Argonne NATIONAL LABORATORY

EPICS Database Practice

Tim Mooney (AES/SSG, Argonne)

Includes material from:

Andrew Johnson (AES/SSG, Argonne)

U.S. DEPARTMENT OF ENERGY

EPICS record types

- Where do record types come from?
  - EPICS Base (<base>/src/rec)
    - General purpose record types
    - Documented in the EPICS Record Reference Manual
    - No record-type specific operator displays or databases
  - EPICS collaboration
    - General purpose, and application-specific, record types
    - Some are supported for use by collaborators (some are NOT)
    - Some come with record-type specific displays and database templates
  - Custom record types can be written by any EPICS developer, and added to specific EPICS IOC applications as needed.
    - Not in the scope of this lecture

The Record Reference Manual

- Where is it?
  - EPICS Home > Base > R3.14.12 > Reference Reference Manual (wiki)
- What is in it?
  - Database Concepts (good review)
  - Fields common to all records
  - Fields common to many records
  - Record Type Documentation
    - Provides a description of the fields and record processing functionality of each record type in base.
- When would I use it?
  - Skim through before writing any databases
  - Read through before writing any new record types
  - Otherwise, use as reference

Manual Outline

- Preface, Concepts, Common fields: Essential background information
  - Note special meaning of the words *scan*, *process*, *address*, *link*, and *monitor*
- Record reference
  - Some parts may be out of date
  - Descriptions of each field, record processing, and other information needed when writing device support
  - Contains lots of tables like this:

Field	Summary	Type	DCT	Initial	Access	Modify	Rec Proc Monitor	PP
EGU	Engineering Units	STRING [16]	Yes	null	Yes	Yes	No	No
HOPR	High Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
LOPR	Low Operating Range	FLOAT	Yes	0	Yes	Yes	No	No
PREC	Display Precision	SHORT	Yes	0	Yes	Yes	No	No
NAME	Record Name	STRING [29]	Yes	Null	Yes	No	No	No
DESC	Description	STRING [29]	Yes	Null	Yes	Yes	No	No

Collaboration supported records

- Where are they found?
  - Soft-support list: http://www.aps.anl.gov/epics/modules/soft.php
  - The tech-talk email list (check archives first, then ask)
- The soft-support list contains entries like this (and entries for other kinds of soft support):

Class	Name	Description	Contact	Link
record	epid	Enhanced PID record	Mark Rivers	CARS:epid Record
record	genSub	Multi-I/O subroutine, handles arrays	Andy Foster	OSL:epics
...	...	...	...	...
record	table	Control an optical table	Tim Mooney	APS:synApps /optics
record	timestamp	...exports its timestamp as a string	Stephanie Allison	SLAC:time stamp

Input Records

- ai - Analog input [BASE]
  - Read analog value, convert to engineering units, four alarm levels, simulation mode
- aai – Array analog input [BASE]
  - Read array of analog values, simulation mode
- bi - Binary input [BASE]
  - Single bit, two states, assign strings to each state, alarm on either state or change of state, simulation mode
- mbbi - Multi-bit binary input [BASE]
  - Multiple bit, sixteen states, assign input value for each state, assign strings to each state, assign alarm level to each state, simulation mode
- mbbiDirect – mbbi variant [BASE]
  - Read an unsigned short and map each bit to a field (16 BI records in one)

Input Records (cont..)

- stringin - String input [BASE]
  - 40 character (max) ascii string, simulation mode
- longin - Long integer input [BASE]
  - Long integer, four alarm levels, simulation mode
- waveform – array input [BASE]
  - Configurable data type and array length
- *mca – multichannel analyzer [synApps]*
  - Supports multichannel analyzers, multichannel scalers, and other array-input hardware
- *scaler [synApps]*
  - Controls a bank of counters

Algorithms/Control Records - Calc

- calc - run-time expression evaluation [BASE]
  - 12 input links, user specified expression (algebraic, trig, relational, boolean, bit-wise, “?:” and “:=” operators), four alarm levels
  - Sample expressions:
    - `0` read: “VAL = 0”
    - `A` returns value of record’s “A” field
    - `A+B`
    - `b*sin(a*d2r); a:=a+1`
    - `(A+B)<(C+D) ? E : F*G/100`
- calcout – calc variant [BASE]
  - Conditional output link, second output CALC expression (OCAL), output delay, and output event
  - Output-link options : "Every Time”, "On Change", "When Zero", "When Non-zero", "Transition To Zero", "Transition To Non-zero“

Algorithms/Control Records - Calc

- *sCalcout – calcout variant* [synApps]
  - Has both numeric fields (A,B,..L) and string fields (AA,BB,..LL)
  - Supports both numeric and string expressions. E.g.,
    - `A+DBL("value is 3.456")` -> 4.456
    - `printf("SET:VOLT:%.2lf", A+4)` -> "SET:VOLT:5.00"
  - Additional output-link option: "Never"
  - Can wait for processing started by output link to finish

