DevIOC

Easy EPICS Soft Records for Python

Michel Fodje

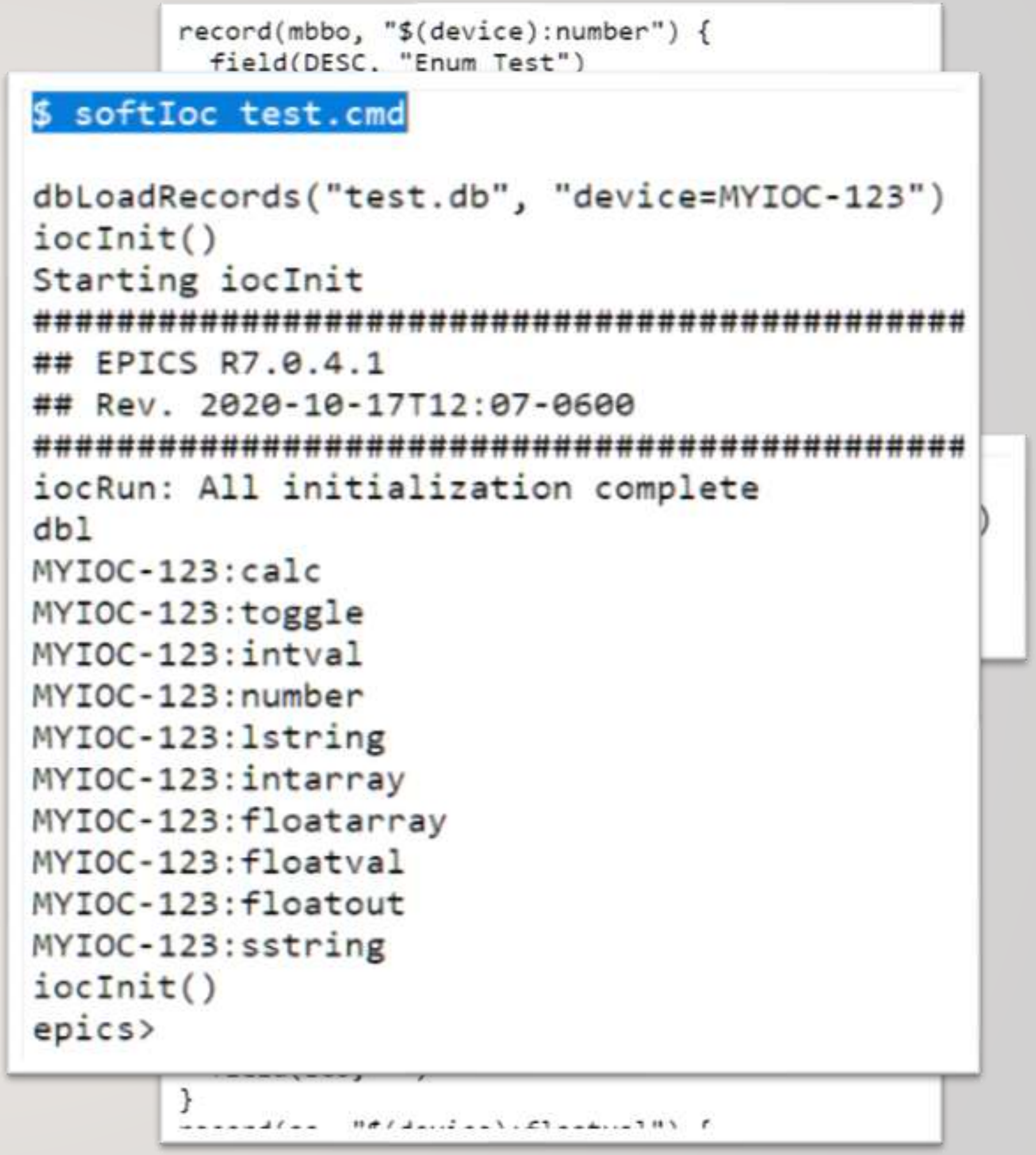
MX Beamlines, CLS

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
