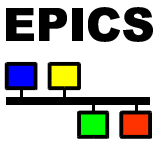
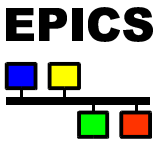


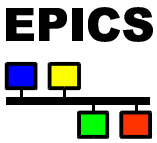
EPICS Database II

Bob Dalesio
LANL



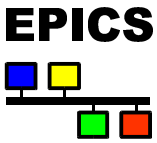
The Record Reference Manual

- 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 each record type.



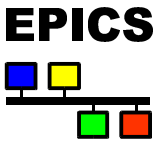
Input Records

- Analog in
  - Read analog value, convert to engineering units, four alarm levels, simulation mode
- Binary in
  - Single bit, two states, assign strings to each state, alarm on either state or change of state, simulation mode
- Multi-bit binary in
  - Multiple bit, sixteen states, assign input value for each state, assign strings to each state, assign alarm level to each state, simulation mode
- Multi-bit binary in Direct
  - Read an unsigned short and map each bit to a field (16 BI records in one)
- String in
  - 40 character (max) ascii string, simulation mode



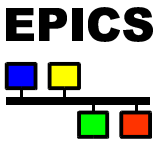
Input Records (cont..)

- Long in
  - Long integer, four alarm levels, simulation mode
- Pulse counter
  - Written to support a Mizar 8310 timing module
- Waveform
  - Configurable data type and array length (16,000 bytes max for CA)



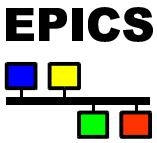
Algorithms/Control Records

- Calc
  - 12 input links, user specified “calc expression” (algebraic, trig, relational, Boolean, Logical, “?”), four alarm levels
  - Examples:
    - `(A-B)*C`
    - `((A<<2)&B)|C`
    - `(A+B)<(C+D)?E:F+L+10`
- Calcout
  - Same as CALC with a *conditional* output link, separate output CALC expression, output delay, and output event
  - Options : "Every Time”, "On Change", "When Zero", "When Non-zero", "Transition To Zero", "Transition To Non-zero"
- PID (CPID, EPID)
  - Proportional/Integral/Derivative Control



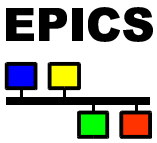
Algorithms/Control Records (cont..)

- Select
  - 12 input links, four select options (specified, highest, lowest, median), four alarm levels
- Compress
  - Input link can be scalar or array
  - Algorithms include N to 1 compression (highest, lowest, or average), circular buffer of scalar input
- Subroutine
  - 12 input links, user provided subroutine, four alarm levels
- Fanout
  - Forward links to six other records
- Dfanout
  - Writes a single source of data to eight output links



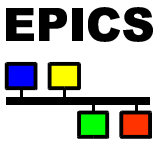
Algorithms/Control Records (cont..)

- Sequence
  - Ten "Input link to Output link" pairs with a specified delay between link execution. Subsets of the ten pairs can be executed by specifying a mask or a specific link pair (Select options include ALL, SPECIFIED, MASK).
- Event
  - Posts a "soft" event which may trigger other records to process, simulation mode
- Scan
  - Four "positioners", fifteen "detectors". A scan steps through values of the positioners and records the detector values at each point. All arrays are accumulated within the record and posted when the scan is complete
- SubArray
  - Extracts a sub-array from a waveform



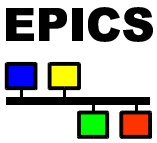
Output Records

- Analog out
  - 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
- Binary out
  - 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
- Multi-bit binary out
  - 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
- Multi-bit binary out direct
  - 16 settable bit fields that get written as a short integer to the hardware, closed_loop mode, INVALID alarm action simulation mode



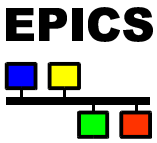
Output Records (cont..)

- Stepper motor
  - Position control, retry, speed, ramps, etc
- Pulse delay
  - Written to support a Mizar 8310 timing module
- Pulse Train
  - Written to support a Mizar 8310 timing module
- Long out
  - Write long integer value, four alarm levels, closed_loop mode, INVALID alarm action, simulation mode
- String out
  - Write a character string (40 max), closed_loop mode, INVALID alarm action, simulation mode



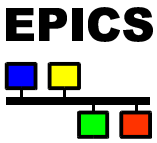
Examples of Custom Records

- RF Amplitude Measurements
  - Sample time, measurement in watts and db, waveform acquired through sweeping sample time
- Beam Position Monitor record
  - Four voltage inputs, numerous calibration constants, X-Y-I outputs, waveforms for each input
- Multi-Channel Analyzer record
  - Controls and acquires data from a multichannel analyzer (MCA). Currently, there are 3 supported multichannel analyzers.
- 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)
- Multiple output actions?
  - Sequence record can have a different data source for each output link vs. the dfanout record which fanouts 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.