- *transform – calc/seq variant* [synApps]
  - Like 16 calcout records (but outlinks are not conditional)
  - Expressions read all variables, but write to just one.
  - Uses sCalcout record's calculation engine
  - Example expressions:
    - CLCA: `2` read: "<transform>.A = 2"
    - CLCB: `A+1+C` uses new value of 'A', old value of 'C'
  - Expressions can use variable names:
    - ($left+$right)/2 where $left and $right are defined in comment fields

Algorithms/Control Records - Array

- compress [BASE]
  - Collect a series of scalar values into a scalar or an array.
  - Input link can be scalar or array
  - Algorithms include N to 1 compression (highest, lowest, or average), circular buffer of scalar input.
- histogram [BASE]
  - Accumulates histogram of the values of a scalar PV.
- subArray[BASE]
  - Extracts a sub-array from an array PV.
- aSub – Array subroutine [BASE]
  - Multiple input and output fields and links
  - Handles arrays

Algorithms/Control Records - List

- dfanout – Data fanout [BASE]
  - Writes a single value to eight output links.
- fanout - Execution fanout [BASE]
  - Forward links to six other records
  - Selection mask
- sel – Select [BASE]
  - 12 input links, four select options [specified, highest, lowest, median], four alarm levels.
- seq – Sequence [BASE]
  - Ten “Input link/Value/Output link” sets: [inlink, delay, value, outlink]
  - Selection mask.
- *sseq – seq variant* [BASE]
  - seq record for string or numeric data
  - optional wait for completion after each set executes.

Algorithms/Control Records - Loop

- *sscan – scan variant* [synApps]
  - Uses ca_put_callback() to detect completion.
  - Four triggers, 70 detector signals (arrays, scalars, or mixed)
  - Array-prepare trigger at end of scan
  - Number of data points stored is limited only by memory
  - Supports scan pause; before/after-scan action; move-to-peak.
  - Handshake permits data-storage client to write old data while new data are being acquired.

Algorithms/Control Records - Subroutine

- sub – Subroutine [BASE]
  - 12 input links, user provided subroutine, four alarm levels.
- aSub – Array Subroutine [BASE]
  - 21 input links, 21 output links, controllable data types and array sizes
  - user provided initialize and process subroutines

Algorithms/Control Records - Other

- event [BASE]
  - Posts a “soft” event which may trigger other records to process.
  - Simulation mode
- epid [synApps]
  - Proportional/Integral/Derivative Control
- permissive – handshake [BASE]
  - Implements a client-server handshake
- state – string state value [BASE]
  - Implements a string, for client-server communication

Output Records

- ao - Analog output [BASE]
  - Write analog value, convert from engineering units, four alarm levels, closed_loop mode, drive limits, output rate-of-change limit, INVALID alarm action, simulation mode
- aao – Array analog output [BASE]
  - ao for arrays
- bo - Binary output [BASE]
  - Single bit, two states, assign strings to each state, alarm on either state or change of state, closed_loop mode, momentary 'HIGH', INVALID alarm action, simulation mode
- longout – 32-bit integer output [BASE]
  - Write long integer value, four alarm levels, closed_loop mode, INVALID alarm action, simulation mode

Output Records (cont..)

- mbbo - Multi-bit binary output [BASE]
  - Multiple bit, sixteen states, assign output value for each state, assign strings to each state, assign alarm level to each state, closed_loop mode, INVALID alarm action, simulation mode
- mbboDirect - mbbo variant [BASE]
  - 16 settable bit fields that get written as a short integer to the hardware, closed_loop mode, INVALID alarm action, sim. Mode
- motor – Control a motor [synApps]
  - Controls both stepper and servo motors
  - Supports most common motor controllers
- stringout – String output [BASE]
  - Write a character string (40 max), closed_loop mode, INVALID alarm action, simulation mode

Examples of Custom Records

- rf - RF Amplitude Measurements [ANL]
  - Sample time, measurement in watts and db, waveform acquired through sweeping sample time
- bpm - Beam Position Monitor [ANL]
  - Four voltage inputs, numerous calibration constants, X-Y-I outputs, waveforms for each input
- runcontrol – Client Semaphore [ANL]
  - Client requests temporary control of part of machine, must ping record regularly to keep it; other clients can see current owner (cooperative, not enforced)
- *Many* others that are site-specific written at other labs

