EPICS Class – Process Database

• Most slides here are taken from:
  – “EPICS Database Principles” by Andrew Johnson

• Some slides modified by me

• After lecture, study:
  – “EPICS Database Practice” by Andrew Johnson
EPICS Class – Process Database Outline

• What is a process database and why use it?
• Records
• Fields in Records (PVs)
• Record Processing
• Input and Output Records
• Record Linking
• Device Support and Soft Records
• Synchronous and Asynchronous I/O
• Common Fields/Functions in Most Records
Control System w/o EPICS Database*

* An evolutionary tale.
Control System w/o EPICS Database

Network

TCP

Value Probe
Some Host1
Control System w/o EPICS Database

Network

TCP

Value Probe
Some Host1

History
Some Host2
Control System w/o EPICS Database

- Value Probe
- History
- Alarm Handler

Network

TCP

Only 2 Clients Allowed!
Control System w/o EPICS DB

Some Server1

TCP

Network

Some Host1

Value Probe

Some Driver1

TCP

History

Some Host2

Alarm Handler

Some Host3
Control System w/o EPICS DB

Some Driver1

Some Server1

TCP

Network

TCP

Value Probe
Some Host1

History
Some Host2

Alarm Handler
Some Host3
Control System w/o EPICS DB

- Value Probe
  - Some Host1
- Network
- Channel Access
  - Some Driver1 with CA Server*
  - Some Server1
- History
  - Some Host2
- Alarm Handler
  - Some Host3

* Very few people actually do this – what they do instead will be discussed later.
Control System w/o EPICS DB

TCP

Value Probe
Some Host1

Network
Channel Access

History
Some Host2

Alarm Handler
Some Host3

UDP

Some Driver1 with CA Server
Some Server1

Some Driver2 with CA Server
Some Server2
Control System w/o EPICS DB

TCP

CC Drv
CC Analog Input Processing
CAS1
Some Server1

Network

Channel Access

Value Probe
Some Host1

History
Some Host2

Alarm Handler
Some Host3

UDP

PSC Drv
PSC Analog Input Processing
CAS2
Some Server2
Control Sys with EPICS DB

TCP

IOC1 on Server1

Config files on NFS

Network

Channel Access

IOC2 on Server2

UDP

Value Probe

Some Host1

History

Some Host2

Alarm Handler

Some Host3
Database = Records + Fields + Links

• A control system using EPICS will contain one or more IOCs
• Each IOC loads one or more Database files telling it what to do
• A Database is a collection of Records of various types:
  – ai, ao, bi, bo, mbbi, mbbo, stringin, stringout, calc, etc
• A Record is an object with:
  – A unique name
  – A behavior defined by its record type (class)
  – Controllable properties (fields)
  – Optional associated hardware I/O (device support)
  – Links to other records
Record Activity

• Records are active — they can do things:
  – Get data from other records or from hardware
  – Perform calculations
  – Check values are in range & raise alarms
  – Put data to other records or to hardware
  – Activate or disable other records
  – Wait for hardware signals (interrupts)

• What a record does depends upon its record type and the settings of its fields

• No action occurs unless a record is processed
How is a Record type implemented?

- A ‘C’ structure with a data member for each record field
  - All records start with a standard set of fields (dbCommon) that the system needs, including pointers to record type information (base/include/dbCommon.h)
  - Additional fields are added depending on record type
  - The configuration of the ‘C’ structure is provided by a database definition ASCII file (<name>Record.dbd) which is unique to the record type <name>
    - Standard records in base/src/rec/<name>Record.dbd
- Code (ie, ‘C’) which implements the record behavior
  - Standard records in base/src/rec/<name>Record.c
- New record types can be added to an application as needed – recommend only if absolutely necessary
How are individual Records defined?

• A database configuration ASCII file (.db or .template) providing record field values:
  – Record name
  – The record’s type
  – Values for each design field
  – Can use macros for fields so that same configuration file is used for many records

• IOC application build (make) can create and/or install .db files

• Record instantiation (ie, memory allocation on IOC) done before iocInit by dbLoadRecords in the IOC startup file.
Operator Display Parameters

These parameters are used to present meaningful data to the operator. They display the value and other parameters of the analog input either textually or graphically. EGU is a string of up to 16 characters describing the units that the analog input measures. It is retrieved by the get_units record support routine.

The HOPR and LOPR fields set the upper and lower display limits for the VAL, HIHI, HIGH, LOW, and LOLO fields. Both the get_graphic_double and get_control_double record support routines retrieve these fields.

The PREC field determines the floating point precision with which to display VAL. It is used whenever the get_precision record support routine is called.

