



DevIOC

Easy EPICS Soft Records for Python

Michel Fodje
MX Beamlines, CLS



Canadian Light Source
Centre canadien de rayonnement synchrotron

EPICS softIOC

- EPICS binary “softIoc”
- Can be used to create EPICS IOC with Process Variables not linked to hardware
 - Define a *DB* file
 - Create a *CMD* file
 - Run softIOC:

```
softIoc test.cmd
```

```
record(mbbo, "$(device):number") {  
    field(DESC, "Enum Test")
```

```
$ softIoc test.cmd
```

```
dbLoadRecords("test.db", "device=MYIOC-123")  
iocInit()  
Starting iocInit  
#####  
## EPICS R7.0.4.1  
## Rev. 2020-10-17T12:07-0600  
#####  
iocRun: All initialization complete  
dbl  
MYIOC-123:calc  
MYIOC-123:toggle  
MYIOC-123:intval  
MYIOC-123:number  
MYIOC-123:lstring  
MYIOC-123:intarray  
MYIOC-123:floatarray  
MYIOC-123:floatval  
MYIOC-123:floatout  
MYIOC-123:sstring  
iocInit()  
epics>
```

```
}  
----- "$(device):floatval" }  
}
```

Limitations

- DB file is un-intuitive for EPICS newbies
- Not very useful as is
 - Needs a separate program to handle logic

```
michel@fedora:tmp/junk/bin/_dbcache_
## load record instances
dbloadRecords("Junk.db", "device=MYIOC-123")
iocInit()
Starting iocInit
#####
## EPICS R7.0.4.1
## Rev. 2020-10-17T12:07-0600
#####
iocRun: All initialization complete
db1
MYIOC-123:calc
MYIOC-123:toggle
MYIOC-123:intval
MYIOC-123:number
MYIOC-123:lstring
MYIOC-123:intarray
MYIOC-123:floata
MYIOC-123:floatv
MYIOC-123:floato
MYIOC-123:ssstring
# End Junk.cmd
iocInit()
iocBuild: IOC ca
epics>
```

```
michel@fedora:tmp
[michel@localhost tmp]$ vi test_logic.py
[michel@localhost tmp]$ python test_logic.py
Monitoring PV and performing logic operations
```

Why DevIOC?

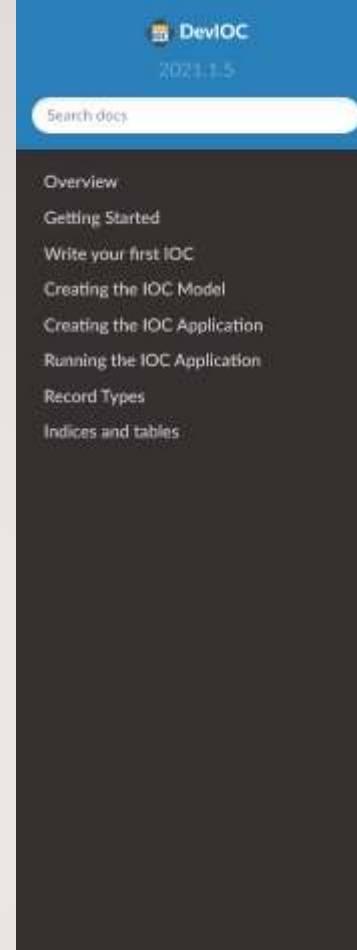
- Do everything in Python
 - Defining Process variables
 - Implementing the logic
 - Running the application
- Easy to use
- Works on Windows & Linux
- Small code-base:
 - *500 sloc*, documented.

Use Cases

- Access “Singleton” Applications from multiple locations
- Modularization/Separation of concerns
- Python-based device drivers
- Add PV interfaces to existing python application
- Use EPICS as transport for multi-client RPC

Resources

- Code:
 - <https://github.com/michel4j/devioc/>
- Documentation:
 - <https://michel4j.github.io/devioc/>



Overview

DevIOC is a package which enables python based EPICS IOC Soft Device support all within python. It allows you to define the IOC database model in a manner similar to Django Database models, and to use the model to develop dynamic, IOC servers.

To use the full capabilities, it is highly recommended to use a GObject compatible main loop, such as the one provided by PyGObject or even better, the GObject compatible Twisted reactor.

This library has been used to support very complex network IOC devices with non-trivial communication protocols. It works!