Which record is right for …

- Soft parameters entered by an operator
  - ao has **DRVH**, **DRVL**, **OROC**, **OMSL** = supervisory/closed_loop
  - mbbo provides enumerated options which can be converted to numeric constant values (use **DTYP** = Raw Soft Channel)
  - Normally one does *not* use input records for operator parameters
- Multiple output actions
  - The seq (sequence) record can read a different data source for each link output value
  - The dfanout record “fans out” data from a single source to multiple links
- Different output actions based on an operator selection
  - A calcout record can conditionally process one or more sequence records
  - An mbbo (**DTYP** = Raw Soft Channel) forward linked to a sequence record in “masked” mode. The mask value is read from the mbbo’s **RVAL**, each bit controls whether one link fires

Defining the Database

- How does an IOC know what record *types* and device support options are available ?
  - Record types, device support options, choice menus, and other configuration options are defined in Database Definition files (.dbd)
  - The IOC build process creates a .dbd file containing everything the IOC needs
  - That .dbd file is loaded by the IOC when it starts up

- How does an IOC know about record *instances* (the user's database) ?
  - Record instances are describe in Database files (.db)
  - During the IOC booting process, one or more .db files are loaded
  - The .db files define the record instances for that IOC

Database Definition File

```
menu(menuPriority) {
  choice(menuPriorityLOW,"LOW")
  choice(menuPriorityMEDIUM,"MEDIUM")
  choice(menuPriorityHIGH,"HIGH")
}
menu(menuScan) {
  choice(menuScanPassive,"Passive")
  choice(menuScanEvent,"Event")
  choice(menuScanI_O_Intr,"I/O Intr")
  choice(menuScan10_second,"10 second")
  choice(menuScan5_second,"5 second")
  choice(menuScan2_second,"2 second")
  choice(menuScan1_second,"1 second")
  choice(menuScan_5_second,".5 second")
  choice(menuScan_2_second,".2 second")
  choice(menuScan_1_second,".1 second")
}
```

```
device(ai,CONSTANT,devAiSoftRaw,
            "Raw Soft Channel")
device(ai,BITBUS_IO,devAiIObug,
            "Bitbus Device")
device(ao,CONSTANT,devAoSoftRaw,
            "Raw Soft Channel")
device(ao,VME_IO,devAoAt5Vxi,
            "VXI-AT5-AO")
device(bi,VME_IO,devBiAvme9440,
            "AVME9440 I")
device(bi,AB_IO,devBiAb,
            "AB-Binary Input")
driver(drvVxi)
driver(drvMxi)
driver(drvGpib)
driver(drvBitBus)
```

Extracts from a typical .dbd file

Database Definition File continued...

```
menu(aoOIF) {
  choice(aoOIF_Full,"Full")
  choice(aoOIF_Incremental,
         "Incremental")
}
recordtype(ao) {
  include "dbCommon.dbd"
  field(VAL,DBF_DOUBLE) {
    prompt("Desired Output")
    promptgroup(GUI_OUTPUT)
    asl(ASL0)
    pp(TRUE)
  }
  field(OUT,DBF_OUTLINK) {
    prompt("Output Specification")
    promptgroup(GUI_OUTPUT)
    interest(1)
  }
```

```
  field(OIF,DBF_MENU) {
    prompt("Out Full/Incremental")
    promptgroup(GUI_OUTPUT)
    interest(1)
    menu(aoOIF)
  }
  field(OVAL,DBF_DOUBLE) {
    prompt("Output Value")
  }
  field(PREC,DBF_SHORT) {
    prompt("Display Precision")
    promptgroup(GUI_DISPLAY)
    interest(1)
  }
  ...
}
```

Parts of the ao record type definition from a typical .dbd file

Database File

```
record(bo,"$(user):gunOnC") {
    field(DESC,"E-Gun Enable")
    field(DTYP,"Soft Channel")
    field(ZNAM,"Beam Off")
    field(ONAM,"Beam On")
}
record(calc, "$(user):flameM") {
  field(CALC, "A<32?A+1:0")
  field(SCAN, ".1 second")
  field(INPA, "$(user):flameM.VAL NPP NMS")
}
record(ao,"$(user):cathodeCurrentC") {
    field(DESC,"set cathode current")
    field(DTYP,"Raw Soft Channel")
    field(SCAN,"1 second")
    field(OROC,".5")
    field(PREC,"2")
    field(EGU,"Amps")
    field(DRVH,"20")
    field(DRVL,"0")
    field(HOPR,"20")
    field(LOPR,"0")
}
```

```
record(calc,"$(user):rampM") {
    field(CALC,"A>6.27?0:A+.1")
    field(SCAN,"1 second")
    field(INPA,"$(user):rampM.VAL")
}
record(calc,"$(user):cathodeTempM") {
    field(DESC,"Measured Temp")
    field(SCAN,"1 second")
    field(CALC,"C+(A*7)+(SIN(B)*3.5)")
    field(INPA,"$(user):cathodeCurrentC.OVAL")
    field(INPB,"$(user):rampM.VAL")
    field(INPC,"70")
    field(EGU,"degF")
    field(PREC,"1")
    field(HOPR,"200")
    field(LOPR,"")
    field(HIHI,"180")
    field(LOLO,"130")
    field(HIGH,"160")
    field(LOW,"140")
    field(HHSV,"MAJOR")
    field(HSV,"MINOR")
    field(LLSV,"MAJOR")
    field(LSV,"MINOR")
}
```