See Fields Common to All Record Types for more on the record name (NAME) and description (DESC) fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Summary</th>
<th>Type</th>
<th>DCT</th>
<th>Initial</th>
<th>Access</th>
<th>Modify</th>
<th>Rec Proc Monitor</th>
<th>PP</th>
</tr>
</thead>
<tbody>
<tr>
<td>EGU</td>
<td>Engineering Units</td>
<td>STRING [16]</td>
<td>Yes</td>
<td>null</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>HOPR</td>
<td>High Operating Range</td>
<td>DOUBLE</td>
<td>Yes</td>
<td>0</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>LOPR</td>
<td>Low Operating Range</td>
<td>DOUBLE</td>
<td>Yes</td>
<td>0</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>PREC</td>
<td>Display Precision</td>
<td>SHORT</td>
<td>Yes</td>
<td>0</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>NAME</td>
<td>Record Name</td>
<td>STRING [29]</td>
<td>Yes</td>
<td>0</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESC</td>
<td>Description</td>
<td>STRING [29]</td>
<td>Yes</td>
<td>Null</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Graphical View of a Record (VDCT)
record(ai, "MS1-BD:CntlTempF")
{
    field(DESC, "Controller Temp (deg F)")
    field(DTYP, "EtherPSC")
    field(INP, "#L0 N0 P0 S41 @172.22.247.141")
    field(EGU, "F")
    field(PREC, "1")
    field(HOPR, "95")
    field(LOPR, "80")
    field(MDEL, "0.1")
    field(FLNK, "MS1-BD:CntlTemp")
}
### IOC View of a Record

```
soft-iocpsspr>dbpr("MS1-BD:Cnt1TempF",5)
ACKS: NO_ALARM ACKT: YES ADEL: 0
ALST: 95.4601745605469 AOFF: 0 ASG:
ASLO: 1 ASP: (nil) BKPT: 00
DESC: Controller Temp (deg F) DISA: 0 DISP: 0
DISS: NO_ALARM DISV: 1 DPVT: 0x1123b90 DSET: 0x768de0
DTYP: EtherPSC EGUF: 0 EGUL: 0
EOFF: 0 ESLO: 1 EVNT: 0
FLNK:DB_LINK MS1-BD:Cnt1Temp HHSS: NO_ALARM HIGH: 0
HIHI: 0 HOPR: 95 HSV: NO_ALARM HYST: 0
INIT: 0 INP:BITBUS_IO #LO NO PO S41 @172.22.247.141
LALM: 95.4601745605469 LBRK: 0 LCNT: 0
LINR: NO CONVERSION LLLV: NO_ALARM LOLO: 0 LOPR: 80
LOW: 0 LSET: 0x1132ed0 LSV: NO_ALARM MDEL: 0.1
MLIS: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MLOK: 40 bd 0b 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
MLST: 95.4074172973633 NSEV: NO_ALARM NSTA: NO_ALARM
NAME: MS1-BD:Cnt1TempF NPPR: (nil) PHAS: 0
ORAW: 0 PACT: 0 PBRC: (nil) PREC: 1
PINI: NO PPN: (nil) PPRN: (nil)
Prio: LOW PROC: 0 PUTF: 0 RDES: 0x5e770
ROFF: 0 RPRO: 0 RSET: 0x76c3e0 RVAL: 0
SCAN: Passive SDIS:CONSTANT SEVR: NO_ALARM SIML:CONSTANT
SIMM: NO SIMS: NO_ALARM SIOL:CONSTANT SMOO: 0
SPVT: (nil) STAT: NO_ALARM SVAL: 0
TIME: 2012-04-20 15:15:16.521440756 TPRO: 0 TSE: 0
TSEL:CONSTANT UDF: 0 VAL: 95.4601745605469
```
Fields – Process Variable (PV)

- PV = <record_name>.<field>
- Example: MS1-BD:CntrlTempF.DESC
- If .<field> is not provided, .VAL is assumed.
Fields are for...

- **Defining**
  - What causes a record to process
  - Where to get/put data from/to
  - How to turn raw I/O data into a numeric engineering value
  - Limits indicating when to report an alarm
  - When to notify value changes to a client monitoring the record
  - A Processing algorithm
  - Anything else which needs to be set for each record of a given type

- **Holding run-time data**
  - Input or output values
  - Alarm status, severity and acknowledgments
  - Processing timestamp
  - Other data for internal use
Field types — fields can contain:

• Integers
  – char, short or long
  – signed or unsigned

• Floating-point numbers
  – float or double

• Fixed length strings
  – normally 40 characters

• Enumerated/menu choices
  – select one of up to 16 strings
  – stored as a short integer

• Arrays of any of the above types

• Links
  – to other records in this or other IOCs
  – to hardware signals (device support)
  – provide a means of getting or putting a value

• Other private data
  – not accessible remotely
All Records have these design fields:

