

EPICS Lecture @ KEK

Record Types and Examples

Takashi Nakamoto
June 25th, 2013

Based on presentation by T. Mooney, APS

Scope

- This lecture:
 - Existing record types and what they can do
 - Record-type documentation
 - Where to look for record types
- Related topics not covered in this lecture:
 - What is a record?
 - *Database – Concepts and linking*
 - How do I connect a record *instance* to a device?
 - set the link field (*Database– Concepts and linking*)
 - How do I connect a record *type* to a device?
 - *Finding and deploying I/O support -- or, if not found...*
 - *Writing device support*
 - How do I write a new record type?
 - *Writing Record Support*

EPICS record types

- Where do record types come from?
 - **EPICS Base (<base>/src/rec)**
 - **General purpose record types**
 - **No record-type specific operator displays or databases**
 - **Documentation in *EPICS Record Reference Manual***
 - **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, databases**
 - **Custom record types can be written by an EPICS developer, and added to an EPICS application.**
 - ***Not in the scope of this lecture***

The Record Reference Manual

- Where is it?
 - http://www.aps.anl.gov/epics/wiki/index.php/RRM_3-14 (wiki)
- What is in it?
 - **Database Concepts (good review)**
 - **Fields common to all records (covered earlier)**
 - **Fields common to many records (covered earlier)**
 - **Record Types – provides a description of the record processing routines for most of the record types *in base*.**
- When would I use it?
 - **Skim through before writing any databases**
 - **Read through before writing any records**
 - **Otherwise, use as reference**

...The Record Reference Manual

- Preface, Chapter 1: Essential background information
 - **Note special meaning of the words *scan*, *process*, *address*, *link*, and *monitor***
- Chapter 2-39: Record reference
 - **Somewhat out of date**
 - **Descriptions of record fields, processing, and useful info for writing device support**
 - **Contains lots of tables like the following:**

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:** tech-talk@aps.anl.gov (for information, send a blank message to listserv@aps.anl.gov)
 - **The soft-support list contains entries like this (among 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:timestamp

...*Collaboration supported records*

- What should I expect to find?
 - **EPICS 3.13-compatible records:**
 - varies with source
 - at minimum: xxxRecord.dbd, xxxRecord.c
 - **EPICS 3.14 and later:**
 - a buildable *module*
 - a statement of requirements (e.g., which version of base)
 - maybe record-type specific displays, databases, programs, etc.

Input Records

- ai - Analog input [\[BASE\]](#)
 - **Read analog value, convert to engineering units, four alarm levels, 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 (16,000 bytes max for CA in EPICS 3.13, governed by EPICS_MAX_ARRAY_BYTES environment variable in EPICS 3.14)**

Input Records (cont..)

- *scaler* [[synApps](#)]
 - **Controls a bank of counters**
- *mca* – *multichannel analyzer* [[synApps](#)]
 - **Supports multichannel analyzers, multichannel scalers, and other array-input hardware**
- *Mbbi32Direct* – *32-bit mbbiDirect* [[ORNL](#)]
 - **Read an unsigned long and map each bit to a field (32 BI records in one)**
- *timestamp* [[SLAC](#)]
 - **Exports its timestamp as a string**

Algorithms/Control Records - Calc

- calc - run-time expression evaluation [\[BASE\]](#)
 - 12 input links, user specified “calc expression” (algebraic, trig, relational, Boolean, Logical, “?”), four alarm levels
 - Sample expressions:
 - 0 read: “<calc_record>.VAL = 0”
 - A note ‘A’ refers to <calc_record>.A
 - A+B
 - sin(a)
 - (A+B) < (C+D) ?E : F+L+10
- calcout – calc variant [\[BASE\]](#)
 - Conditional output link, separate 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:%.21f", A+4)` -> **"SET:VOLT:5.00"**
 - Additional output-link option: "Never"

- *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:
 - **A: 2 read: "<transform>.A = 2"**
 - **B: A+1+C uses new value of 'A', old value of 'C'**

Algorithms/Control Records - Array

- compress [\[BASE\]](#)
 - **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 a waveform.**

Algorithms/Control Records – List

- dfanout – Data fanout [\[BASE\]](#)
 - **Writes a single value to eight output links**
- 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**

Algorithms/Control Records - Loop

- `sscan` [[synApps](#)]
 - **Six “positioners”, four “detector triggers”, 74 “detectors”.**
 - **Systematically sets conditions, triggers detectors, and acquires data into arrays.**
 - **Database detects completion and drives scan to next step.**

Algorithms/Control Records - Subroutine

- sub – Subroutine [[BASE](#)]
 - **12 input links, user provided subroutine**
- genSub – sub variant [[synApps](#)]
 - **Multiple inputs and outputs**
 - **Handles arrays and structures**