Getting Started

Before you can use DevIOC, you'll need to install it and its dependencies. We recommend installing it inside a virtual environment using the following commands on the shell

```
$ python -m venv devioc  
$ source devioc/bin/activate
```

```
$ python -m venv devioc  
$ source devioc/bin/activate  
(devioc) $ pip install devioc  
(devioc) $ devioc-startproject myioc  
(devioc) $ myioc/bin/app.server --device MYIOC-123
```

Dependencies:

- Twisted (PyPI)
- PyGObject (PyPI*)
- pyepics (PyPI)
- gepics (PyPI) – PyGObject Wrapper for PyEPICS

\$ *devioc-startproject* **myioc**

```
myioc
├── bin
│   └── app.server          # Command to run IOC Application
├── deploy
│   └── init-template      # Sample deployment script
├── myioc                   # Package for Application and supporting modules
│   ├── __init__.py
│   └── ioc.py             # IOC module containing your IOC application
├── README.md
└── setup.py
```

app.server

```
#!/usr/bin/env python
import os
import logging
import sys
import argparse

# Twisted boiler-plate code.
from twisted.internet import gireactor
gireactor.install()
from twisted.internet import reactor

# add the project to the python path and inport it
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from devioc import log
from myioc import ioc

# Setup command line arguments
parser = argparse.ArgumentParser(description='Run IOC Application')
parser.add_argument('--verbose', action='store_true', help='Verbose Logging')
parser.add_argument('--device', type=str, help='Device Name', required=True)

if __name__ == '__main__':
    args = parser.parse_args()
    if args.verbose:
        log.log_to_console(logging.DEBUG)
    else:
        log.log_to_console(logging.INFO)

    # initialize App
    app = ioc.MyIOCApp(args.device)

    # make sure app is properly shutdown
    reactor.addSystemEventTrigger('before', 'shutdown', app.shutdown)

    # run main-loop
    reactor.run()
```



device=args.device

ioc.py

```
$(device):enum  
$(device):toggle  
$(device):sstring  
$(device):lstring  
$(device):intval  
$(device):floatval  
$(device):floatout  
$(device):intarray  
$(device):floatarray  
$(device):calc
```

```
from devioc import models
```

```
class MyIOC(models.Model):
```

```
    enum = models.Enum('enum', choices=['ZERO', 'ONE', 'TWO'], default=0, desc='Enum Test')  
    toggle = models.Toggle('toggle', zname='ON', oname='OFF', desc='Toggle Test')  
    sstring = models.String('sstring', max_length=20, desc='Short String Test')  
    lstring = models.String('lstring', max_length=512, desc='Long String Test')  
    intval = models.Integer('intval', max_val=1000, min_val=-1000, default=0, desc='Int Test')  
    floatval = models.Float(  
        'floatval', max_val=1e6, min_val=1e-6, default=0.0,  
        prec=5, desc='Float Test'  
    )  
    floatout = models.Float('floatout', desc='Test Float Output')  
    intarray = models.Array('intarray', type=int, length=16, desc='Int Array Test')  
    floatarray = models.Array('floatarray', type=float, length=16, desc='Float Array Test')  
    calc = models.Calc(  
        'calc', calc='A+B',  
        inpa='$(device):intval CP NMS',  
        inpb='$(device):floatval CP NMS',  
        desc='Calc Test'  
    )
```


ioc.py

```
$(device):toggle  
ioc.toggle  
do_toggle
```

```
class MyIOApp(object):  
  
    def __init__(self, device_name):  
        self.ioc = MyIOC(device_name, callbacks=self)  
  
    def do_toggle(self, pv, value, ioc):  
        """  
        I am called whenever the `toggle` record's value changes  
        """  
        if value == 1:  
            # Command activated, value will return to 0 after some time  
            print('Toggle Changed Value', value)  
            ioc.enum.put((ioc.enum.get() + 1) % 3, wait=True) # cycle through values  
  
    def do_enum(self, pv, value, ioc):  
        print('New Enum Value', value)  
  
    def shutdown(self):  
        # needed for proper IOC shutdown  
        self.ioc.shutdown()
```

devioc.models.Enum
devioc.models.BinaryInput
devioc.models.BinaryOutput
devioc.models.Toggle
devioc.models.Integer
devioc.models.Float
devioc.models.String
devioc.models.Array
devioc.models.Calc
devioc.models.CalcOut