Loading Database Files into the IOC

- A typical startup script (st.cmd) might contain

```
dbLoadDatabase("../../dbd/linacApp.dbd")
dbLoadRecords("../../db/xxLinacSim.db","user=studnt1")
iocInit
```

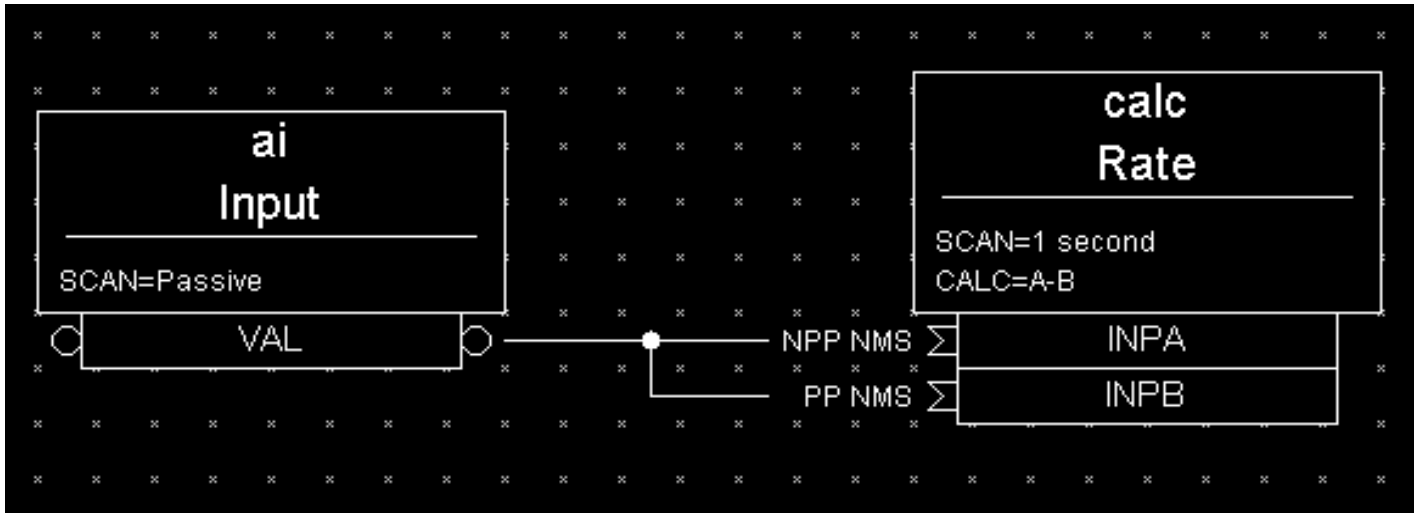
- One or more database definition files (.dbd) must be loaded first
  - All record types used in the database files must have been defined in the definition file
- Values for macros used within the database file (e.g. `$(user)`) can be specified when loading
  - This allows a database to be loaded more than once with different record names and I/O addresses each time
- The iocInit command starts database processing

Creating Database Files

- Since the database file is a simple ASCII text file, it can be generated by numerous applications … as long as the syntax is correct
  - Text editor
  - Script (Perl, Python, shell, awk, sed, …)
  - Relational Database (Oracle, MySQL, Postgresql)
  - EPICS-aware graphical Database Configuration Tools:
    - gdct (not recommended for new users)
    - VDCT (CosyLab)
    - TDCT (TRIUMF)
- An EPICS-aware tool reads the IOC's .dbd file to get the available record types, fields in each record, choice values for enumerated fields and all available device support
- A hierarchical graphical tool is helpful for developing complex databases