- **NAME**: 60 Character unique name (using more than 28 can cause problems)
- **DESC**: 28 Character description
- **ASG**: Access security group
- **SCAN**: Scan mechanism
- **EVNT**: Event number
- **PHAS**: Scan order (phase)
- **PINIT**: Process at IOC initialization?
- **PRIO**: Scheduling priority
- **SDIS**: Scan disable input link
- **DISV**: Scan disable value
- **DISS**: Disabled severity
- **FLNK**: Forward link

Plus more...
All Records have these Run-time fields:

- **PROC**: Force processing
- **PACT**: Process active
- **STAT**: Alarm status
- **SEVR**: Alarm severity
- **TPRO**: Trace processing
- **UDF**: Non-zero if record value undefined
- **TIME**: Time when record was last processed

*Plus more...*
Record Processing

IOC

Scan Task (0.1 sec)  ...  Scan Task (10 sec)

EPICS Database with Record Fields

Callback Task (High Prio)
Callback Task (Medium Prio)
Callback Task (Low Prio)

Hardware
How are records given CPU time?

Several IOC tasks are used:

• callback (3 priorities) — I/O Interrupt and Event
• scanPeriod (periods are configurable) — Periodic
  – A separate task is used for each scan period
  – Faster scan rates are given a higher task priority (if supported by the IOC’s Operating System)
• scanOnce – special purpose
• sequence and other run-time database tasks
• Channel Access tasks use lower priority than record processing
  – If a CPU spends all its time doing I/O and record processing, you may be unable to control or monitor the IOC via the network
Record Scanning

• **SCAN** field is a menu choice from
  – Periodic — 0.1 seconds .. 10 seconds
  – I/O Interrupt (if device supports this)
  – Soft and Hard event — **EVNT** field
  – Passive (default)
• The number in the **PHAS** field allows processing order to be set within a scan
  – Records with **PHAS=0** are processed first
  – Then those with **PHAS=1**, **PHAS=2** etc.
• Records with **PINI=YES** are processed once at startup
• **PRIO** field selects Low/Medium/High priority for event and I/O Interrupts (selects which callback task will process the record)
• A record is also processed whenever any value is written to its **PROC** field
Listing Records for each SCAN Type

```bash
iocpsgendev> scanppl
Scan Period = 1 seconds
  I0CPSGENDEV:HEARTBEAT
  118-PSD4:BitbusCountCmd
  118-PSD4:BitbusCountRsp
  LTB-CORB2:BitbusCountCmd
  LTB-CORB2:BitbusCountRsp
  I0CPSGENDEV:TOD
  I0CPSGENDEV:UPTIME

iocpsgendev> scanpel

iocpsgendev> scanpiol

IO Event: Priority Low
  I0CPSGENDEV:CA_CLNT_CNT
  I0CPSGENDEV:CA_CONN_CNT

IO Event: Priority Low
  I0CPSGENDEV:FD_CNT

IO Event: Priority Low
  I0CPSGENDEV:IOC_CPU_LOAD
  I0CPSGENDEV:SUSP_TASK_CNT
  I0CPSGENDEV:SYS_CPU_LOAD

IO Event: Priority Low
  I0CPSGENDEV:MEM_FREE
  I0CPSGENDEV:MEM_MAX
  I0CPSGENDEV:MEM_USED
```
Input and Output Records
Input records often have these fields:

- **INP**: Input link
- **DTYP**: Device type
- **RVAL**: Raw data value
- **VAL**: Engineering value
- **LOPR**: Low operator range
- **HOPR**: High operator range
Analog I/O (ai, ao) records have these fields:

- **EGU**  Engineering unit string
- **LINR**  *Unit conversion control*: No conversion, Linear, Slope, breakpoint table name
- **EGUL**  Low engineering value
- **EGUF**  High engineering value
- **ESLO**  Unit conversion slope
- **EOFF**  Unit conversion offset
Periodically Scanned Analog Input

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

- Binary Input “Vent Valve”
- Reads from Allen-Bradley TTL I/O Link 0, Adaptor 0, Card 3, Signal 5
- Processed whenever value changes
- 0 = “Closed”, 1 = “Open”
- Major alarm when valve open
Most output records have these fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td>Output link</td>
</tr>
<tr>
<td>DTYP</td>
<td>Device type</td>
</tr>
<tr>
<td>VAL</td>
<td>Engineering value</td>
</tr>
<tr>
<td>RVAL</td>
<td>Raw output value</td>
</tr>
<tr>
<td>DOL</td>
<td>Input link to fetch output value</td>
</tr>
<tr>
<td>OMSL</td>
<td>Output mode select: Supervisory, Closed Loop</td>
</tr>
<tr>
<td>LOPR</td>
<td>Low operator range</td>
</tr>
<tr>
<td>HOPR</td>
<td>High operator range</td>
</tr>
</tbody>
</table>
Analog outputs (ao) records also have these fields:

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

Note: Restoration of field values (ie, analog output VAL, aka setpoint) on IOC restart is not provided by epics base. Software (or module, ie autosave module) external to epics base must be added to the IOC application and additional configuration done to add this very important functionality.
Passive Binary Output