Record Types

Examples



Canadian Light Source
Centre canadien de rayonnement synchrotron

Look-Up-Table (LUT)

- Change **target** PV based on changes to **input** PV value using a LUT.
- Supports interpolations
- Allow updating/saving values on the fly

```
1
2 from enum import Enum
3 from devioc import models, log
4 from scipy import interpolate
5 from datetime import datetime
6 import numpy
7 import glob
8 import os
9
10
11 class FitType(Enum):
12     LINEAR = 0
13     NEAREST = 1
14     NEXT = 2
15     PREVIOUS = 3
16     SPLINE0 = 4
17     SPLINE1 = 5
18     SPLINE2 = 6
19     SPLINE3 = 7
20     POLY = 8
21
22
23
24 COARSE_SIZE = 120
25 FINE_SIZE = 255
26
27 logger = log.get_module_logger(__name__)
28
29
30 class EpicsLUT(models.Model):
31     target = models.Float('target', desc='Target Value')
32     output = models.Float('output', desc='Target Output')
33     nocontrol = models.Float('null', desc='Null Output')
34
35     newvalue = models.Float('new', desc='New Target')
36     xoff = models.Float('xoff', desc='X Offset')
37     yoff = models.Float('yoff', desc='Y Offset')
38
```

Look-Up-Table (LUT)

- Change **target** PV based on changes to **input** PV value using a LUT.
- Supports interpolations
- Allow updating/saving values on the fly

The image displays three overlapping screenshots of a software interface for Look-Up-Tables (LUTs).

Top Window: Look-Up-Table: LUT1608-BM-PITCH

- Input Value: 9.4829
- Graph: Output vs. Input (Y-axis: 0.26 to 0.31, X-axis: 0 to 10)
- X-Offset: 0.0000
- Y-Offset: 0.0000
- Type of fit: POLY
- Energy/Pitch

Middle Window: Look-Up-Table: LUT1608-BM-IONC

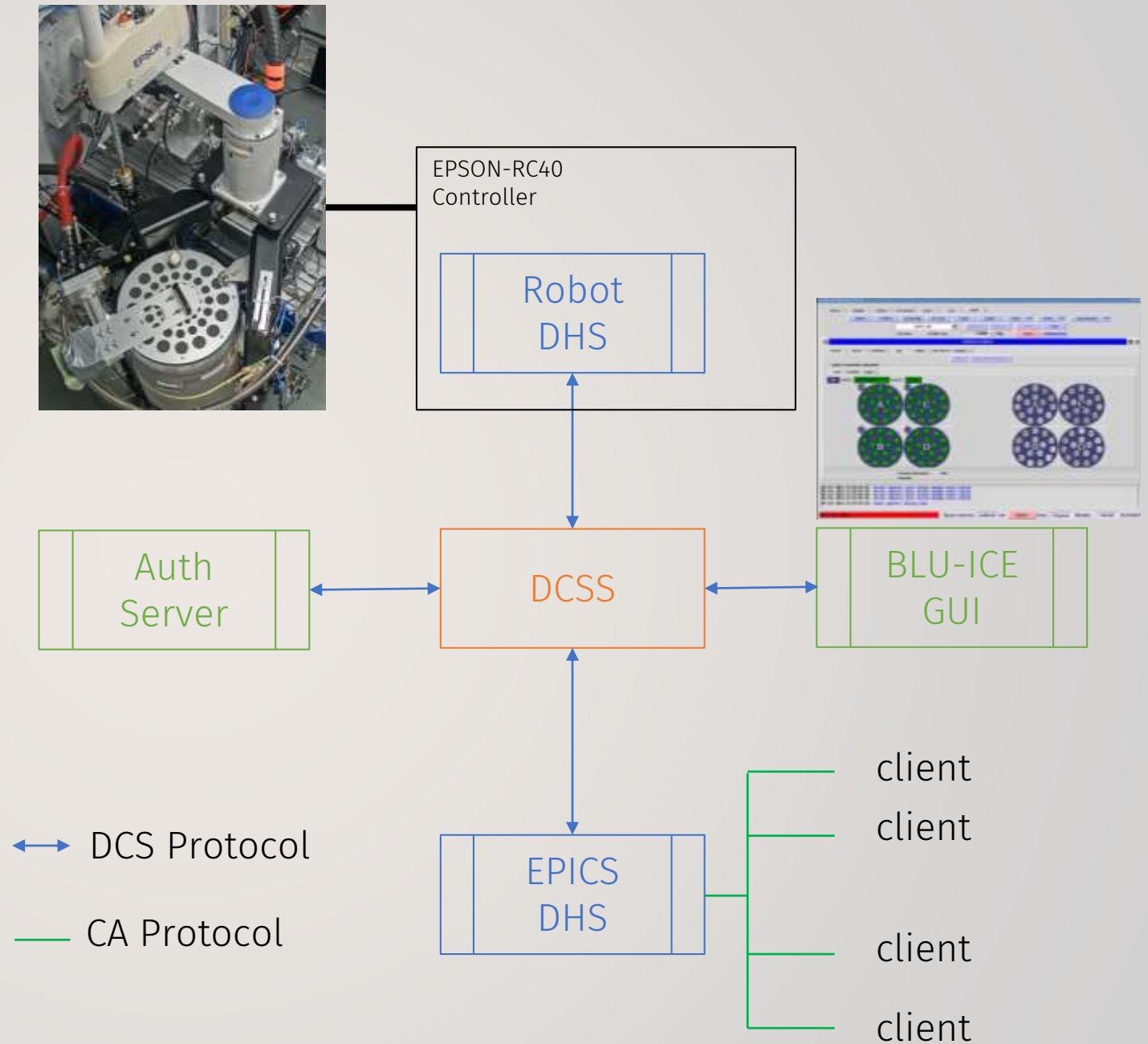
- Input Value: 8.1498
- Graph: Output vs. Input (Y-axis: -200 to 1200, X-axis: 4 to 6)
- X-Offset: 0.0000
- Y-Offset: 0.0000
- Type of fit: SPLINE
- Energy-I/Current