Algorithms/Control Records - Other

- event [\[BASE\]](#)
 - **Posts a “soft” event which may trigger other records to process.**
 - **Simulation mode**
- Permissive – handshake [\[BASE\]](#)
 - **Implements a client-server handshake**
- state – string state value [\[BASE\]](#)
 - **Implements a string, for client-server communication**
- EPID [\[synApps\]](#)
 - **Extended PID**
 - **Useful also to run faster than record processing if the right driver is used**

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**
- 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 [\[BASE\]](#)
 - **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**
- *mbbo32Direct* - *mbbo* variant [\[ORNL\]](#)
 - **16 settable bit fields that get written as a long integer, closed_loop mode, INVALID alarm action, simulation mode**
- *motor* [\[synApps\]](#)
 - **Controls stepper and servo motors**
- stringout [\[BASE\]](#)
 - **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**
- *Many others that are site-specific*

Which record is right for ...

- “operator entered” soft parameters
 - AO has DRVH, DRVL, OROC, closed loop
 - MBBO provides enumerated options which can be converted to constants (DTYP = Raw Soft Channel)
 - Normally one does not use input records for this purpose

- Multiple output actions
 - **Sequence record can have a different data source for each output link vs. the dfanout record which “fans out” a single source to multiple links**

- Different output actions based on an operator selection
 - **CALCOUT records that conditionally process sequence records**
 - **MBBO (Soft Raw Channel) forward linked to a single sequence record in “masked” mode. Mask is provided in MBBO for each state.**

Defining the Database

- How does an IOC know what record *types* and device support options are available ?
 - **Record types, device support options, enumerated menus, and other configuration options are defined in “database definition files” (.dbd)**
 - **During the IOC booting process, one or more .dbd files are loaded**
 - **.dbd files are created on the workstation to include the desired information for that IOC.**
- 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**
 - **.db files are created on the workstation to include the desired information for that IOC.**

Database File Formats

- Typical content of a database definition file (.dbd)

```

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)

```

Database File Formats

- Typical content of database definition file (.dbd):

```
recordtype(ai) {
include "dbCommon.dbd"
field(VAL,DBF_DOUBLE) {
prompt("Current EGU Value")
promptgroup(GUI_INPUTS)
asl(ASL0)
pp(TRUE)
}
field(INP,DBF_INLINK) {
prompt("Input Specification")
promptgroup(GUI_INPUTS)
interest(1)
}
field(PREC,DBF_SHORT) {
prompt("Display Precision")
promptgroup(GUI_DISPLAY)
interest(1)
}
```

```
menu(scalerCNT) {
choice(scalerCNT_Done,"Done")
choice(scalerCNT_Count,"Count") }
....
field(CNT,DBF_MENU) {
prompt("Count")
special(SPC_MOD) menu(scalerCNT)
pp(TRUE) interest(1)
}
}
device(ao,CONSTANT,devAoSoftRaw,
"Raw Soft Channel")
driver(drvVxi)
```