```



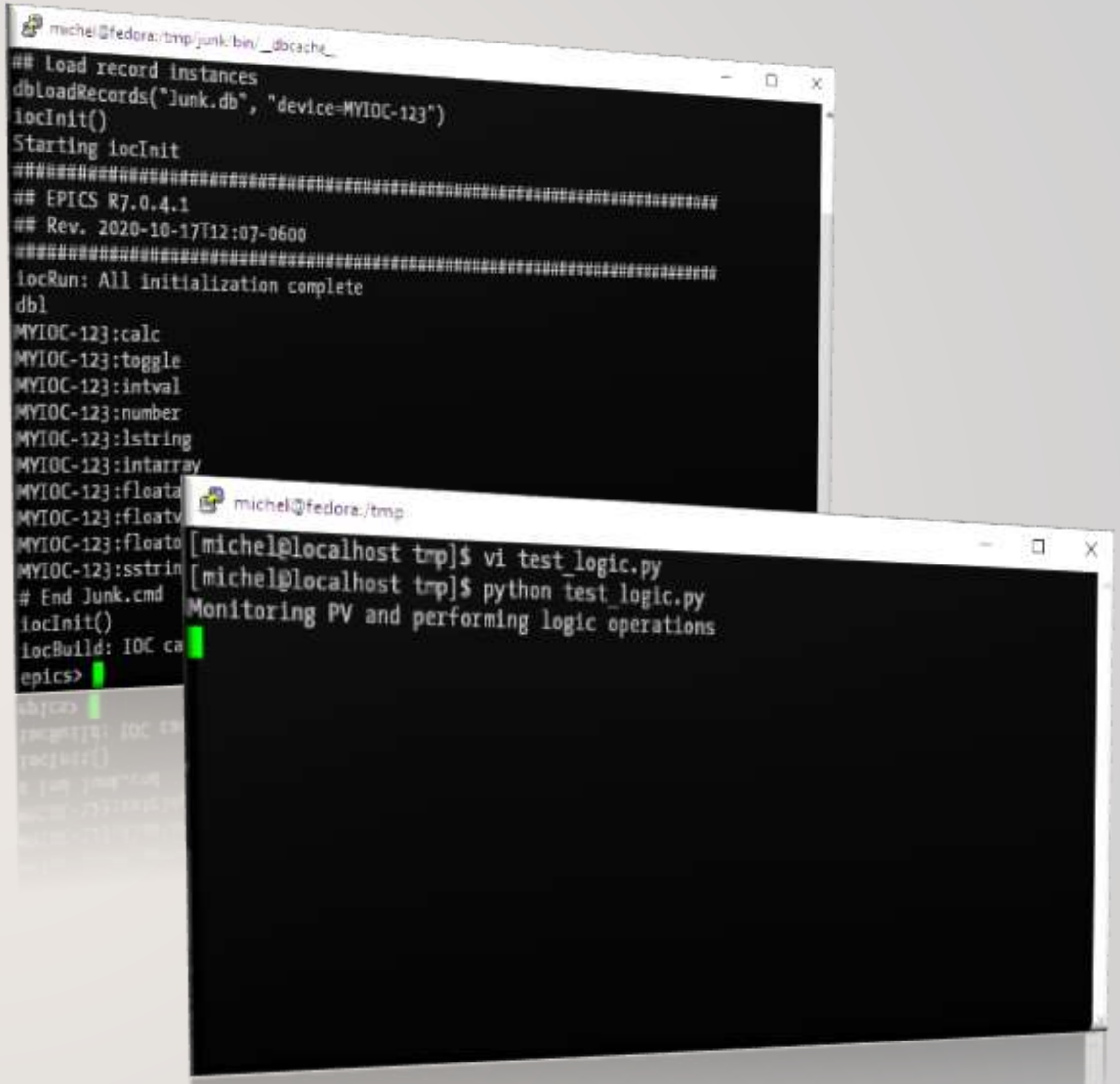
```
record(mbbo, "$(device):number") {
  field(DESC, "Enum Test")
```