- Binary Output “Solenoid”
- Controls Xycom XY220 Digital output Card 2 Signal 12
- Record is only processed by
  - Channel Access ‘put’ to a PP field (e.g. .VAL)
  - Another record writes to a PP field
  - Forward Link from another record
  - Another record reads this with PP

<table>
<thead>
<tr>
<th>bo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solenoid</td>
</tr>
<tr>
<td>DTYP=XY220</td>
</tr>
<tr>
<td>CUT=#C0 S12</td>
</tr>
<tr>
<td>SCAN=Passive</td>
</tr>
<tr>
<td>PHAS=0</td>
</tr>
<tr>
<td>ZNAM=Locked</td>
</tr>
<tr>
<td>CNAM=Unlocked</td>
</tr>
<tr>
<td>CMSL=supervisory</td>
</tr>
</tbody>
</table>
A link is a type of field, and is one of

- Input link (ie, input record INP field, output record DOL field)
  - Fetches data
- Output link (ie, output record OUT field)
  - Writes data
- Forward link (ie, any record FLNK field)
  - Points to the record to be processed once this record finishes processing
Input and Output links may be...

• Constant numeric value, e.g.:
  
  0
  
  3.1415926536
  
  -1.6e-19

• “Hardware” link

  A hardware (or external) I/O signal selector, the format of which depends on the device support layer

• Process Variable link — the name of a record, which at run-time is resolved into
  
  — Database link - Named record is in this IOC
  
  — Channel Access link - Named record not found in this IOC
## Hardware links

<table>
<thead>
<tr>
<th>Hardware Link</th>
<th>Parameters</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>VME_IO</td>
<td>#Cn Sn @parm</td>
<td>Card, Signal</td>
</tr>
<tr>
<td>INST_IO</td>
<td>@parm</td>
<td></td>
</tr>
<tr>
<td>CAMAC_IO</td>
<td>#Bn Cn Nn An Fn @parm</td>
<td>Branch, Crate, Node, Address, Function</td>
</tr>
<tr>
<td>AB_IO</td>
<td>#Ln An Cn Sn @parm</td>
<td>Link, Adapter, Card, Signal</td>
</tr>
<tr>
<td>GPIB_IO</td>
<td>#Ln An @parm</td>
<td>Link, Address</td>
</tr>
<tr>
<td>BITBUS_IO</td>
<td>#Ln Nn Pn Sn @parm</td>
<td>Link, Node, Port, Signal</td>
</tr>
<tr>
<td>BBGPPIB_IO</td>
<td>#Ln Bn Gn @parm</td>
<td>Link, Bitbus Address, GPIB Address</td>
</tr>
<tr>
<td>VXI_IO</td>
<td>#Vn Cn Sn @parm or #Vn Sn @parm</td>
<td>Frame, Slot, Signal</td>
</tr>
</tbody>
</table>

EPICS Class – Process Database  
4/23/2012
Database links

• These comprise:
  – PV – process variable name
  – Process Passive flag
    • NPP (default), or PP
  – Maximize Severity flag
    • NMS  No maximize severity (default)
    • MS   Maximize severity
    • MSS  Maximize Status and Severity
    • MSI  Maximize Severity when Invalid

• Example
  M1:current.RBV NPP MS

• Beware: Database links with the PP flag set never wait for asynchronous record processing to finish, so an input link that triggers a read from slow hardware will return the previous data in that record
Channel Access links

- Similar to a database link
- Names a record that does not have to be in this IOC
- Use Channel Access protocol to communicate with the record
  - Just like any other CA client, even for local records
  - Input sets up a CA monitor on the channel
- May include a field name (default \texttt{.VAL})
- \texttt{PP} Link flags are ignored
  - Input links are always \texttt{NPP}
  - Output links follow \texttt{PP} attribute of destination field
  - These are how all CA clients behave
- \texttt{MS} Link flags apply to Input links
  - Input links honor a given \texttt{NMS} (default) or \texttt{MS/MSS/MSI} flag
  - Output links are always \texttt{NMS}
- Additional flags for CA links
  - \texttt{CA} Forces a “local” link to use CA
  - \texttt{CP} On input link, process this record on CA monitor event
  - \texttt{CPP} Like \texttt{CP} but only process me if \texttt{SCAN} is Process Passive
Forward links

• Usually a Database link, referring to a record in same IOC
• No flags (PP, MS etc.), although VDCT includes them erroneously
• Destination record is only processed if its **SCAN** field is Passive
• Does not pass a value, just causes subsequent processing
• Forward linking to another IOC via Channel Access is possible, but the link must explicitly name the **PROC** field of the remote record
  – In this case, the remote record does not need to have **SCAN** set to Passive
Chapter 5 of the IOC Application Developer’s Guide covers record links and scanning in detail, and is worth reading.