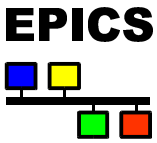
Defining the Database

- 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.

A typical database definition (.dbd) file

```
recordtype(ai) {
        field(NAME,DBF_STRING) {
                prompt("Record Name")
                special(SPC_NOMOD)
                size(29)
        }
        field(DESC,DBF_STRING) {
                prompt("Descriptor")
                promptgroup(GUI_COMMON)
                size(29)
        }
        field(ASG,DBF_STRING) {
                prompt("Access Security Group")
                promptgroup(GUI_COMMON)
                special(SPC_AS)
                size(29)
        }
        field(SCAN,DBF_MENU) {
                prompt("Scan Mechanism")
                promptgroup(GUI_SCAN)
                special(SPC_SCAN)
                menu(menuScan)
                interest(1)
        }
…
recordtype(ao) {  …
recordtype(bi) { …
recordtype(bo) { …
recordtype(calc) { ...
```

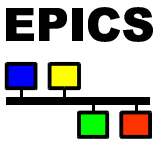
```
menu(menuPriority) {
  choice(menuPriorityLOW,"LOW")
  choice(menuPriorityMEDIUM,"MEDIUM")
  choice(menuPriorityHIGH,"HIGH")
}
menu(menuScan) {
  choice(menuScanPassive,"Passive")
  choice(menuScanEvent,"Event")
  choice(menuScanI_O_Intr,"I/O Intr")
  choice(menuScan10_second,"10 second")
  choice(menuScan5_second,"5 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)
```



A typical database (.db) file

```
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,"50")
    field(HIHI,"180")
    field(LOLO,"130")
    field(HIGH,"160")
    field(LOW,"140")
    field(HHSV,"MAJOR")
    field(HSV,"MINOR")
    field(LLSV,"MAJOR")
    field(LSV,"MINOR")
}
```

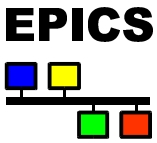


Loading Database Files into the IOC

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

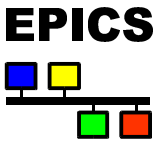
```
dbLoadDatabase("../../dbd/linacApp.dbd")
dbLoadRecords("../../db/xxLinacSim.db","user=studnt1")
iocInit
```

- 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.



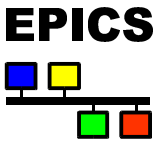
Creating Database Files

- 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
    - JDCT
    - VDCT
    - CAPFAST (a schematic entry application)
- 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 to document and support complex databases



Steps to Creating and Loading a New Database File

- Create the database file in an appropriate Db directory
- Edit the Makefile so this .db file is managed properly
- Run *make*
- Edit the IOC's startup script (st.cmd) to load the new database
  - dbLoadRecords(_____)
- Restart the IOC