Bottom Window: Look-Up-Table: LUT1608-BM-BOSS

- Input Value: 1.5700
- Output Value: 0.0100
- New Output Value: 0.0000 (highlighted in yellow)
- Update button
- Graph: Output vs. Input (Y-axis: 0 to 0.05, X-axis: 0 to 20)
- X-Offset: 0.0000
- Y-Offset: 0.0000
- Type of fit: SPLINE
- Input Source: SMTR1608-4-B10-17:deg CPP NMS
- Control Target: BL08B1:PicoControl:SetYOUT PP NMS
- Apply Automatically: ON
- Save and Reload buttons
- Energy/Boss

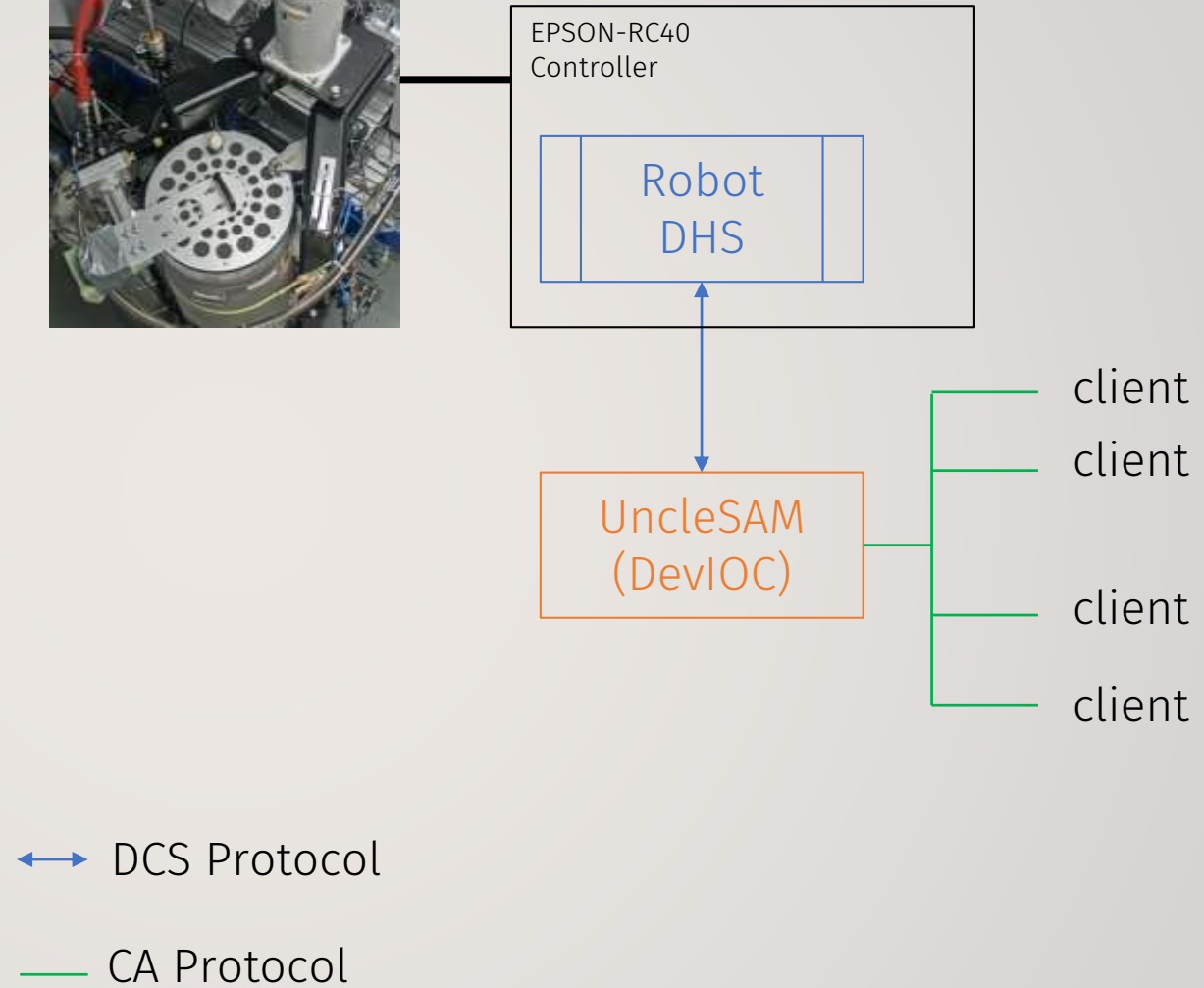
Stanford Auto Mounter (SAM)

- SAM Robot Control
 - Blu-ICE/DCSS/DHS control system for SAM Robot (SSRL)
 - Complicated, multiple services, Tcl/Tk/Incr
 - Full duplicate control system just to use RobotDHS
 - Difficult to maintain/update



UncleSAM

- Speaks DCS protocol to DHS
- Directly provides EPICS PVs for CA clients
- Easy to deploy
- We can make changes to DHS without worrying about DCSS ecosystem



unclesam/dcs.py

- Implementation of DCS Protocol using Twisted Framework

```
55
56 class RobotProtocol(protocol.Protocol):
57     """DCS Protocol"""
58
59     def __init__(self, factory):
60         self.factory = factory
61         self.ready = False # True after successful handshake
62         self.data = ""
63
64     def connectionMade(self):
65         reactor.addSystemEventTrigger('before', 'connect',
66
67         try:
68
```

```

69         self.factory.receive_message(message)