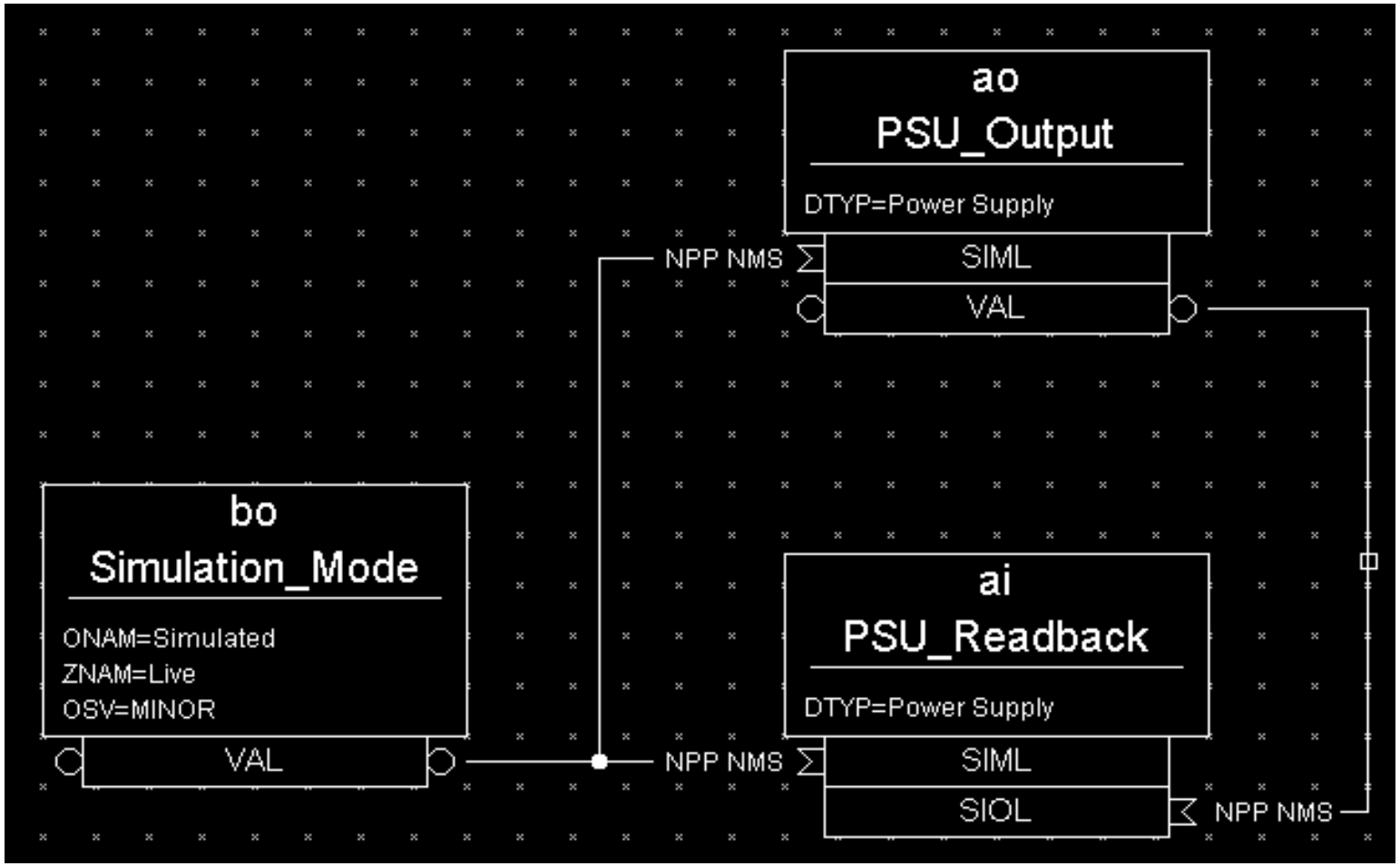
Database Examples

Calculating “Rate-of-Change” of an Input



Rate.INPA fetches data that is 1 second old — it does not request processing of the ai record. Rate.INPB fetches current data by requesting the ai record to process. The subtraction of these two values reflects the ‘rate of change’ (difference/sec) of the Input signal.