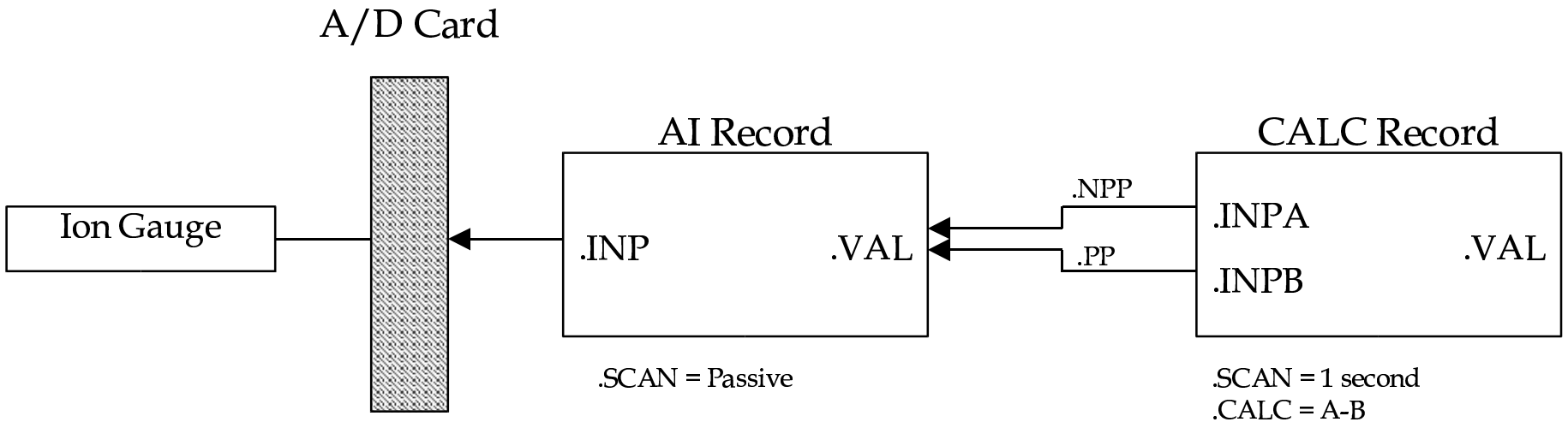


JDCT and VDCT

- Run *jdct* or *vdct* (usually in a Db directory)
- Open one or more .dbd files (usually in the directory '../../dbd') to define available record types, menus, available device options, etc.)
- Create, copy, edit record instances
- Save the .db file
- Run *make* to install the .db into another directory
  - You must modify the Makefile each time you add a new database file

Database Examples

Calculating "Rate-of-Change" of an Input

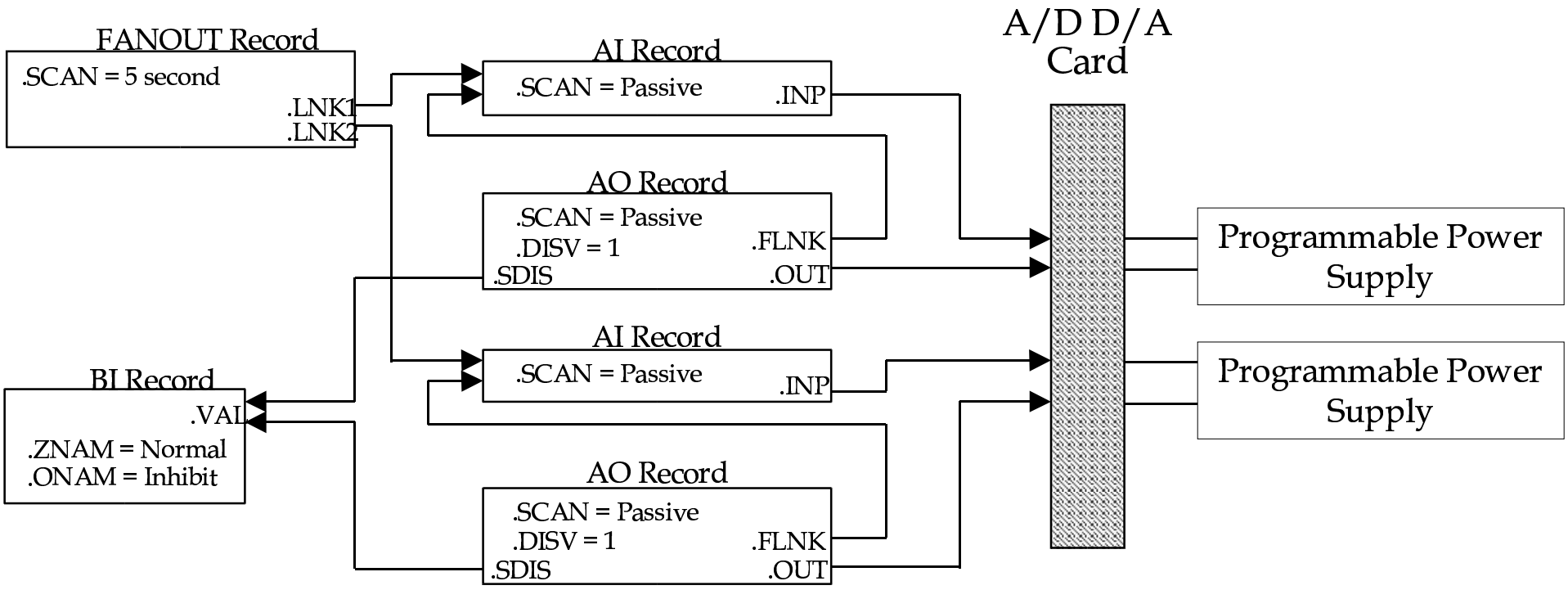


INPA fetches data that is 1 second old because it does not request processing of the AI record. INPB fetches current data because it requests the AI record to process. The subtraction of these two values reflects the 'rate of change' (difference/sec) of the pressure reading.