Good idea to just read the whole manual…
Simple FLNK and INP Link Example (VDCT)
Trace Processing

ioreg
dbpf("118-PSD4:Cnt1TempF.TPRO","0")
DBR_UCHAR: 0 0x0
ioreg
dbpf("118-PSD4:Cnt1TempF.TPRO","1")
DBR_UCHAR: 1 0x1
ioreg

scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
scan1: Process 118-PSD4:Cnt1Temp
scan1: Process 118-PSD4:Cnt1TempF
Finding Typos in Links

beldar:"$caput 118-PSD4:CntlTemp.INPA "118-PSD4:CntlTempFtypo NPP MS"
Old : 118-PSD4:CntlTemp.INPA 118-PSD4:CntlTempF NPP MS
New : 118-PSD4:CntlTemp.INPA 118-PSD4:CntlTempFtypo NPP MS
beldar:"$cmonitor 118-PSD4:CntlTempF 118-PSD4:CntlTemp
118-PSD4:CntlTempF 2012-04-22 13:09:57.242178 85.6773
118-PSD4:CntlTemp 2012-04-22 13:09:57.242179 29.8154 LINK INVALID
118-PSD4:CntlTempF 2012-04-22 13:09:58.142895 85.6869
118-PSD4:CntlTempF 2012-04-22 13:09:59.043299 85.6885
^C
beldar:"$caput 118-PSD4:CntlTemp.INPA "118-PSD4:CntlTempF NPP MS"
Old : 118-PSD4:CntlTemp.INPA 118-PSD4:CntlTempF NPP MS
New : 118-PSD4:CntlTemp.INPA 118-PSD4:CntlTempFtypo NPP MS
beldar:"$cmonitor 118-PSD4:CntlTempF 118-PSD4:CntlTemp
118-PSD4:CntlTempF 2012-04-22 13:10:03.545845 85.6877
118-PSD4:CntlTemp 2012-04-22 13:10:03.545845 29.8265
118-PSD4:CntlTempF 2012-04-22 13:10:04.446311 85.6875
118-PSD4:CntlTemp 2012-04-22 13:10:04.446317 29.8264
118-PSD4:CntlTempF 2012-04-22 13:10:05.346903 85.6856
118-PSD4:CntlTemp 2012-04-22 13:10:05.346911 29.8254

iocpsgendev>help dbcar
dbcar 'record name' level
iocpsgendev>dbcar "" 2
CA links in all records

118-PSD4:CntlTemp.INPA --> 118-PSD4:CntlTempFtypo (0, 0)
Total 1 CA link; 0 connected, 1 not connected.
  0 can't read, 0 can't write. (0 disconnects, 0 writes prohibited)
iocpsgendev>dbcar "" 2
CA links in all records
Total 0 CA links; 0 connected, 0 not connected.
  0 can't read, 0 can't write. (0 disconnects, 0 writes prohibited)
iocpsgendev>
Processing chains
Which record is never processed?
How often is Input_1 processed?
Device Support

- Records do not access hardware directly
- The Device Support layer performs I/O operations on request
- A particular device support provides I/O for a single record type
- The `DTYP` field determines which device support to use, default is normally “Soft Channel” support when not set
- The device support selected determines the format of the link (INP or OUT field) containing device address information
- Device support uses `DPVT` pointer to store device information for the record instance.
- Device support may also reset the `UDF` flag when the record is properly initialized.
- Additional fields added per record type for purpose of device support (examples, `MASK` and `SHFT` for mbbi, `ROFF` and `RVAL` for ai)
Device Support, cont

• Adding new device support does not require any changes or recompilation of the record type code
• Device support often calls other software to do work for it (Driver Support or other libraries)
• Device support most often thin layer – record and driver support do most of the legwork
Record/Device/Driver Example Software Hierarchy

- Record Support (ie, aiRecord.c)
- Device Support (ie, devAiSomeHW.c)
- Driver Support (ie, drvSomeHW.c)

EPICS Database with Record Fields

Device/Driver Shared Memory

Hardware

Driver Task
Soft Device Support

• “Hard” input and output records do external I/O via device support
• “Soft” records access data from other records via DB or CA links
• 2 or 3 kinds of support are provided in recent R3.14 releases:
  – Soft Channel
    • Get/Put \textbf{VAL} through link, no units conversion preformed
  – Async Soft Channel (currently output records only)
    • Put \textbf{VAL} through CA link, no conversions, wait for completion
  – Raw Soft Channel
    • Inputs
      – Get \textbf{RVAL} via input link
      – Convert \textbf{RVAL} to \textbf{VAL} (record-type specific)
    • Outputs
      – Convert \textbf{VAL} to \textbf{RVAL} (record-type specific)
      – Put \textbf{RVAL} to output link
• Note – remember \textbf{RVAL} is integer!!
Control System w/o traditional EPICS

Network

Value Probe
Some Host1

History
Some Host2

Alarm Handler
Some Host3

Channel Access

TCP
Driver1 w CA Client
Some Server1

CAS
Soft Records

IOC on some Server3

UDP
Driver2 w CA Client
Some Server2
Control System with just Soft Records

- Do not have to read the App Dev Guide, take EPICS classes, or be an active EPICS collaborator.