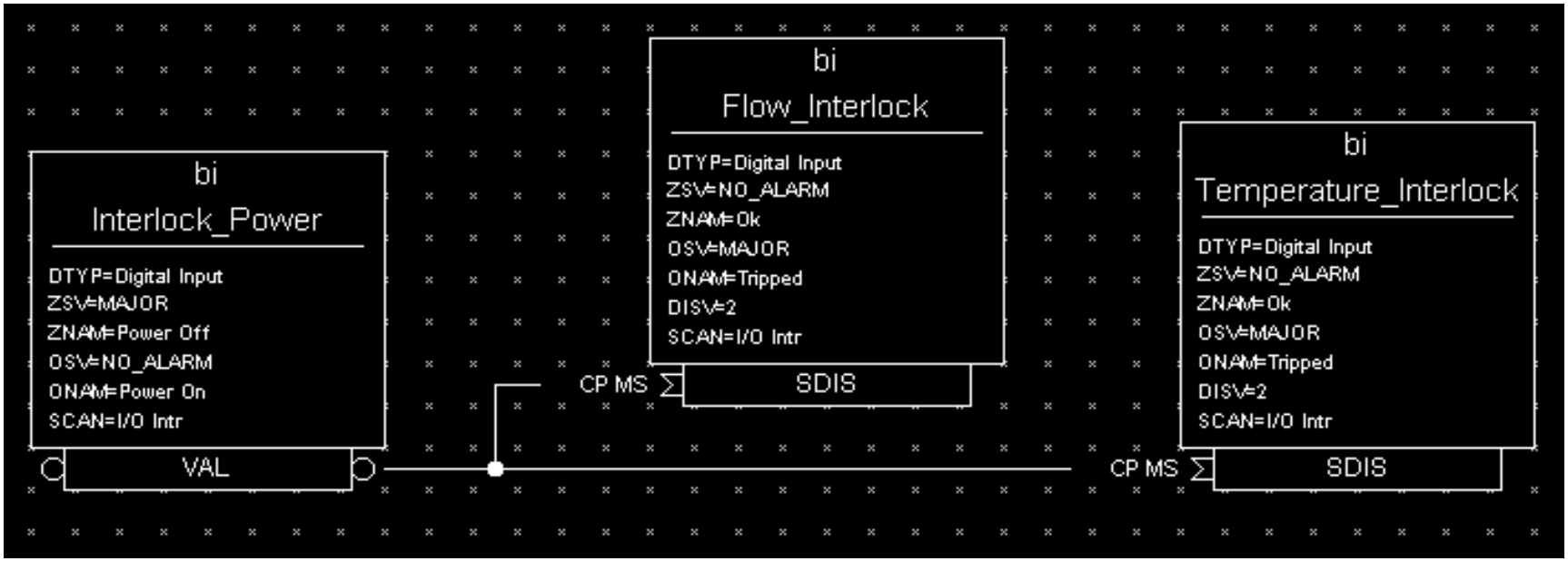
Database Examples



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

Database Examples

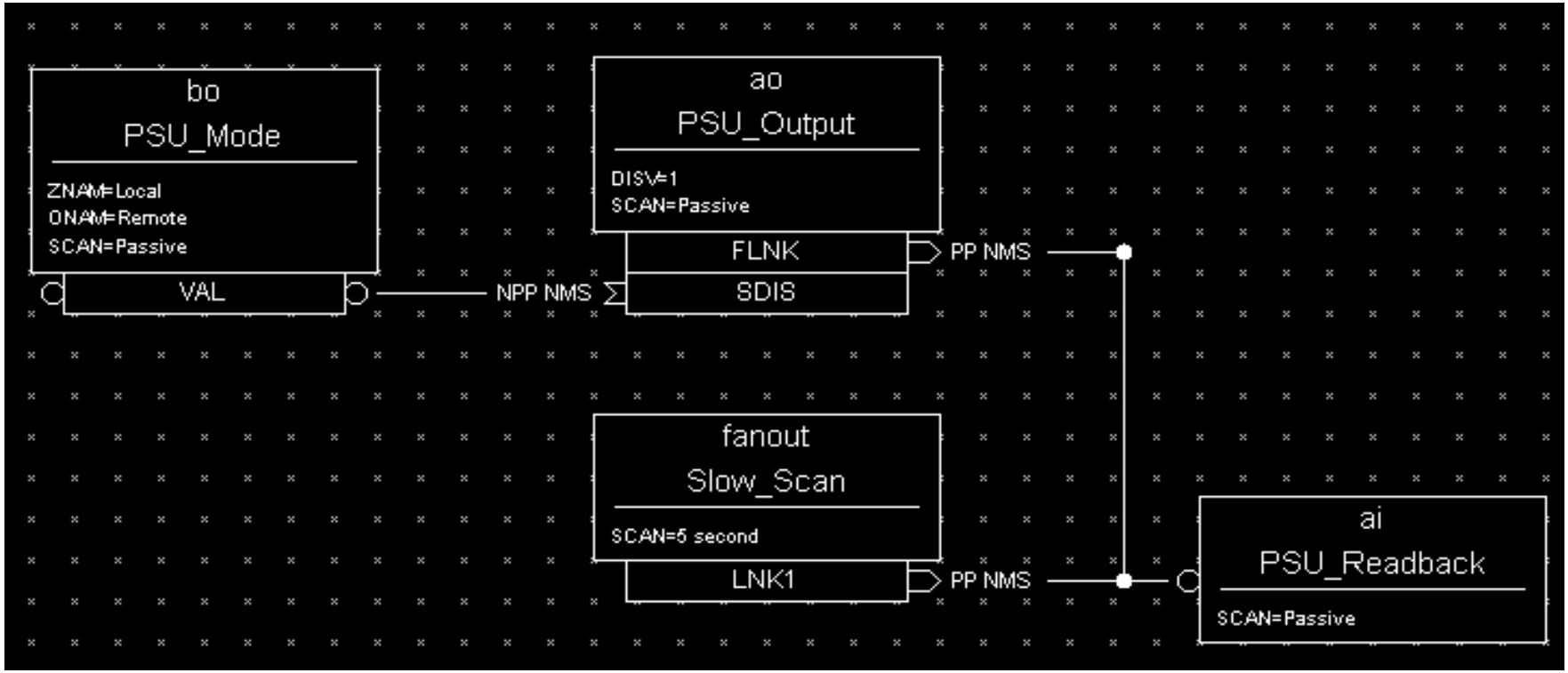
Maximize Severity



When the chassis is powered off, both Flow_Interlock and Temperature_Interlock will indicate Ok. We can still force these records into an alarm state by pointing their SDIS links to the Interlock_Power record with an MS (maximize severity) flag. Their DISV fields must be 2 so record processing is not disabled.