* The direction of the arrows indicates where a link points to, not necessarily the direction of the data flow.

Database Examples

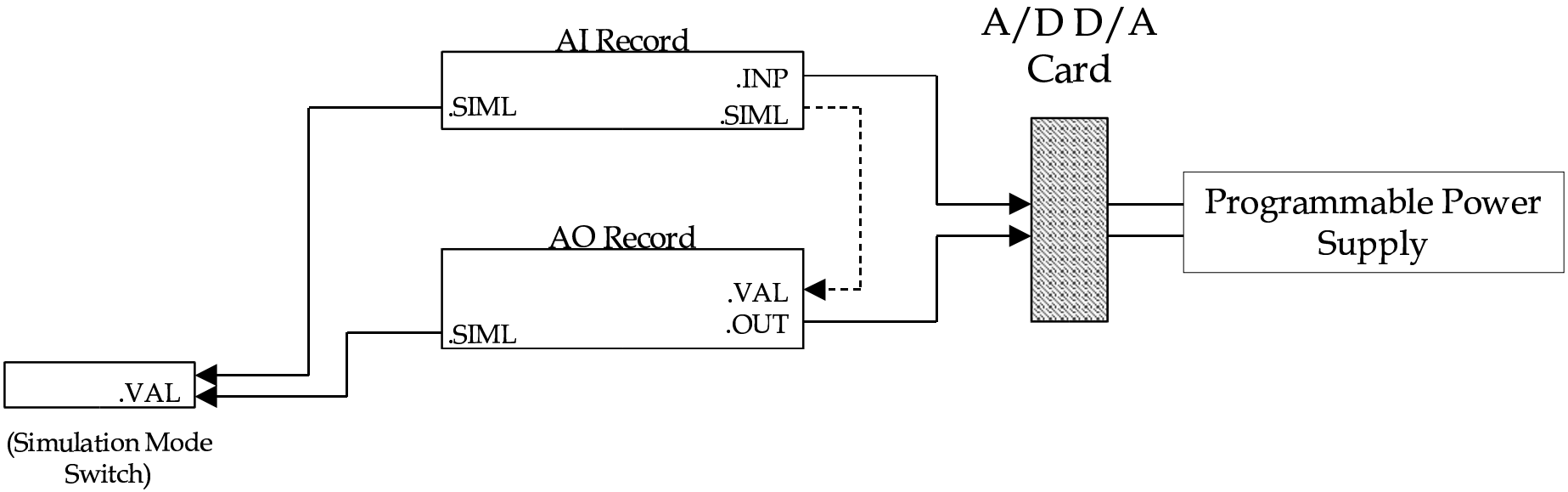
Slow Periodic Scan with Fast Change Response



AI records get processed every 5 seconds AND when the associated AO record is changed. This provides immediate response to a change even though the normal 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

