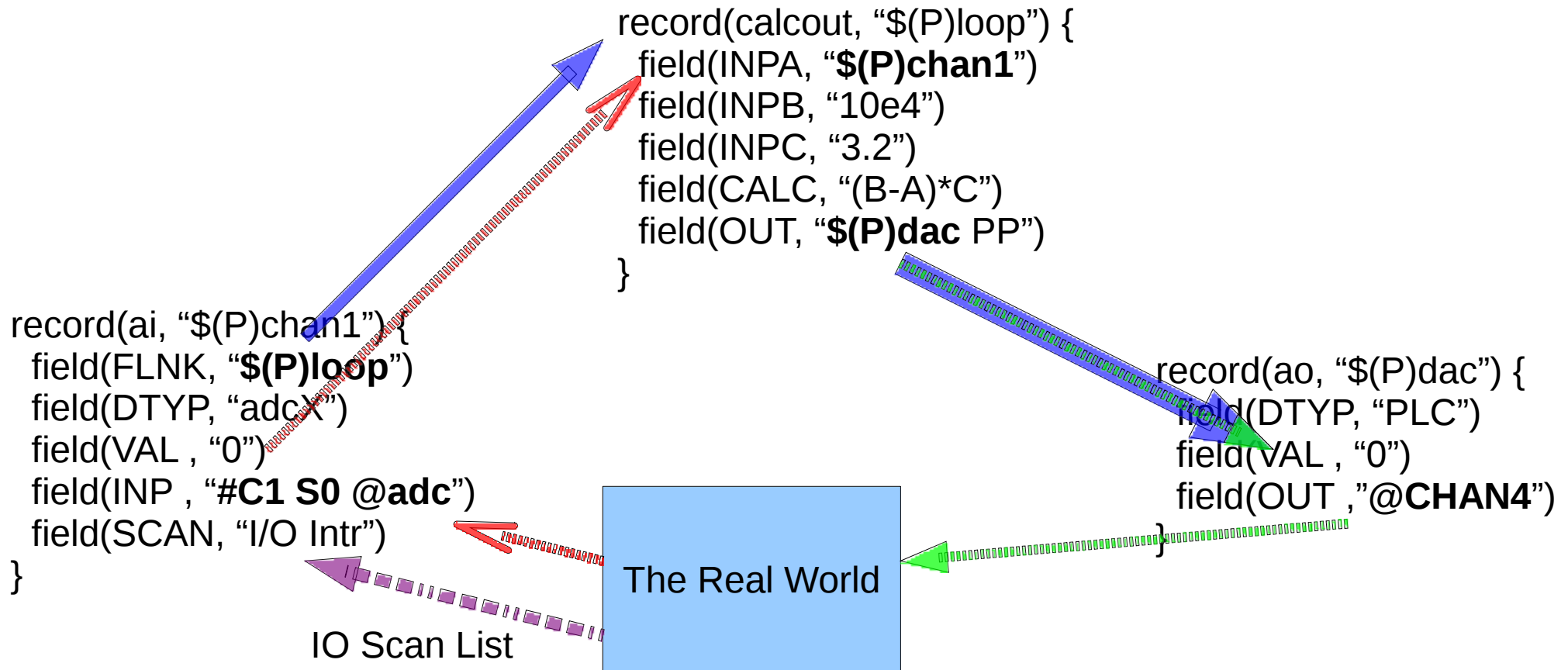
Read a value from a field

Write a value to a field

HW Link

Read a value from a device/driver

Write a value to a device/driver



Add process chain examples
Pass 1 overview
Pass 2 step through

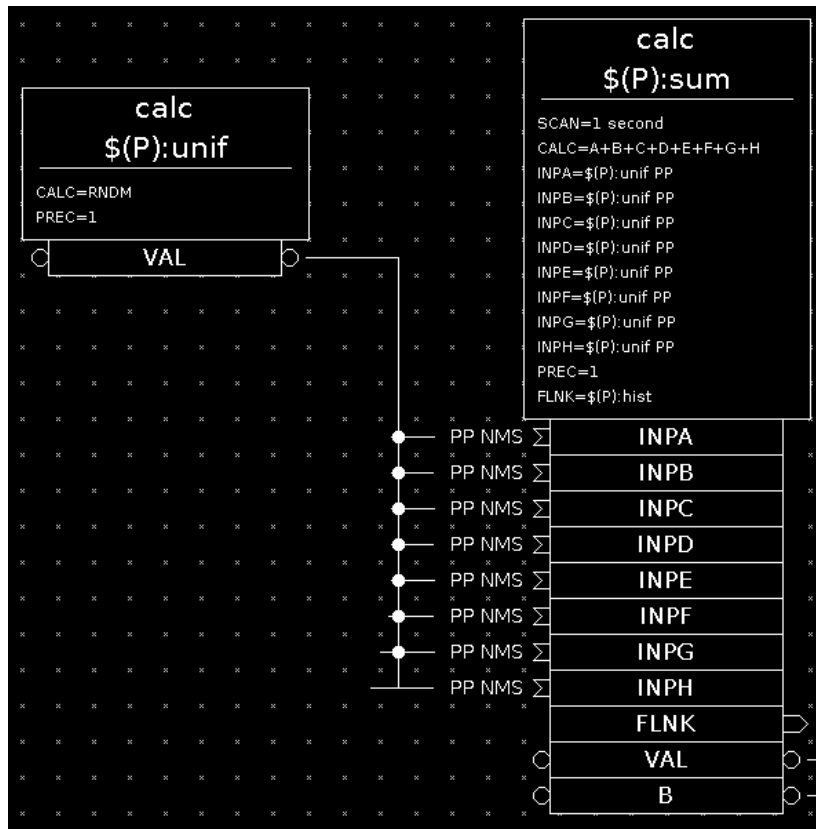
10 Minute Break

Exercise 2 – Random numbers

- calc expressions can contain 'RNDM' to generate a uniform random number [0,1].
 `field(CALC, "RNDM")`
- A Gaussian distribution can be approximated by summing 8 such numbers.
- What does '\$(P):sum' in gauss.db actually use.

Example 2 – Database

```
record(calc, "$(P):unif") {
  field(CALC, "RNDM")
  field(PREC, "1")
}
```



```
record(calc, "$(P):sum") {
  field(SCAN, "1 second")