Database File Formats

- A typical database file (.db)

```

record(bo,"$(user):gunOnC") {
    field(DESC,"Controls e-gun")
}
record(bo,"$(user):gunOnC") {
    field(DESC,"Controls e-gun")
    field(DTYP,"Soft Channel")
    field(ZNAM,"Beam Off")
    field(ONAM,"Beam On")
}
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 NPP NMS")
}
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 NPP NMS")
    field(INPB,"$(user):rampM.VAL NPP NMS")
    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

- Part of a typical startup script (st.cmd)

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

One or more database definition files (.dbd) must be loaded first.

Any record type specified in the database files must have been defined in the definition file

Macros (variables) within the database files (e.g. \$(user)) can be specified at boot time. This allows the same database to be loaded with different names or channel assignments.

Creating Database Files

- Since the database file is a simple ascii file, it can be generated by numerous applications ... as long as the syntax is correct.
 - **Text editor**
 - **Script**
 - **Relational Database Tool**
 - **EPICS-aware Database Configuration Tools:**
 - *VDCT (recommended for new designs)*
 - *CAPFAST (a schematic entry application)*
 - *JDCT (not graphical)*
 - *GDCT (no longer supported)*
- An EPICS-aware tool will read the .dbd file (library provided) and provide menu selections of enumerated fields. It may also detect database errors prior to the boot process
- A graphical tool is helpful for complex databases

Macro substitutions

- EPICS features simple string substitution macros
 - **\$(macro) can be used in .db files**
 - **This allows db files to function as templates (e.g. use the same db file for all vacuum sectors, just with different names (and possibly other parameters))**
- Database with \$(macro) cannot be loaded – all macros need to be expanded
- This can be done in st.cmd or by means of a separate substitutions file
 - Creating a new EPICS application using `example` template will provide an example substitution file
- For more complex macro handling, there is an EPICS extension called **msi**

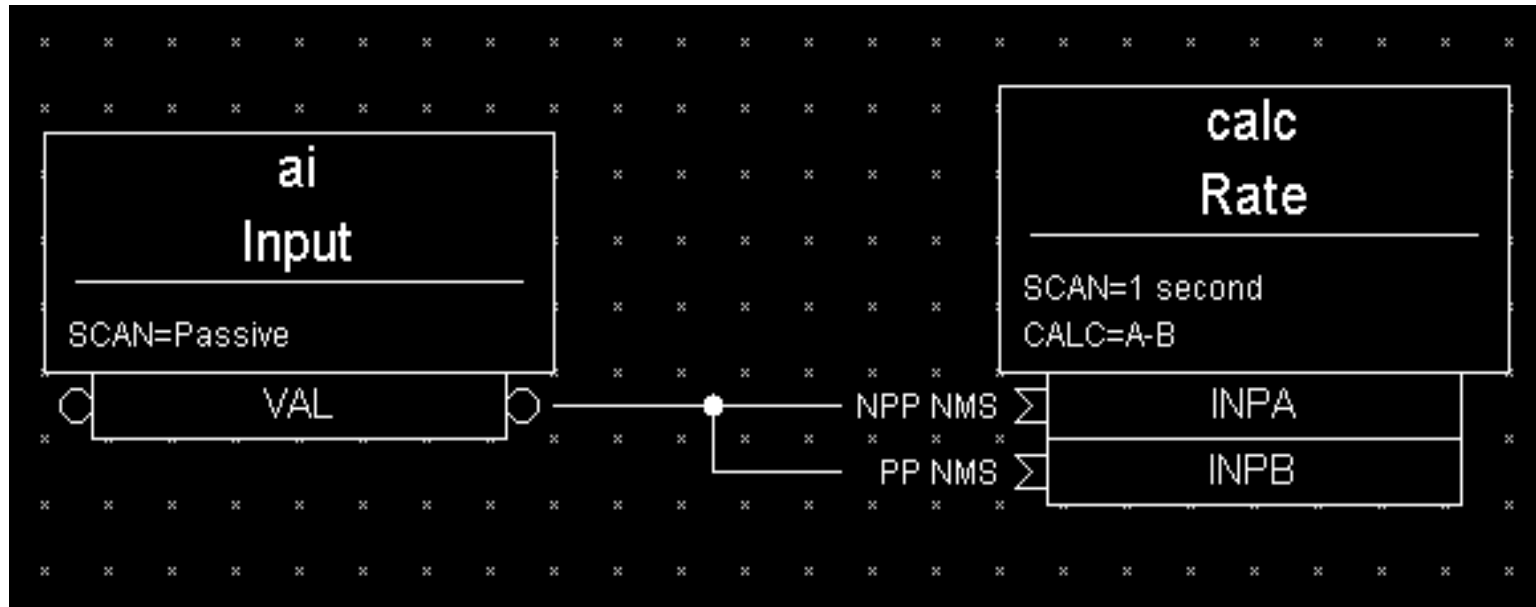
Analog input example