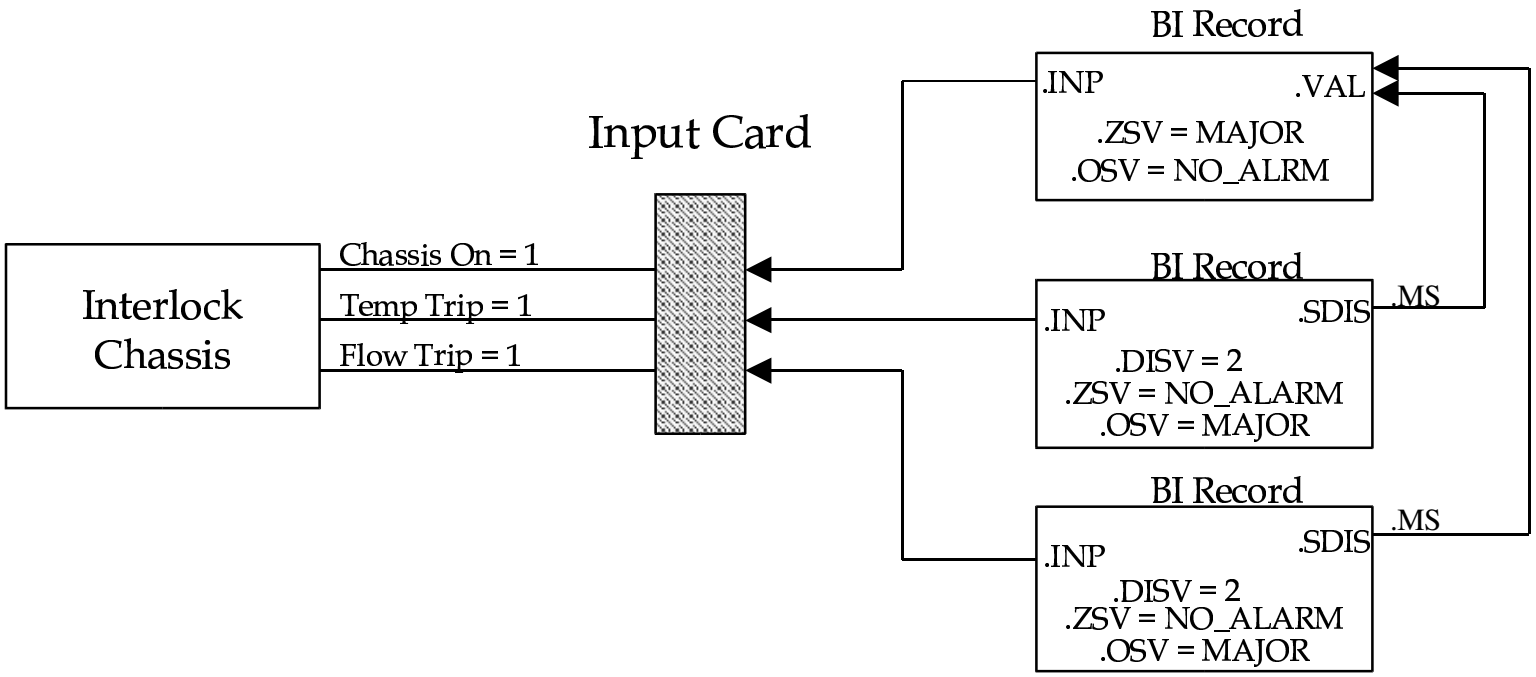
Simulation Mode



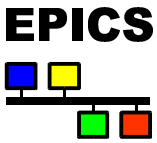
When in simulation mode, the AO record does not call device support and the AI record fetches its input from the AO record.

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

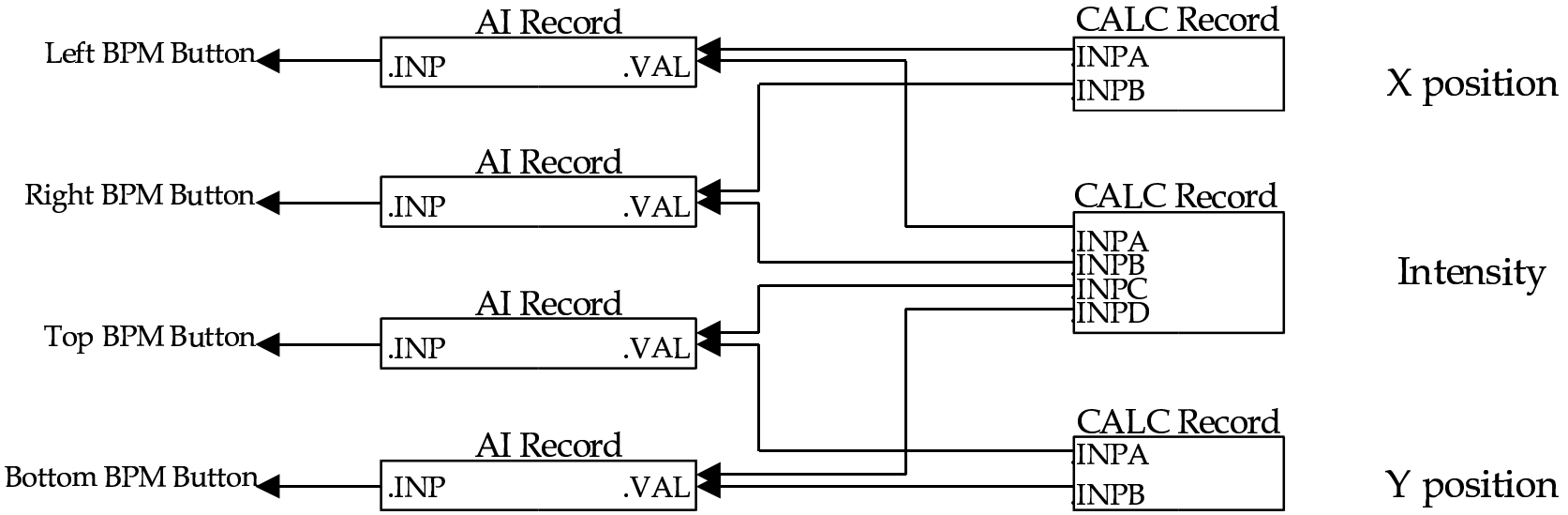
Different Actions Based on Operator Selection

```
record(mbbo,"$(user):PS:Control") {
        field(DTYP,"Raw Soft Channel")
        field(FLNK,"$(user):PS:ControlSQ.VAL PP NMS")
        field(ZRVL,”0x3")
        field(ONVL,”0x5")
        field(TWVL,"0xc")
        field(ZRST,"Off")
        field(ONST,"On)")
        field(TWST,"Set @ Default")
}
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)
}
```