- EPICS is a convenient scapegoat – system too slow? Blame it on EPICS. Changes take time to implement? Blame it on EPICS. Call mythbusters.

- However:
  - Changes DO take more time to implement on systems like this.
  - Harder to timestamp and correlate data.
  - Slower due to extra network traffic.
  - Harder to upgrade to different versions and platforms.
  - Cannot use other people’s code (ie, areaDetector, motor).
  - Etc
Synchronous vs Asynchronous I/O

• EPICS rules do not allow device support to busy-wait (i.e. delay record processing while waiting for the results of a slow I/O operation)
  – Fast I/O can be handled synchronously
  – Slow operations must operate asynchronously

• Register-based VME/PCI cards usually give an immediate response: synchronous
  – When called, a synchronous read or write call to device support performs all I/O needed before returning

• Serial, network or field-bus I/O usually takes some time (>10ms) to return data: asynchronous
  – Asynchronous device support starts an I/O operation when the record calls it, flagging it as incomplete by setting PACT to true before returning
  – When the results are available (discovered by a CPU interrupt or polling background thread), the device support must call the record’s process() routine to finish the record processing operations and set PACT to false
More about the PACT field

• Every record has a boolean run-time field called **PACT** (Process Active)
• **PACT** breaks loops of linked records
• It is set to true early in the act of processing the record (but it's not the first thing that the process routine does)
  – **PACT** should always be true whenever a link in that record is used to get/put a value
• **PACT** gets reset to false after all record I/O and forward link processing are finished
• A PP link can never make a record process if it has **PACT** true
  – Input links will take the current field value
  – Output links just put their value to the field
  – Investigate cached puts
What happens here?
Other Common Fields and Functions
Record Status and Severity (STAT/SEVR)

• When record processing is started, the record STAT and SEVR enum fields are initialized to OK (NO_ALARM and NO_ALARM) by the IOC. Any record links or other processing may “maximize” STAT and SEVR (make them worse).

• Device support read/write routines must update record STAT/SEVR on a warning or error condition (such as a device or communication problem) so that clients can use the data intelligently.

• SEVR values are:
  – NO_ALARM, MINOR_ALARM, MAJOR_ALARM, INVALID_ALARM
  – INVALID_ALARM is most often used for failures.

• STAT values are listed in base/include/alarm.h and include:
  – READ, WRITE, UDF, HIGH, LOW, STATE, COS, CALC, DISABLE, etc

• Some processing also set VAL to NaN for failures.
Record Alarms

• Record **STAT** and **SEVR** can be set based on alarm limit or state checks done in record processing after the value is determined.

• Most numeric records check **VAL** against limits:
  – **HIHI**, **HIGH**, **LOW**, **LOLO**

• The **HYST** field prevents alarm chattering

• A separate alarm severity can be set for each numeric limit exceeded:
  – **HHSV**, **HSV**, **LSV**, **LLSV**

• Discrete (binary) records can raise alarms on entering a particular state, or on a change of state (COS)

• There is nothing to prevent alarm chattering for binary records so filtering in the alarm log CA client is normally used when needed.
Record Timestamp (**TIME**) 

- The record timestamp (**TIME**) is set at the end of processing and is set based on the value of **TSE**:  
  - 0 (default) - set to the system time (registered with \texttt{generalTime})  
  - -2 - set by device support  
  - -1 - set by the default event time provider (registered with \texttt{generalTime})  
  - >0 - set by the event time provider with the **TSE** value as input  

- If a **TSEL** PV link is provided, the timestamp is copied from that PV instead.  
- Timestamps are useful to correlate data across IOCs when IOCs share a timing system.  
- Might want a timestamp to reflect the time of a trigger vs the time that it takes to read and process the data.
Record Alarm and Timestamp Example