```
ai
Input
-----
SCAN=I/O Intr
DTYP=Analog Input
EGU=Volt
HOPR=100
LOPR=0
HIGH=80
HIHI=90
HHSV=MAJOR
HSV=MINOR
HYST=0.1
MDEL=0.1
ADEL=1
```

An analog input record reading a voltage in volts in an operating range from zero volts to 100 volts. The limit for a minor alarm is 80 volts and the limit for a major alarm is 90 volts. There is a hysteresis associated with the alarm limits and a deadband for reporting value changes to monitors and archivers.

Database Examples

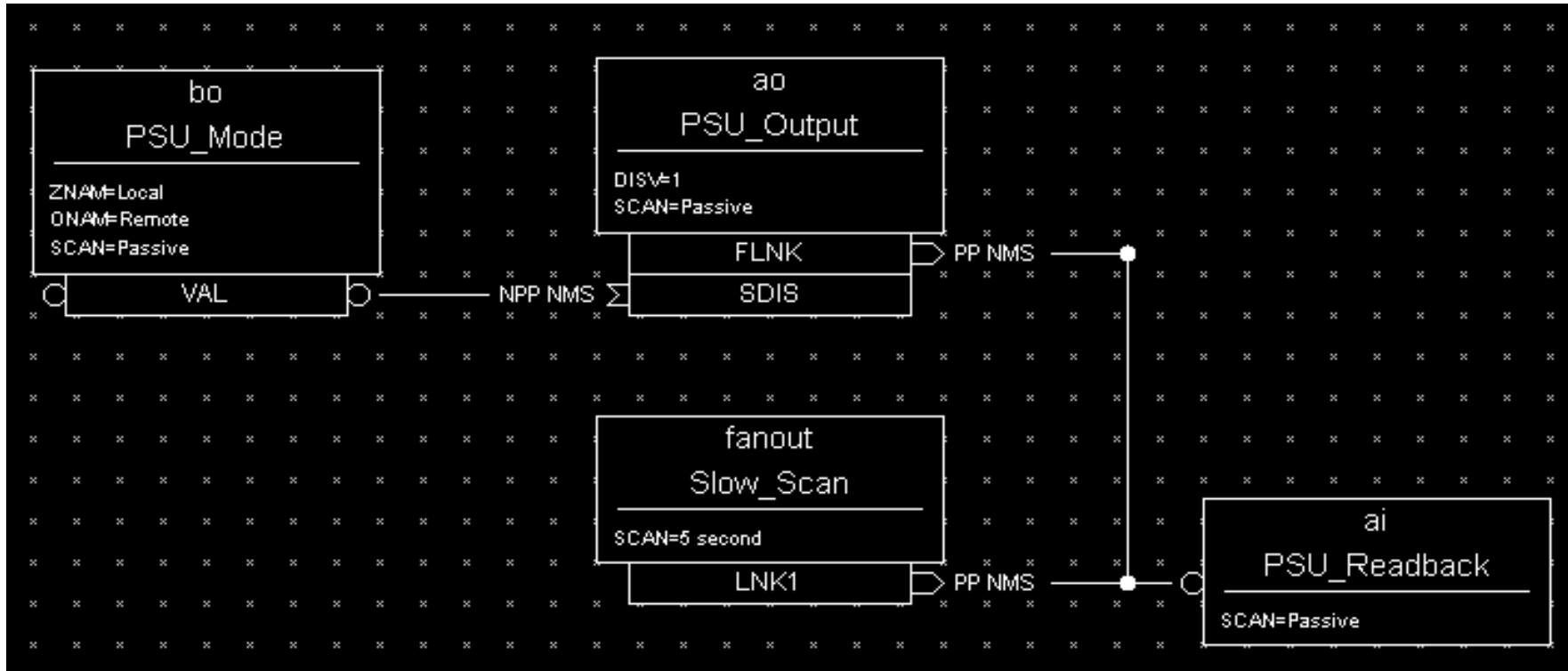
Calculating “Rate-of-Change” of an Input



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.

Database Examples