Database Examples

Slow Periodic Scan with Fast Change Response

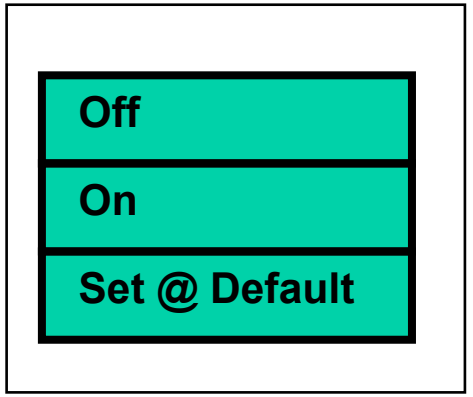


The ai record gets processed every 5 seconds AND whenever the ao record is changed. This provides immediate response to an operator's changes, even though the normal scan rate is very slow. Changes to the power supply settings are inhibited by the bo record, which provides a Local/Remote switch.

Database Examples

Different Actions Based on Operator Selection

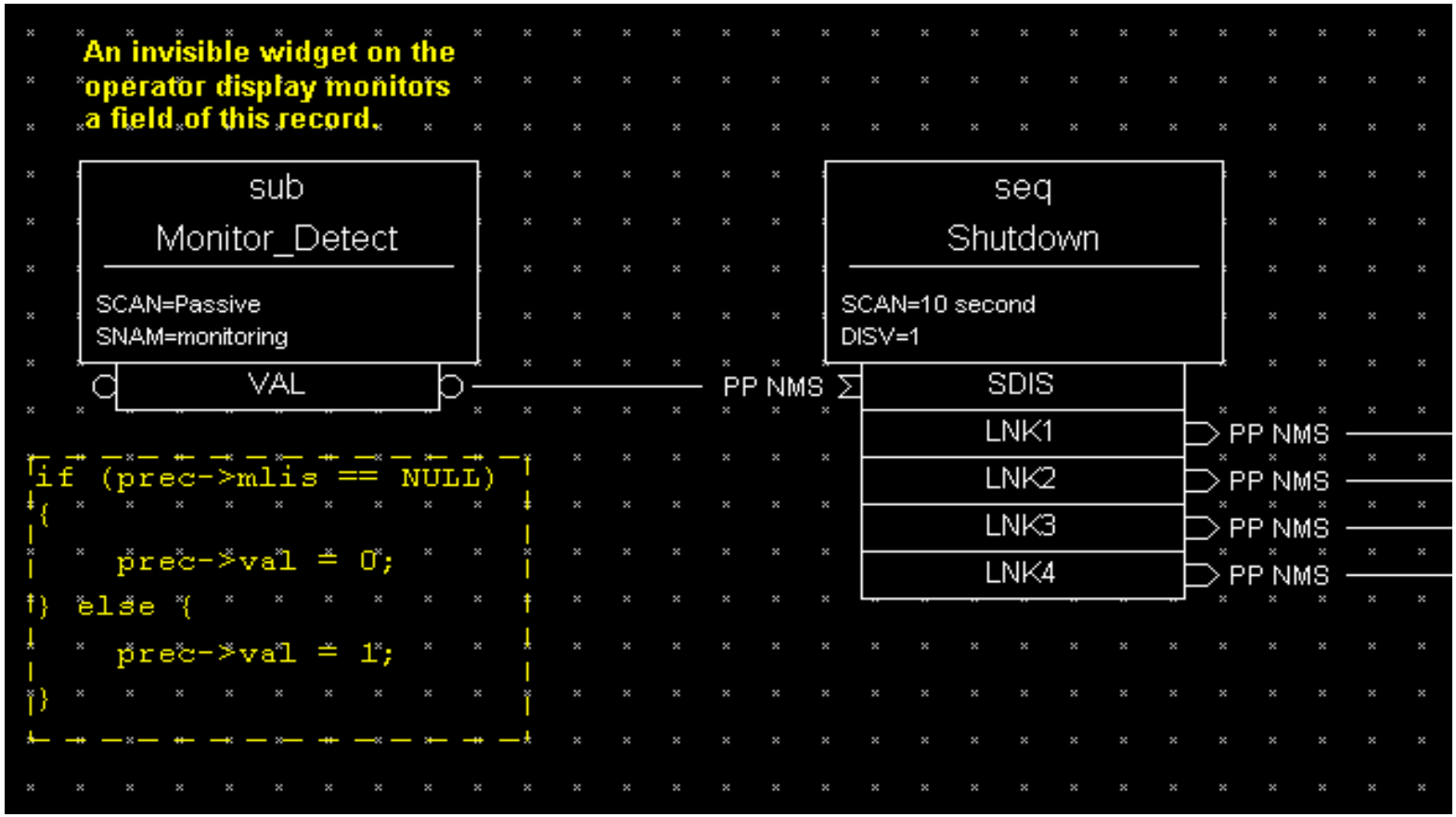
```
record(mbbo,"$(user):PS:Control") {
        field(DTYP,"Raw Soft Channel")
        field(FLNK,"$(user):PS:ControlSQ.VAL PP NMS")
        field(ZRVL,"0x3")           BIT MAP: 0011 -> do LNK1, LNK2
        field(ZRST,"Off")           menu item operator sees
        field(ONVL,"0x5")           BIT MAP: 0101 -> do LNK1, LNK3
        field(ONST,"On)")           menu item operator sees
        field(TWVL,"0xc")           BIT MAP: 1100 -> do LNK3, LNK4
        field(TWST,"Set @ Default") menu item operator sees
}
record(seq,"$(user):PS:ControlSQ") {
        field(SELM,"Mask")
        field(SELL,"$(user):PS:Control.RVAL NPP NMS")
        field(DLY1,"0")
        field(DOL1,"0")
        field(LNK1,"$(user):PS:setCurrent.VAL PP NMS")
        field(DLY2,"2")
        field(DOL2,"0")
        field(LNK2,"$(user):PS:pwrControl.VAL PP NMS")
        field(DLY3,"0")
        field(DOL3,"1")
        field(LNK3,"$(user):PS:pwrControl.VAL PP NMS")
        field(DLY4,"1")
        field(DOL4,"3.75")
        field(LNK4,"$(user):PS:setCurrent.VAL PP NMS)
}
```