beldar: "$caput 118-PSD4:Cnt1TempF.MDEL 0
Old : 118-PSD4:Cnt1TempF.MDEL 0.03
New : 118-PSD4:Cnt1TempF.MDEL 0
beldar: "$caput 118-PSD4:Cnt1Temp.MDEL 0
Old : 118-PSD4:Cnt1Temp.MDEL 0.1
New : 118-PSD4:Cnt1Temp.MDEL 0
beldar: "$camonitor 118-PSD4:Cnt1TempF 118-PSD4:Cnt1Temp
118-PSD4:Cnt1TempF 2012-04-22 12:46:32.607235 85.6614
118-PSD4:Cnt1Temp 2012-04-22 12:46:32.607236 29.8119
118-PSD4:Cnt1TempF 2012-04-22 12:46:33.507582 85.6656
118-PSD4:Cnt1Temp 2012-04-22 12:46:33.507586 29.8142
^C
beldar: "$caput 118-PSD4:Cnt1Temp.HIHI 29
Old : 118-PSD4:Cnt1Temp.HIHI 41
New : 118-PSD4:Cnt1Temp.HIHI 29
beldar: "$camonitor 118-PSD4:Cnt1TempF 118-PSD4:Cnt1Temp
118-PSD4:Cnt1TempF 2012-04-22 12:47:09.526784 85.6887
118-PSD4:Cnt1Temp 2012-04-22 12:47:09.526785 29.827 HIHI MAJOR
118-PSD4:Cnt1TempF 2012-04-22 12:47:10.427425 85.7041
118-PSD4:Cnt1Temp 2012-04-22 12:47:10.427429 29.8356 HIHI MAJOR
^C
beldar: "$caput 118-PSD4:Cnt1Temp.TSEL "118-PSD4:Cnt1TempF.TIME"
Old : 118-PSD4:Cnt1Temp.TSEL 0
New : 118-PSD4:Cnt1Temp.TSEL 118-PSD4:Cnt1TempF.VAL NPP NMS
beldar: "$camonitor 118-PSD4:Cnt1TempF 118-PSD4:Cnt1Temp
118-PSD4:Cnt1TempF 2012-04-22 12:47:39.242830 85.6405
118-PSD4:Cnt1Temp 2012-04-22 12:47:39.242830 29.8003 HIHI MAJOR
118-PSD4:Cnt1TempF 2012-04-22 12:47:40.143155 85.6432
118-PSD4:Cnt1Temp 2012-04-22 12:47:40.143155 29.8018 HIHI MAJOR
118-PSD4:Cnt1TempF 2012-04-22 12:47:41.043640 85.6447
118-PSD4:Cnt1Temp 2012-04-22 12:47:41.043640 29.8026 HIHI MAJOR
^C
beldar: "$
Record Disabling (**DISV/DISS**) 

• It is useful to be able to stop an individual record from processing on some condition

• Before record-specific processing is called, a value is read through the **SDIS** input link into **DISA** (which defaults to 0 if the link is not set)

• If **DISA**=**DISV**, the record will *not* be processed

• The default value of the **DISV** field is 1

• A disabled record may be put into an alarm state by giving the desired severity in the **DISS** field (default of NO_ALARM)

• The **FLNK** of a disabled record is never triggered

• Trick – some applications use **SDIS** to propagate alarms instead of disable the record.
Record Lock-Sets (**LSET**)

- Prevent a record from being processed simultaneously from two different (ie, scan) tasks
  - PACT can’t do that, it isn’t set early enough and is not a Mutex
- A lock-set is a group of records interconnected by database links
- Lock-sets are determined automatically by the IOC at start-up, or whenever a database link is added, deleted or modified
- When lock-sets are too big and records process at a high rate, channel access (and other low priority tasks) performance may be affected.
- A lock-set can be split into different lock sets by making the link(s) joining them into Channel Access ones, using the CA flag
  - Remember that CA links behave slightly differently than DB links, make sure your design still works!
- A lock-set can also be split using event processing.
Lock Set Example

```
iorcpsgendev> dblsr "118-PSD4:CntlTemp" 2
globalLock 0x10043b0
lockSetModifyLock 0x1004420
Lock Set 201 2 members epicsMutexId 0x1006190 Not Locked
118-PSD4:CntlTempF
     FLNK     FWDLINK NPP NMS 118-PSD4:CntlTemp
118-PSD4:CntlTemp
     INPA     INLINK NPP MS 118-PSD4:CntlTempF
iorcpsgendev> dbpf("118-PSD4:CntlTempF.INPA","118-PSD4:CntlTempF.CPP MS")
DBR_STRING:   "118-PSD4:CntlTempF.CPP MS"
iorcpsgendev> dbpf("118-PSD4:CntlTempF.FLNK",""")
DBR_STRING:   "0"
iorcpsgendev> dblsr "118-PSD4:CntlTemp" 2
globalLock 0x10043b0
lockSetModifyLock 0x1004420
Lock Set 204 1 members epicsMutexId 0x10060b0 Not Locked
118-PSD4:CntlTemp
iorcpsgendev> dblsr "118-PSD4:CntlTempF" 2
globalLock 0x10043b0
lockSetModifyLock 0x1004420
Lock Set 203 1 members epicsMutexId 0x1006190 Not Locked
118-PSD4:CntlTempF
```
What could go wrong here?
Change Notification: Monitor Dead-bands

- Channel Access notifies clients that are monitoring a numeric record when
  - **VAL** changes by more than the value in field:
    - **MDEL** Value monitors
    - **ADEL** Archive monitors
  - Record’s Alarm Status changes
    - **HYST** Alarm hysteresis