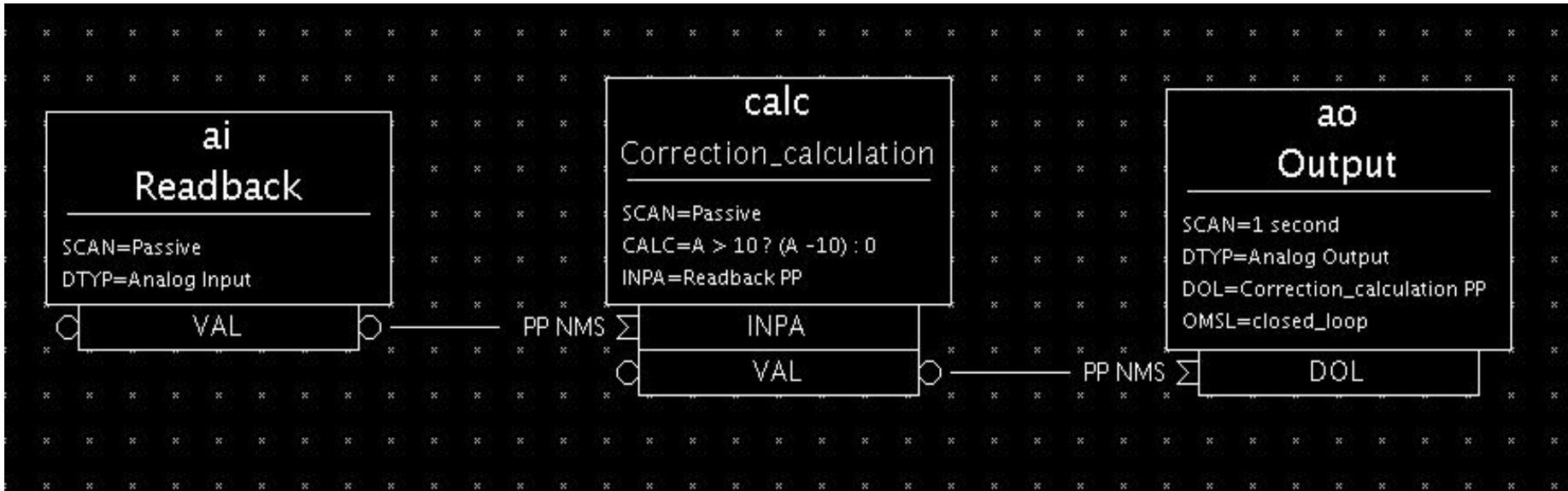
Slow Periodic Scan with Fast Change Response



AI record gets processed every 5 seconds AND when the associated AO record is changed. This provides immediate response to a change even though the desired scan rate is very slow. Changes to the power supply settings are inhibited by the BI record, which could represent a Local/Remote switch.

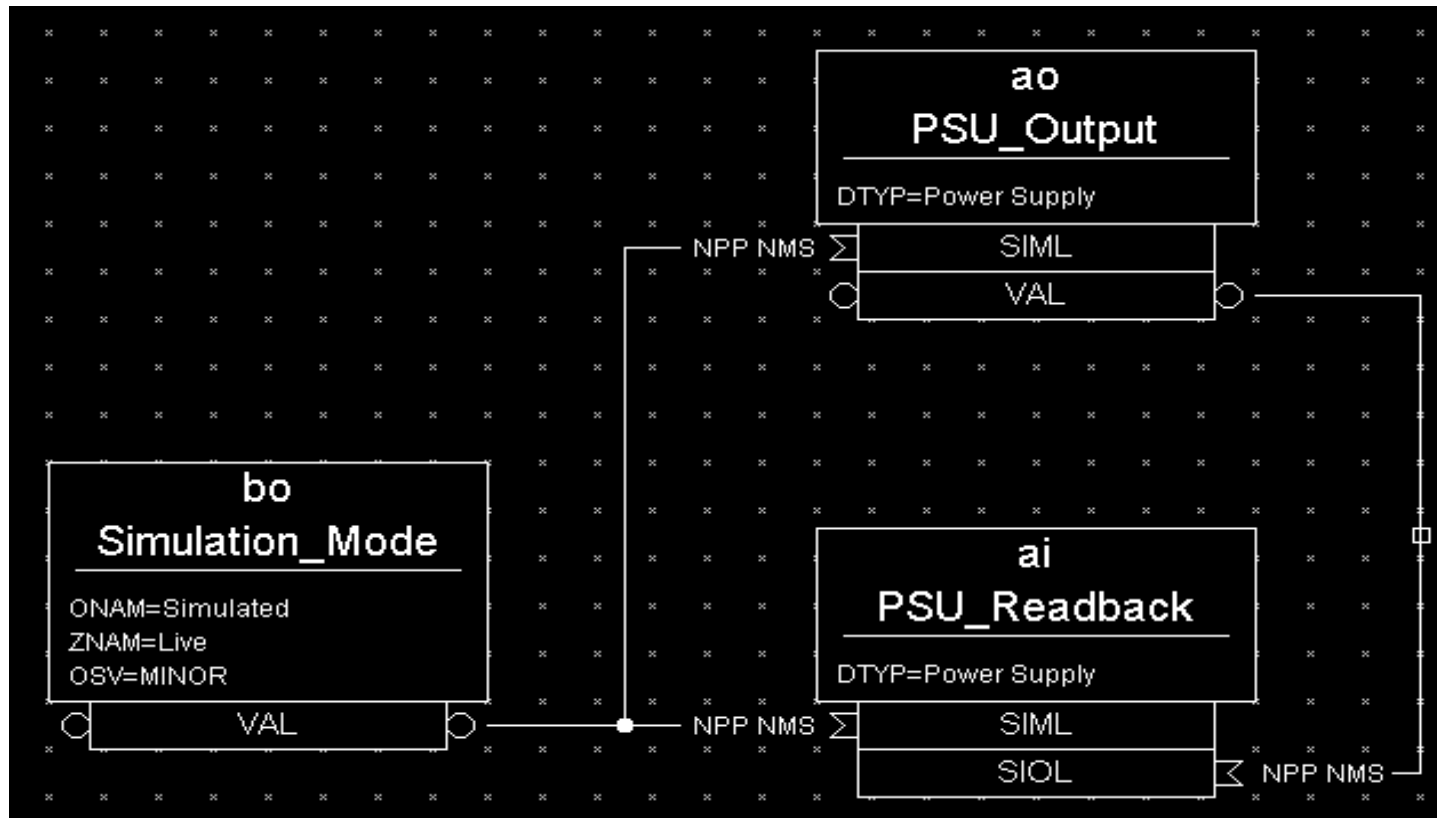
Database Examples

Closed loop control



An analog output record triggers every second and requests the correction calculation from the calc record. The calc record requests the readback