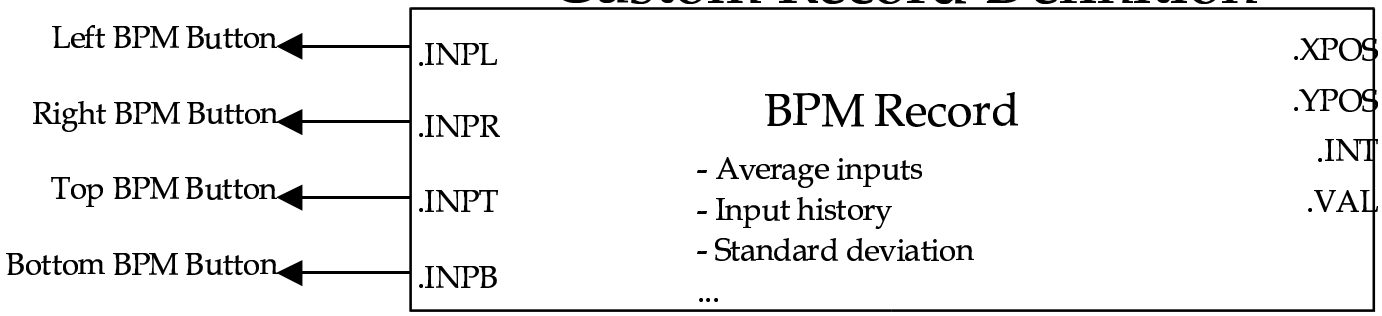
Different links in the sequence record are executed for each selection of the mbbo. This allows much functionality to be specified in only two records.

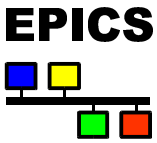
Database Examples

Quick Prototyping with Standard Records



Custom Record Definition

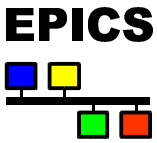




Database - Analog Input

- Used to read an analog transducer.
- Conversions for linear transducers accomplished using LINR = Linear and placing the full range of the conversion in EGUF and the low end of the range in EGUL
- A weighted average is built in for smoothing in the form

  VAL(new) * (1-SMOO) + VAL(previous) * SMOO
- Break point tables used for non-linear conversions or tables.



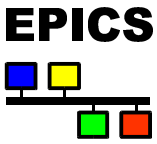
Analog Input – Breakpoint Tables

- To add a break point table enter the breakpoints into the .dbd file with counts in the first column and engineering units in the second column

```
breaktable (IGtable){
        0, 0,
        2048, .00000000001,
        2068, .000000000011,
              :
        4077, .085,
        4097, .1
}
```

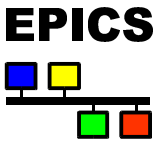
- Add the breakpoint to a local copy of the conversion file menuConvert.dbd

```
menu(menuConvert) {
        choice(menuConvertNO_CONVERSION,"NO CONVERSION")
        choice(menuConvertLINEAR,"LINEAR")
        choice(menuConverttypeKdegF,"typeKdegF")
        choice(menuConverttypeKdegC,"typeKdegC")
        choice(menuConverttypeJdegF,"typeJdegF")
        choice(menuConverttypeJdegC,"typeJdegC")
            :
        choice(menuConverttypeSdegC,"typeSdegC")
        choice(menuConverttypeIG,"IGtable")
}
```



Database – Analog Output

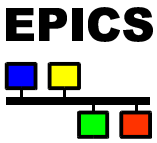
- VAL gives the output value expressed in engineering units.
- If OIF is Incremental, then the OVAL will be set to be the previous OVAL + VAL.
- If OIF is Full, VAL is only changed by at most OROC on every scan as long as OROC is a non-zero number.
- After a new value for OVAL is computed, it is limited by DRVH and DRVL. These are software drive limits on the output.
- Also note the fields that are used by all output records for performing closed loop control and reacting to errors on the values used to determine the new outputs.
- OVAL is converted to an integer and sent to the hardware.