- The Analogue Input record has a smoothing filter to reduce noise on the input signal (SMOO)
Breakpoint Tables

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

```plaintext
breaktable (typeKdegC) {
  0.000000  0.000000
  299.268700  74.000000
  660.752744  163.000000
  1104.793671 274.000000
  1702.338802 418.000000
  2902.787322 703.000000
  3427.599045 831.000000
  ...
}
```
Simulation

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

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

- Security rules are loaded from an Access Security Configuration File, for example:
  
  UAG(users) {user1, user2}
  HAG(hosts) {host1, host2}
  ASG(DEFAULT) {
      RULE (1, READ)
      RULE (1, WRITE) {
        UAG(users)
        HAG(hosts)
      }
  }

- If no security file is loaded, Security will be turned off and nothing refused
- For more details and the rule syntax, see Chapter 8 of the IOC Application Developers Guide
Order of Operations (Synchronous I/O)

1. Every 0.1 seconds, iocCore will attempt to process the Output_1 record
2. The Output_1.PACT field is currently False, so the record is quiescent and can be processed
3. If set, the Output_1.SDIS link would be read into Output_1.DISA
4. Since DISA≠DISV, the ao record type's process() routine is called
5. The ao's process() routine checks the `Output_1.OMSL` field; it is closed_loop, so
6. It sets `Output_1.PACT` to True, then
7. Reads a value through the `Output_1.DOL` link
8. The `Output_1.DOL` link contains `Calculation_1.VAL PP` so this first attempts to process the `Calculation_1` record
Order of Operations (Synchronous I/O)

9. The Calculation_1.SCAN field is Passive and Calculation_1.PACT is False, so processing is possible.

10. If set, the Calculation_1.SDIS link would be read into DISA.

11. Since DISA≠DISV, the calc record type's process() routine is called.
Order of Operations (Synchronous I/O)

12. The calc's process() routine sets Calculation_1.PACT to True, then
13. Starts a loop to read values from the links INPA through INPL
14. The Calculation_1.INPA link is set to Input_1.VAL PP so this first attempts to process the Input_1 record
15. The Input_1_SCAN field is Passive and Input_1_PACT is False, so processing is possible.

16. If set, the Input_1_SDIS link is read into the Input_1_DISA field.

17. Since DISA ≠ DISV, the ai record type's process() routine is called.

18. The ai process() calls the associated device support to read a value from the hardware it's attached to.
19. The device support is synchronous, so it puts the hardware input value into the `Input_1.RVAL` field and returns to the ai record's process() code.

20. The `Input_1.PACT` field is set to True.

21. The record's timestamp field `Input_1.TIME` is set to the current time.

22. The raw value in `Input_1.RVAL` is converted to engineering units, smoothed, and the result put into the `Input_1.VAL` field.
23. The Input_1.VAL is checked against alarm limits and monitor deadbands, and appropriate actions is taken if these are exceeded.

24. If the Forward Link field Input_1.FLNK is set, an attempt is made to process the record it points to.

25. The Input_1.PACT field is set to False, and the process() routine returns control to the Calculation_1 record.
26. The value read through the `Calculation_1.INPA` link is copied into the `Calculation_1.A` field.

27. The Calculation record type's process() routine continues to loop, reading its input links.

28. In this example only the `INPA` link is set, so the routine finishes the loop and evaluates the `Calculation_1.CALC` expression (not shown).

29. The result of the expression is put in the `Calculation_1.VAL` field.
30. The record's timestamp field `Calculation_1.TIME` is set to the current time

31. `Calculation_1.VAL` is checked against alarm limits and monitor deadbands, and appropriate action is taken if these are exceeded

32. If the Forward Link field `Calculation_1.FLNK` is set, an attempt is made to process the record it points to

33. The `Calculation_1.PACT` field is set to False, and the process() routine returns control to the `Output_1` record
Order of Operations (Synchronous I/O)

34. The value read through the $\text{Output}_1.DOL$ link would now be forced into the range $\text{DRV}_L \ldots \text{DRV}_H$ if those fields were set, but they aren't so it's copied to the $\text{Output}_1.VAL$ field unchanged.

35. The $\text{Output}_1.VAL$ value is converted from engineering to raw units and placed in $\text{Output}_1.RVAL$.

36. $\text{Output}_1.VAL$ is checked against alarm limits and monitor dead-bands, and appropriate action is taken if these are exceeded.

37. The associated device support is called to write the value to the hardware.
38. The device support is synchronous, so it outputs the value to the attached hardware and returns

39. The record's timestamp field $Output_1.TIME$ is set to the current time

40. If the Forward Link field $Output_1.FLNK$ is set, an attempt is made to process the record it points to

41. The $Output_1.PACT$ field is set to False, and the process() routine returns