  field(CALC, "A+B+C+D+E+F+G+H")
  field(INPA, "$(P):unif")
  field(INPB, "$(P):unif")
  field(INPC, "$(P):unif")
  field(INPD, "$(P):unif")
  field(INPE, "$(P):unif")
  field(INPF, "$(P):unif")
  field(INPG, "$(P):unif")
  field(INPH, "$(P):unif")
  field(PREC, "1")
  field(FLNK, "$(P):hist")
}
```

Alarms

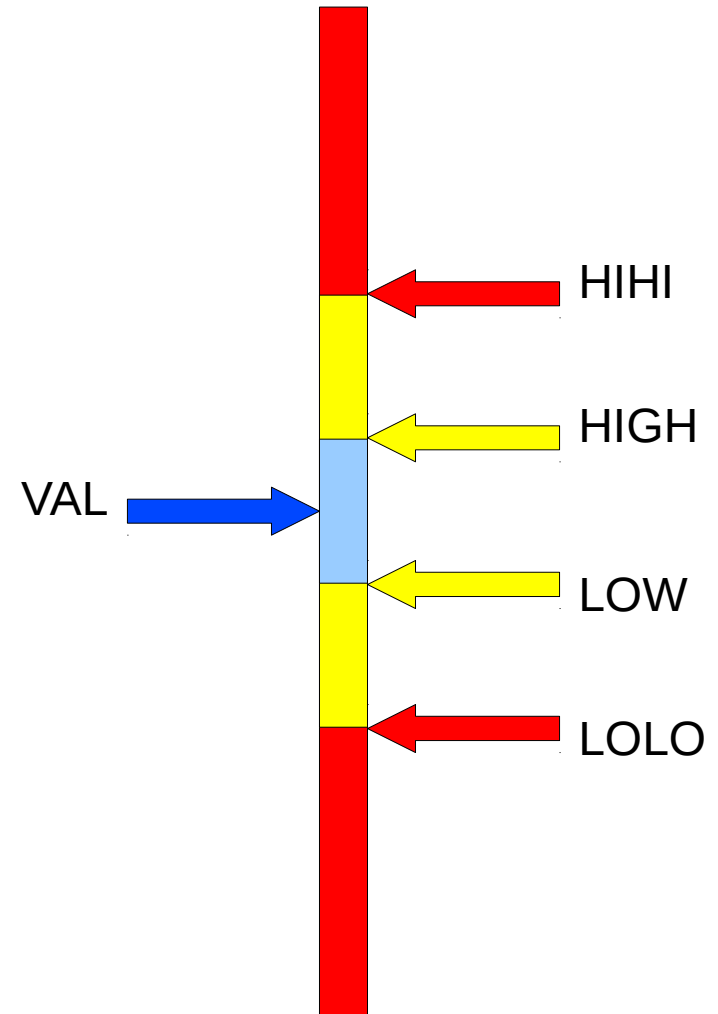
- Each record has an alarm severity and status
- Severity: NO_ALARM, MINOR, MAJOR, INVALID
- Status: READ, WRITE, LINK, ...
- Conditions defined in record and device support
- Highest severity shown

Uses of Alarms

- Alarms are Meta-Data (like timestamps)
- For Operator
 - Value is out of range (Major, Minor)
- For calculations
 - Value not to be believed (Invalid)