70         logger.log(log.INFO, 'SAM Ready: {}'.format(reason.getErrorMessage()))
71
72     def dataReceived(self, data):
73         """
74         Called when data is received from hardware DHS
75         :param data:
76         :return:
77         """
78
79         self.data += data.decode('utf-8')
80
81         if re.match(r'^\s+\d+\s+0[\0\s]*$', self.data[:26]):
82             # DCS2 message
83             size = int(self.data[:26].split()[0]) + 26
84
85             # allow for multiple dcs2 short messages
86             while re.match(r'^\s+\d+\s+0[\0\s]*$', self.data[:26]) and len(self.data) >= size:
87                 self.receive_message(dcs2_to_text(self.data[:size]))
88                 self.data = self.data[size:]
89                 if re.match(r'^\s+\d+\s+0[\0\s]*$', self.data[:26]):
90                     size = int(self.data[:26].split()[0]) + 26
91
92         elif len(data) == 200:
93
94         self.factory.receive_message(message)
```

```
125
126 class RobotFactory(protocol.ServerFactory):
127     protocol = RobotProtocol
128
129     def __init__(self, application):
130         self.application = application
131         self.ready = False
132         self.client = None
133
134     def buildProtocol(self, address):
135         logger.log(log.IMPORTANT, 'SAM Ready: {}'.format(address))
136         self.client = self.protocol(self)
137         return self.client
138
```