Different links in the sequence record are executed for each selection of the mbbo. This allows much functionality to be specified in only two records.

Database Examples

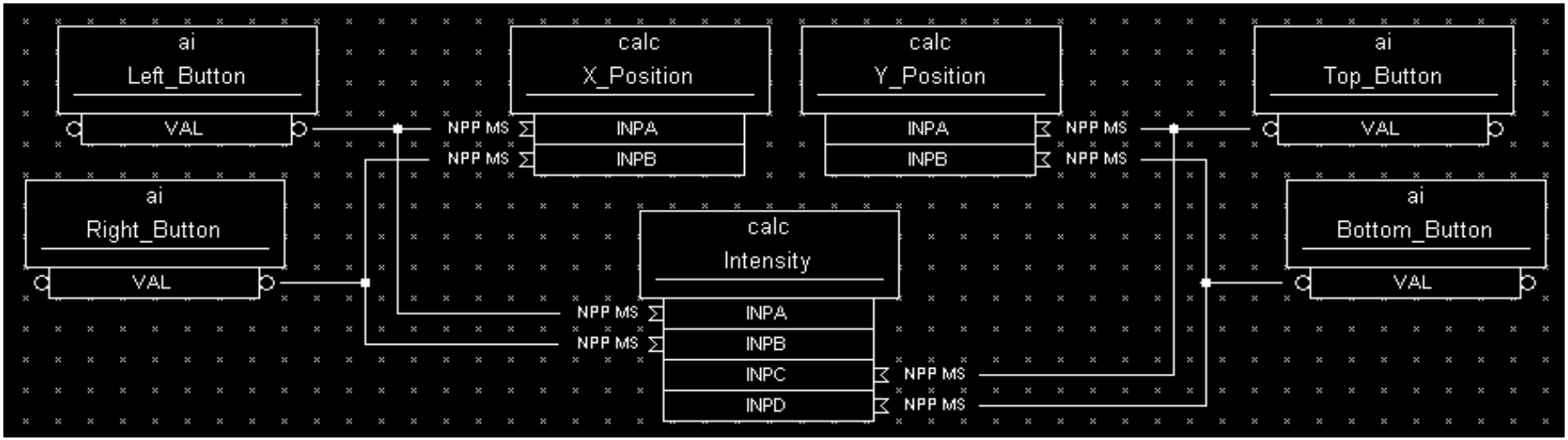
Automatic Shutdown on Logout



If no CA monitor exists on the sub record (i.e. the operator logs out), its MLIS field will be NULL. The subroutine will then set the VAL field to 0, allowing the sequence record to process when this happens.

Database Examples

Quick Prototyping with Standard Records



Custom Record Definition