value, calculates the correction and the analog output finishes it's processing and outputs the correction.

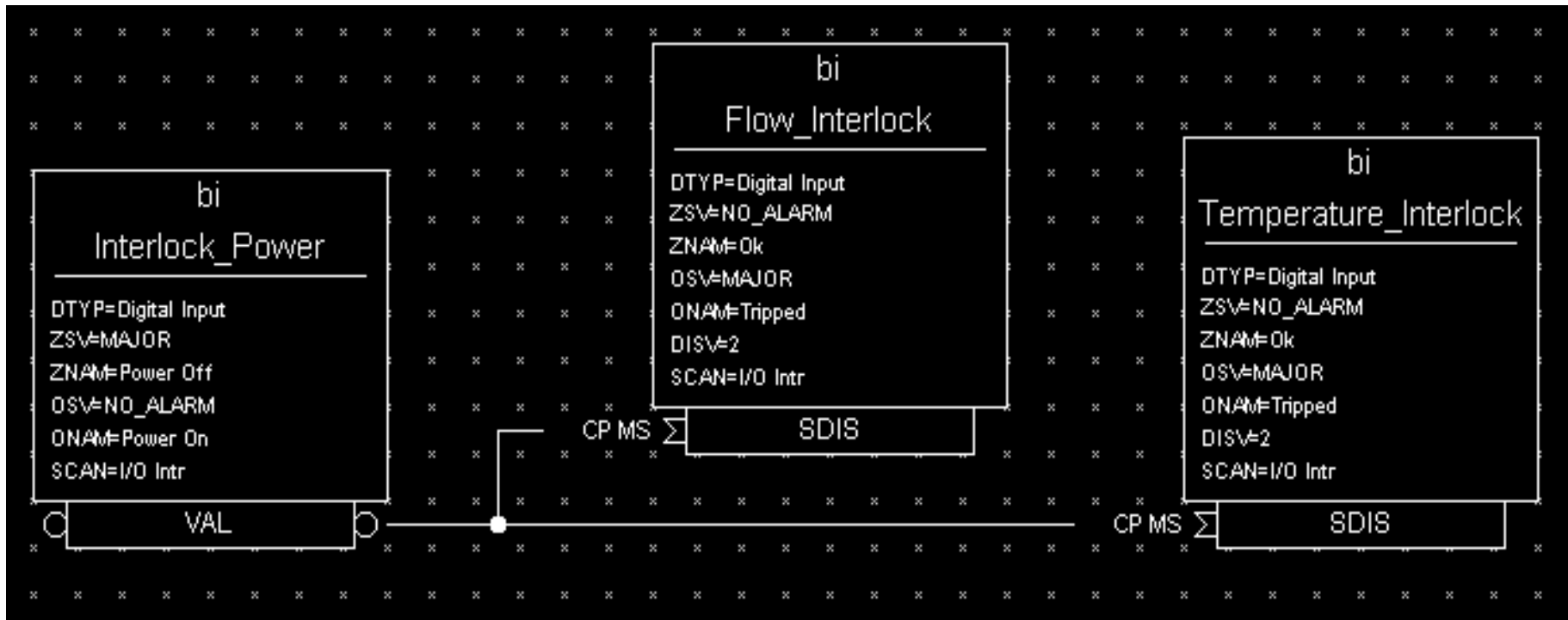
Simulation Mode



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

Database Examples

Maximize Severity



If chassis is powered off, Temp Trip and Flow Trip indicate Normal. Force these PVs into an alarm state by specifying **.SDIS** with **.MS** (maximize severity) to the Chassis On record. Set **.DISV** (disable value) to 2 so processing will never be disabled.

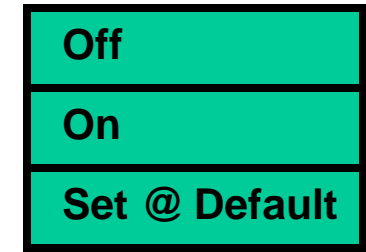
Database Examples

Different Actions Based on Operator Selection