unclesam/ioc.py

```
37 class Robot(models.Model):
38     connected = models.Enum('CONNECTED', choices=('Inactive', 'Active'), default=0, desc="Connection")
39     enabled = models.Enum('ENABLED', choices=('Disabled', 'Enabled'), default=1, desc="Control")
40     heartbeat = models.Enum('HEARTBEAT', choices=('TICK', 'TOCK'), default=0, desc="Heartbeat")
41     normal = models.Enum('HEALTH', choices=('Abnormal', 'Normal'), desc="Health")
42     status = models.Enum('STATUS', choices=StatusType, desc="Status")
43     log = models.String('LOG', desc="Sample Operation Message", max_length=1024)
44     log_alarm = models.Enum('LOG:ALARM', choices=LogType, desc="Log Level")
45     sample_log = models.String('MESSAGE', desc="Sample Log", max_length=512)
46     handle = models.Integer('HANDLE', desc="Last Operation", default=1)
47
48     # Safety flags
49     prepare = models.Enum('SAFETY:PREPARE', choices=OnOffType, desc="Prepare for Approach")
50     gonis_ready = models.Enum('SAFETY:READY', choices=OnOffType, desc="Ready for Approach")
51     approach_on = models.Enum('SAFETY:APPROACH:ON', choices=OnOffType, out="$(approach_on) PP NMS", desc="Approaching Gonis")
52     approach_off = models.Enum('SAFETY:APPROACH:OFF', choices=OnOffType, out="$(approach_off) PP NMS", desc="Left Gonis")
53
54     ports = models.String('PORTS', max_length=256, desc="Port States")
55     ports_left = models.Array('PORTS:L', type='SHORT', length=96, desc="L Port States")
56     ports_middle = models.Array('PORTS:M', type='SHORT', length=96, desc="M Port States")
57     ports_right = models.Array('PORTS:R', type='SHORT', length=96, desc="R Port States")
58     cassette_left = models.Enum('CASSETTE:L', choices=CassetteType, desc="L Cassette")
59     cassette_middle = models.Enum('CASSETTE:M', choices=CassetteType, desc="M Cassette")
60     cassette_right = models.Enum('CASSETTE:R', choices=CassetteType, desc="R Cassette")
61
62     le = models.Array('LE:L', type='SHORT', length=96, desc="L Le")
63     le_middle = models.Array('LE:M', type='SHORT', length=96, desc="M Le")
64     le_right = models.Array('LE:R', type='SHORT', length=96, desc="R Le")
65
66     # Control
67     control = models.Enum('CONTROL', choices=ControlType, desc="Control")
68     control_left = models.Enum('CONTROL:L', choices=ControlType, desc="L Control")
69     control_middle = models.Enum('CONTROL:M', choices=ControlType, desc="M Control")
70     control_right = models.Enum('CONTROL:R', choices=ControlType, desc="R Control")
71
72     # Status
73     status = models.Enum('STATUS', choices=StatusType, desc="Status")
74     status_left = models.Enum('STATUS:L', choices=StatusType, desc="L Status")
75     status_middle = models.Enum('STATUS:M', choices=StatusType, desc="M Status")
76     status_right = models.Enum('STATUS:R', choices=StatusType, desc="R Status")
77
78     # Heartbeat
79     heartbeat = models.Enum('HEARTBEAT', choices=HeartbeatType, desc="Heartbeat")
80     heartbeat_left = models.Enum('HEARTBEAT:L', choices=HeartbeatType, desc="L Heartbeat")
81     heartbeat_middle = models.Enum('HEARTBEAT:M', choices=HeartbeatType, desc="M Heartbeat")
82     heartbeat_right = models.Enum('HEARTBEAT:R', choices=HeartbeatType, desc="R Heartbeat")
83
84     # Connection
85     connected = models.Enum('CONNECTED', choices=ConnectedType, desc="Connection")
86     connected_left = models.Enum('CONNECTED:L', choices=ConnectedType, desc="L Connection")
87     connected_middle = models.Enum('CONNECTED:M', choices=ConnectedType, desc="M Connection")
88     connected_right = models.Enum('CONNECTED:R', choices=ConnectedType, desc="R Connection")
89
90     # Enabled
91     enabled = models.Enum('ENABLED', choices=EnabledType, desc="Enabled")
92     enabled_left = models.Enum('ENABLED:L', choices=EnabledType, desc="L Enabled")
93     enabled_middle = models.Enum('ENABLED:M', choices=EnabledType, desc="M Enabled")
94     enabled_right = models.Enum('ENABLED:R', choices=EnabledType, desc="R Enabled")
95
96     # Log
97     log = models.String('LOG', desc="Sample Operation Message", max_length=1024)
98     log_alarm = models.Enum('LOG:ALARM', choices=LogType, desc="Log Level")
99     sample_log = models.String('MESSAGE', desc="Sample Log", max_length=512)
100
101     # Handle
102     handle = models.Integer('HANDLE', desc="Last Operation", default=1)
```