Database - Binary Output

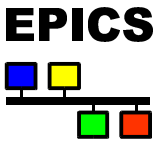
- The binary output provides a momentary output if the HIGH field is set to a non-zero value. If HIGH is 0.5 and the VAL is set to 1, 0.5 seconds later the VAL will be reset to 0 and the record is processed.

- Also note the fields that are used by all output records for performing closed loop control and reacting to errors on the values used to determine the new outputs.



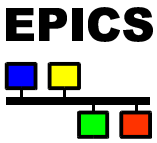
Database – Calculation

- 12 inputs that are specified in the fields INPA through INPL
- The values are fetched and stored in the CALC record in the fields A though L. These values can be entered directly into the A through L fields if their respective INP<x> is not defined.
- Algebraic Operators: ABS, SQR, MIN, MAX, CEIL, FLOOR, LOG, LOGE, EXP, ^, **, +, -, *, /, %, NOT
- Trigonometric Operators: SIN, SINH, ASIN, COS, COSH, ACOS, TAN, TANH, ATAN
- Relational Operators: >=, >, <=, <, #, =
- Logical Operators: &&, ||, ! (Not)
- Bit-wise Operators: |, &, OR, AND, XOR, ~, <<, >>
- Parentheses and nested parens are supported
- The C '?' operator is supported for conditionals (expr)?(true):(false)
- Warning: Only 36 characters are allowed for the expression
- The Infix expression is converted to byte code reverse polish for execution
- When the expression is changed during operation, a new postfix byte code is generated



Database – Calculation Output

- Like a calculation in all ways except that it includes an OUT field for writing the result.
- The OOPT field controls when a value is written and includes the options: every time the record is scanned, when the value changes, when the value is zero, when the value is not zero, when the value transitions to zero, when the value transitions to not zero.
- When DOPT is Calc, then the result of the initial expression is used for output. If the DOPT field is set to OCAL, then the result of the OCAL expression is used for output. This allows the CALC to be used to determine if a value should be output and OCAL then to be the value to output.
- It the OEVT field is non-zero and the OOPT permits the output, a post_event is called with the event number from the OEVT field.
- To delay before the output is sent, set the ODLY field to the second for wait until the output is set. The record has PACT set during this period and further sets DLYA to true while waiting.
- Finally, the record features many status values: those of the link status for each INPx field, CALC valid, OCAL valid and the output address valid.



Database – Compression Record

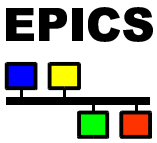
- The compression record is used to compress arrays or to keep some historical data from an scalar. It runs as a circular buffer, a successive average or to compress values.
- When ALG is Circular Buffer, it reads the scalar referenced by INP and places the value into the next element of the NSAM element array.
- When ALG is Average, INP is treated as an array (of possibly 1 element). Each time the record is processed the new array is added to the sum. On the Nth read, the array is averaged and placed into the value of this record. (The Record Reference Manual is incorrect on this record)
- When ALG is any of the N to 1 algorithms (High, Low, Average), and the INP is an array, the INP array is read and values are discarded until one of them is lower that the filter value ILIL or higher than the IHIL. If ILIL and IHIL are 0, no checking is done. The array is then compressed according to the algorithm - every N samples becomes one sample in the new array and is either the High, Low or Average of the N.
- If the INP is a scalar, then no filter is used. N successive reads of the scalar are done to produce one value in the resulting array.
- (The record reference manual mentions an OFF field which does not appear in the code)

Database – CPID (Cebaf PID)

- A control algorithm that changes an output value based on the error between the user specified setpoint and the readback value. This implementation of the algorithm is:
- $$delM(n) = Kp\,(E(n)) - E(n-1)) + Ki * E(i) * dT(n) + Kd * \frac{E(n) - E(n-1)}{dT(n)} - \frac{E(n) - E(n-1)}{dT(n-1)}$$

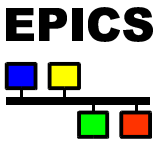
  proportion + integral + derivative

- When OMOD is Change then OVAL is set to DM (The analog output would need to have OIF set to Incremental for an absolute analog output - Absolute for a motor controller). If OMOD is POSITION, the OVAL is set to OVAL + DM.
- LOC, MMOD and SMOD can change the operation from the above description into an override situation. See the record reference manual.
- MIN and MAX limit the OVAL and DMIN and DMAX limit the DM (change in the output).
- There is also a very similar EPID record type available. Both derive from the original (buggy) PID record type.