```

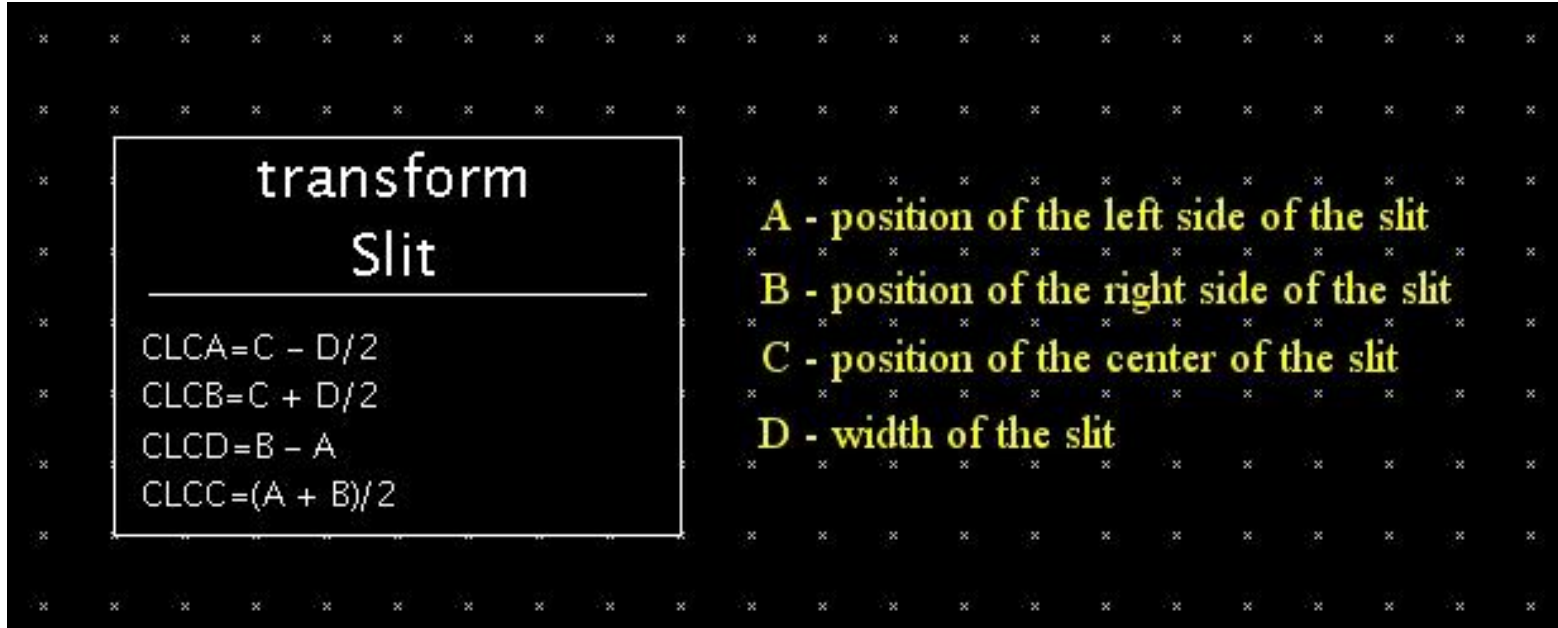
record(mbbo,"$(user):PS:Control") {
  field(DTYP,"Raw Soft Channel")
  field(FLNK,"$(user):PS:ControlSQ.VAL PP NMS")
  field(ZRVL,"0x3") BIT MAP: 0000000000000011 -> do LNK1, LNK2
  field(ZRST,"Off") menu item operator sees
  field(ONVL,"0x5") BIT MAP: 0000000000000101 -> do LNK1, LNK3
  field(ONST,"On") menu item operator sees
  field(TWVL,"0xc") BIT MAP: 0000000000001100 -> 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.

Bidirectional transformation - slit control



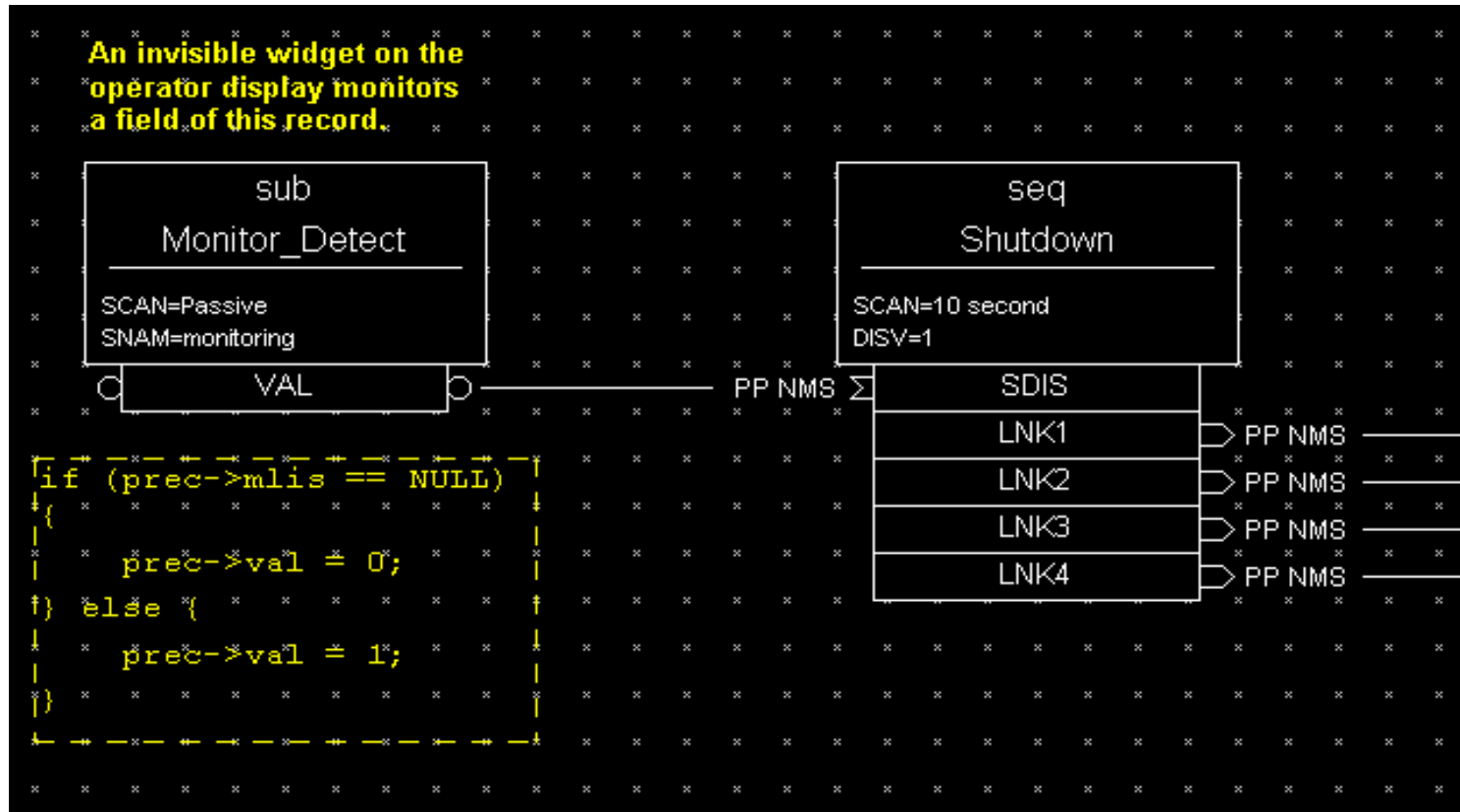
```
transform
Slit
-----
CLCA=C - D/2
CLCB=C + D/2
CLCD=B - A
CLCC=(A + B)/2
```