~100 Records

```
class RobotIOApp(object):
```

```
def __init__(self, device, approach_on='', approach_off=''):
```

```
    """Internal State of the Server"""
```

```
    self.robot = Robot(device, callbacks=self, macros={'approach_on': approach_on, 'approach_off': approach_off})
```

```
    self.operations = {}
```

```
    self.handle = 1
```

```
    self.ready = False
```

```
    self.inbox = Queue()
```

```
    self.outbox = Queue()
```

```
    self.send_on = False
```

```
    self.recv_on = False
```

```
    self.soak_on = False
```

```
    self.status_ready = False
```

```
    self.last_mount_time = time.time()
```

```
    self.setup()
```

```
    self.client = dcs.RobotFactory(self)
```

```
    reactor.listenTCP(14242, self.client)
```

edm Operator Screen

SAM Automounter

Disabled
Enabled

Robot Connection ACTIVE	Robot Status IDLE	Robot Health Abnormal	Cryo Mode LN2 Mode	Last Operation 557	Robot Control Enabled
Robot Arm Position PD	Sample State ON PLACER	Magnet State IN CRADLE	Prefetched LD3	Current Port	Pins Mounted 14069

Mount
Prefetch
Home

Dismount
Soak
Dry

Abort

ERRORS

- Emerg stop
- Safeguard
- Not home
- Cmd Error
- Lid
- Gripper
- No Magnet
- Collision
- Init Error
- Toolset
- LN2 Level
- Heater
- Cassette
- Pin lost
- Wrong State
- Invalid Arg

Inspected

NEEDS

- Inspection
- Reset
- Tool Cal
- Cassette Cal
- Gonio Cal
- Basic Cal
- User Action

L Adaptor

M Empty

R Calib

Probe
Clear Ports

204.8 C

AutoFill ON

LN2 Val CLOSED

NOT_FULL

NOT_LOW

INPUTS OUTPUTS

LN2 Closed

LN2 Level

Autofill OFF

Gripper open

Gripper closed

Lid closed

Lid Open

Heater hot

Gripper

Lid

Approaching Gonio

Prepare for Approach

Ready for Approach

Lid
Heater

Gripper
Dryer

Dryer Heater

Check Gripper
Check Lid
Check Toolset
Calibrate Toolset
Teach Gonio
Save Gonio
RT Mode
Check Heater
L
M
R
Calibrate Cassette
Gonio Home
Calibrate Gonio
LN2 Mode

Reaheat Timeout	0
Sample Countdown	16
Home Timeout	0
Reset Timeout	0 sec
Last Duration	32 sec

<input type="checkbox"/> Check Forces	<input type="checkbox"/> Open Lid on Fill	<input type="checkbox"/> Wash on Mount
<input checked="" type="checkbox"/> Probe Cassette	<input type="checkbox"/> Check Picker	<input checked="" type="checkbox"/> High Speed in Dewar
<input type="checkbox"/> Probe Port	<input type="checkbox"/> Dev Mode	<input checked="" type="checkbox"/> Reheat Tong
<input type="checkbox"/> Check Magnet	<input checked="" type="checkbox"/> Strict Dismount	<input type="checkbox"/> Delay Calib

1200 Max Soak Time	0 Loss Threshold
16 Reheat Count	0 Strip Threshold
120 Reheat Time	30 Cooldown Time
3500 Max LN2 Idle	

Apply Attrs
idle
moving to Dewar
100.0 %

Aug/03 08:55:58 gripper cannot close at picker, maybe toolset calibration is off

Clear All

Raw

Thanks