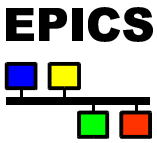
Database - Fanout

- Used to fanout the processing link FLNK to more than one record. This record contains 6 forward links.
- As FLNKs are processed as a recursive subroutine call, using a FANOUT block is a way to limit the use of the stack on very large processing chains. This record is also useful for the conditional processing of sets of records.
- SELM sets the mode for executing the links. ALL - processes all links in order. SELECT will process only the link number set in SELN. MASK will process all links up to SELN.
- When the address field SELL is defined, a database value is read and placed into SELN.



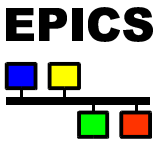
Database – Data Fanout

- Used to fanout the same value to up to 8 database locations. This may be useful for placing a change of an alarm limit to many records.
- Uses the OMSL and DOL fields like all output records.



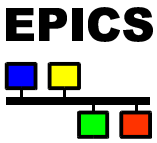
Database - Histogram Record

- Keeps hit counts of the number of times a value was in a certain value range. This array can be used to monitor range of operation for a given signal and watch for excursions outside of the normal operating range.
- SVL specifies the location of the value.
- There are NELM buckets of equal size that range between ULIM and LLIM.
- This record features a timed monitor parameter, SDEL, that causes a callback to be executed every SDEL seconds to post this monitor.



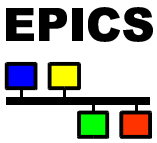
Database - Multi-bit Binary In/Out

- Define up to 16 states with 32 bit value patterns .
- Specify the starting bit number in the INP/OUT field along with the number of bits used (NOBT) that must be adjacent.
- Converts a bit pattern into a state.
- Non-matching bit patterns have an alarm status of STATE ALARM and a value of 65535.
- If no state strings or state values are defined, the bit pattern is right justified and placed in the VAL field.
- Typically the RVAL field is the bit pattern read from the hardware and right shifted to have the least significant bit as bit 0. This is the responsibility of the device support - so there is no guarantee.



Database – Multi-bit Binary In/Out Direct

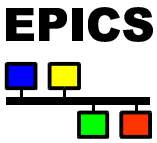
- Reads in up to 16 bits without needing name space and records to support each bit.
- Specify the starting bit number in the INP/OUT field along with the number of bits used (NOBT) that must be adjacent.
- Individual bits are addressed through fields B0 through BF.
- RVAL typically has the right justified value read from the hardware and shifted right by a signal number from the device address.



Database - Scan Record

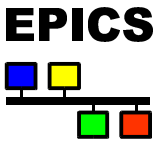
- Used to set up data taking experiments using up to four controllers collecting data from up to four input channels.

- This N dimensional control space can be coordinated through a series of records to wait for movement to be complete before collecting data.



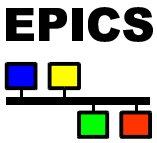
Database - Select

- The VAL field is the result of a selection from up to 12 values from INP links.
- The selection is based on the mode set in SELM.
  - SELM of High, Low or Median selects based on relationship to the other values.
  - An SELM of Specified uses the NVL link to fetch the number of the link to use, or the value in SELN if there is no link defined for NVL.



Database – Sequence Record

- Provides a mechanism for executing up to 10 read and/or write cycles with defined delays.
- Each cycle reads a value through an optional DOLx and then writes the value to an optional LNKx. (Remember: address fields with constants set the value of the D0x field on initialization and subsequent writes will write whatever value is currently in D0x.)
- A configurable delay is available between each cycle.
- When the select mode (SELM) is ALL, each of the 10 cycles is performed. When SELM is SELECT, a value fetched through the SELL field or if no link the current SELN value is used to determine which of the 10 links to execute.



Database – Subroutine Record

- Calls a pair of C subroutines in the context of the database execution. The subroutine named in INAM is called at record initialization, and the routine named in SNAM is called every time the record processes.
- From R3.14.1 onwards, on all OS's except vxWorks the subroutines must be named in a 'function()' statement in the IOC's .dbd file.
- 12 INP links are available to read up to 12 values into fields A-L.
- BRSV is the alarm severity if the subroutine returns less than 0.
- Your subroutine gets a pointer to the database record structure of the calling record. This means that all fields of the subroutine record may be accessed and altered in the context of the subroutine.
- Refer to the record reference manual for examples of asynchronous record completion.