Value Level Alarms

- For ao,ai, longout, longin
- Severity
 - NO_ALARM, MINOR, MAJOR, INVALID
 - Default: NO_ALARM
- Levels
 - HIHI, HIGH, LOW, LOLO



Value Level Alarms

- For ao, ai, longout, longin
- Severity
 - NO_ALARM, MINOR, MAJOR, INVALID
 - Default: NO_ALARM
- Levels
 - HIHI, HIGH, LOW, LOLO

```
record(ai, "my:adc") {  
    field(HIGH, "14")  
    field(HSV, "MINOR")  
    field(LOW, "-12")  
    field(LSV, "MAJOR")  
}
```

Invalid Alarms

- For: ao, bo, calcout, mbbo, stringout

- IVOA

- Continue Normally
 - Don't drive outputs
 - Set output to IVOV

Doesn't clip like
DRVH DRVL

- IVOV

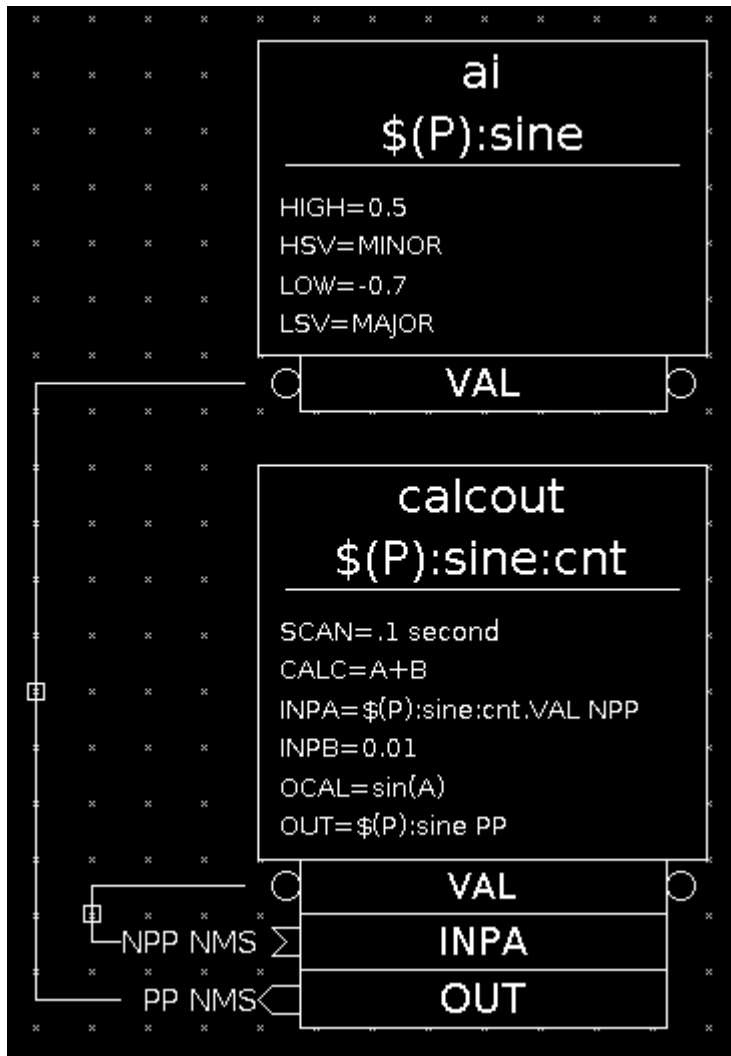
- Same type as VAL

```
record(ao, "my:dac") {  
    field(HIGH, "255")  
    field(HSV, "INVALID")  
    field(LOW, "0")  
    field(LSV, "INVALID")  
    field(IVOA, "Set output  
to IVOV")  
    field(IVOV, "0")  
}
```

Exercise 3 – Sine

- Use a counter calcout and another calcout record to generate a sine wave.
- Set a minor alarm when >0.5 and a major alarm when <-0.7
- Use only one calcout for both count and sine (Hint: OCAL and OVAL fields)

Exercise 3 – Database



```
record(calcout, "$(P):sine:cnt") {
  field(SCAN, ".1 second")
  field(CALC, "A+B")
  field(INPA, "$(P):sine:cnt.VAL NPP")
  field(INPB, "0.01")
  field(OCAL, "sin(A)")
  field(DOPT, "Use OCAL")
  field(OUT, "$(P):sine PP")
}
```

```
record(ai, "$(P):sine") {
  field(HIGH, "0.5")
  field(HSV, "MINOR")
  field(LOW, "-0.7")
  field(LSV, "MAJOR")
}
```

Link and Alarms

- Link attribute
MS/NMS
- NMS (default)
 - Just get/put the value
- MS
 - Get/put value and
alarm severity
- Propagate Alarms

```
record(ai, "one") {  
    field(HIGH, "10")  
    field(HSV, "MINOR")  
}
```

```
record(ai, "two") {  
    field(INP, "one MS PP")  
    field(SCAN, "1 second")  
}
```

