Control System

.. should support automated **control**.

How can EPICS do this?
Monitoring, Supervisory Control

**IOC**

```plaintext
record(ai, "sensor")
{
  field(DTYP, "SensorXYZ")
  field(INP, "@bus1 signal2")
  field(SCAN, "1 second")
  ...
}

record(ao, "voltage")
{
  field(DTYP, "PowerSupplyABC")
  field(OUT, "@bus2 signal4")
  field(SCAN, "Passive")
  ...
}
```

**User Interface**

Channel Access ‘monitor’

Channel Access ‘put’
Automation via Records on IOC

Data flow driven, periodic, steady-state control:

1. Read inputs
2. Compute desired outputs
   • calc, calcout records
3. Write outputs
Distribute Records onto different IOCs

**IOC**

record(ai, "sensor")
{
  field(DTYP, "SensorXYZ")
  field(INP, "@bus1 signal2")
  field(SCAN, "1 second")
  ...
}

record(ao, "voltage")
{
  field(DTYP, "PowerSupplyABC")
  field(OUT, "@bus2 signal4")
  field(SCAN, "Passive")
  ...
}

**IOC**

record(calcout, "control_voltage")
{
  field(INPA, "sensor MS CP")
  field(CALC, "A<10?5:0")
  field(OUT, "voltage PP")
  field(IVOA, "Don’t drive outputs")
}

Almost no additional work!

Anticipate network issues; see ‘MS’, ‘IVOA’
Automation via State Machine

“Sequencer”, “State Notation Language”: Event driven, on-demand, stateful
Automation via Scripts

• Tempting, but
  – Error Handling?
  – caget? Monitor!
  – caput? Connect once, then re-use the connection!
  – Handle disconnects, re-connects

• Should have ‘console’, run under procServ
  – IOC has shell; Calc record has CALC, SCAN, INPA, ..

• Long-term maintenance of “Fred’s script”?
Automation via User Interface

IOC
record(ai, “sensor”)
record(ao, “voltage”)

Channel Access

CSS ‘BOY’ Examples:

• Check allowed values
  – What if other CA client writes to PV?
  Use DRVH, DRVL, calc records, .. to perform check on IOC

• Start threads for automation scripts
  – What if users open multiple user interfaces?
  – What if GUI crashes (which is more likely than IOC)?

Keep user interface as just that!
Automation with EPICS

• Records
  – Steady-data, data flow driven operations
  – Continuous: Read input, compute, write output
  – Limited conditional processing: calcout.OOPT

• State Notation Language
  – On-demand, event driven
  – Stateful: In State X, if Y happens, ..

• Scripts
  – Useful for “I need this just once, but I need it now”
  – Permanent “Python IOCs” require effort similar to IOCs