A - position of the left side of the slit
B - position of the right side of the slit
C - position of the center of the slit
D - width of the slit

A command moves the left side of the slit (A changes) and causes the record to process, A will not be recalculated because its value is new, the right side of the slit will remain still (B is unnecessarily recalculated from old values of C and D), the position of the slit center (C) will change as expected, and the slit width (D) will change as expected. Thus, all four fields will contain consistent information about the two degrees of freedom controlled by the record.

Database Examples

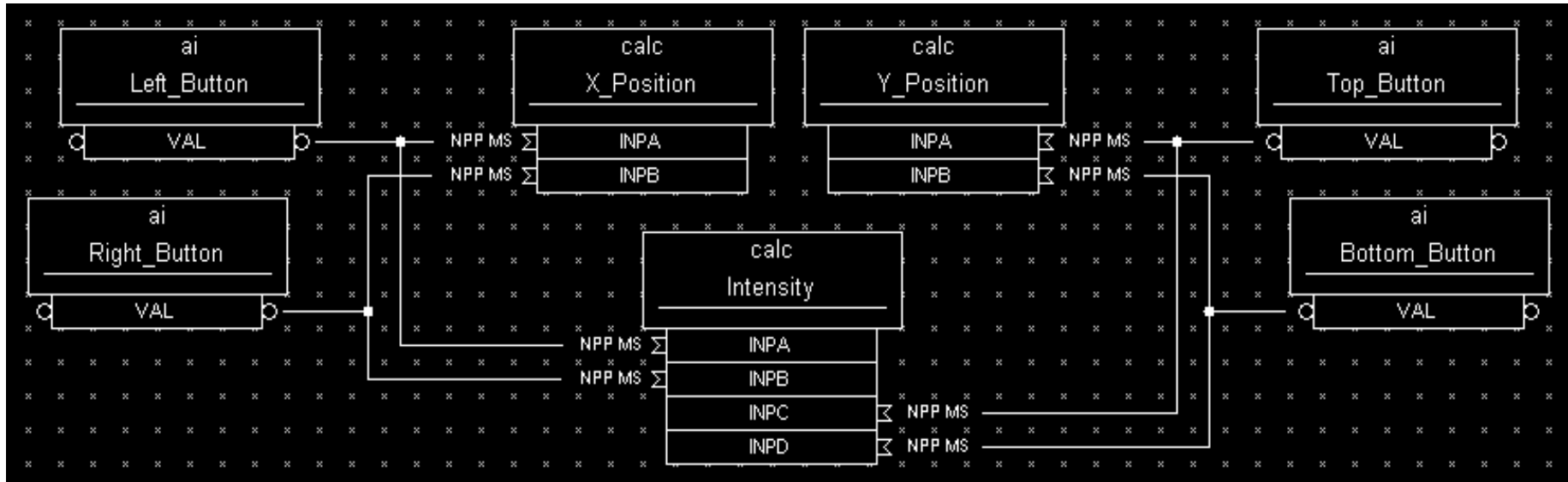
Automatic Shutdown on Logout



If no CA monitor exists on the SUB record (i.e. the operator logs out), MLIS will be NULL. The subroutine will then set the .VAL field to 0, causing the sequence record to process.

Database Examples

Quick Prototyping with Standard Records



Custom Record Definition