Analog Scaling

- For: ao, ai
- Go from RVAL (integer) → VAL (float)
 - field(DTYP, "Raw Soft Channel")
 - Some other device support
- Players: LINR, ROFF, ASLO, AOFF, ESLO, EOFF
- $VAL = (RVAL + ROFF) * ASLO + AOFF$
- If LINR="LINEAR"
- $VAL = ESLO * VAL + EOFF$
- EGU="some string"

Analog Scaling Example

```
record(ai, "adc1") {  
    field(DTYP, "myadc")  
    field(ASLO, "0.1")  
    field(AOFF, "10")  
    field(ESLO, "")  
    field(EOFF, "")  
    field(EGU, "F")  
}
```

- ADC calibration
 - 0.1 degrees/count
 - Count 0 is 10 deg. C
- $C \rightarrow F$

Record Timestamps

- Each record has one timestamp
 - Shared by all fields
- Updated when processed
 - NOT when values change
 - Defaults to 0 (Jan. 1 1990) until first processed
- Problem for non-value fields
 - camonitor rename.SCAN
- Makes archiving these fields tricky

Controlling Timestamps

- Fields:
 - TSEL – Timestamp selection input link
 - TSE – Timestamp Event
 - TIME – Stored time (Not directly readable)
- Choose where timestamp is taken from
 - Uses generalTime framework ($\geq 3.14.9$)

Timestamp Selection

```
/* recGblTimeStamp() */
```

```
If TSEL is not CONSTANT
```

```
    if TSEL points to a TIME field    Special hidden TSEL link option
```

```
        TIME=dbGetTimeStamp(TSEL)    Takes time from record pointer  
                                     or CA metadata
```

```
    return
```

```
    TSE=dbGetLink(TSEL)    Read an integer to use as the event number
```

```
if TSE != -2
```

```
    TIME=epicsTimeGetEvent(TSE)    Takes time of last occurrence  
                                   of event #
```

```
return
```

generalTime Events

- Normal events: 1-32k
 - **Not** database events
 - Hooks to event timing system (if present)
 - No default provider
- Special events
 - 0 – Current wall clock time
 - -1 – Best event time (???)
 - -2 – Allow device support to set TIME

TS Examples (1)

- Current wall clock time
 - Default

```
record(ai, "name") {  
    ...  
}
```
- Specific gT event

```
record(ai, "name") {  
    field(TSE, "42")  
}
```
- From another record

```
record(calc, "name") {  
    field(INPA, "other.VAL")  
    field(TSEL, "other.TIME")  
}
```
- Take event # from another record

```
record(calc, "name") {  
    field(TSEL, "other.TSE")  
}
```

TSEL and CA_LINK

- Potential mistake if 'aaa' and 'bbb' are in the same IOC.
- Example 1 (Wrong)
 - aaa.TSEL is DB_LINK
 - 'aaa' and 'bbb' in same lock set
 - Timestamp may not match value
- Example 2 (Right)
 - aaa.TSEL is CA_LINK
 - 'aaa' and 'bbb' in different lock sets
- Rule: **TSEL must use same type of link as value link.**

```
record(calc, "aaa") {  
    field(INPA, "bbb CPP")  
    field(TSEL, "bbb.TIME")  
}
```

↓
Copies latest
From record struct

```
record(calc, "aaa") {  
    field(INPA, "bbb CPP")  
    field(TSEL, "bbb.TIME CA")  
}
```

↓
Uses meta-data
from previous
monitor/get.

Set Timestamp is Device Support

- Set TSE to -2

```
record(ai, "aaa") {  
    field(TSE, "-2")  
}
```

- In dset

```
read_ai(aiRecord* prec) {  
    if(prec->tse==epicsTimeEventDeviceTime) { /* -2 */  
        prec->time.secPastEpoch=... /* Use EPICS Epoch (Jan 1 1990) */  
        prec->time.nsec=...  
    }  
}
```


Common Record Fields

- Device support: DTYP, INP, OUT
- Links: INP, OUT
- Scanning
 - SCAN, PINI, PACT
- Values: VAL, OVAL, RVAL, RBV, RRBV
- Alarms
 - STAT, SEVR
 - HIHI, HIGH, LOW, LOLO
 - HHSV (high high severity), HSV, LSV, LLSV
- Value+Alarm
 - IVOA (invalid out action), IVOV (invalid out value)
- Display: EGU, DRVH, DRVL

Review

- Records are a collection of fields
- Records have an action
 - Execution of action is Processing
- Links read from/write to a field
 - Links move values
 - Links move meta-data (timestamp and alarm)
 - Links cause Processing

End

Questions?