```
$ 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>
```

```
}
```

Limitations

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



```
## 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
dbl
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:sstrin
# End Junk.cmd
iocInit()
iocBuild: IOC ca
epics>
```

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

Why DevIOC?

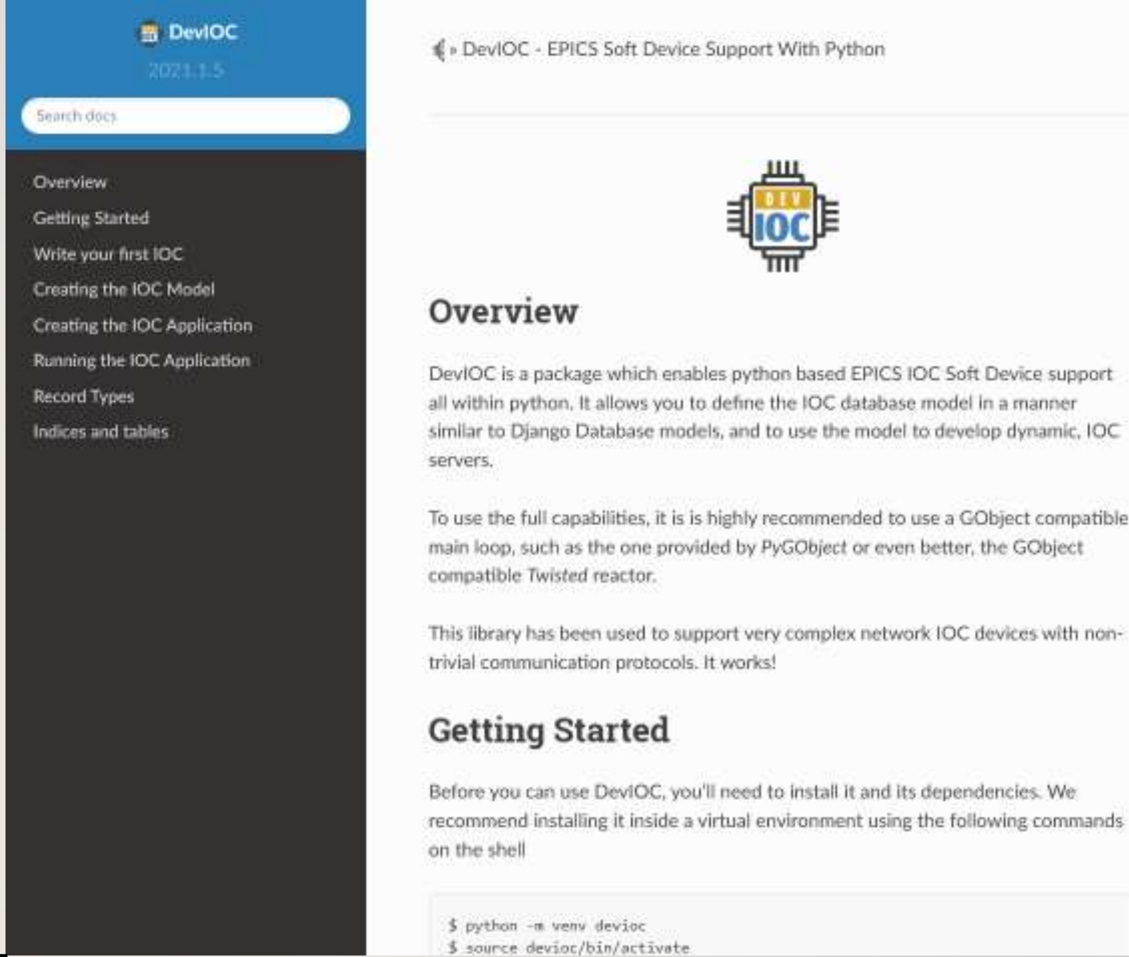
- Do everything in Python
  - Defining Process variables
  - Implementing the logic
  - Running the application
- Easy to use
- Works on Windows & Linux
- Small code-base:
  - *500 sloc*, documented.

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 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
```

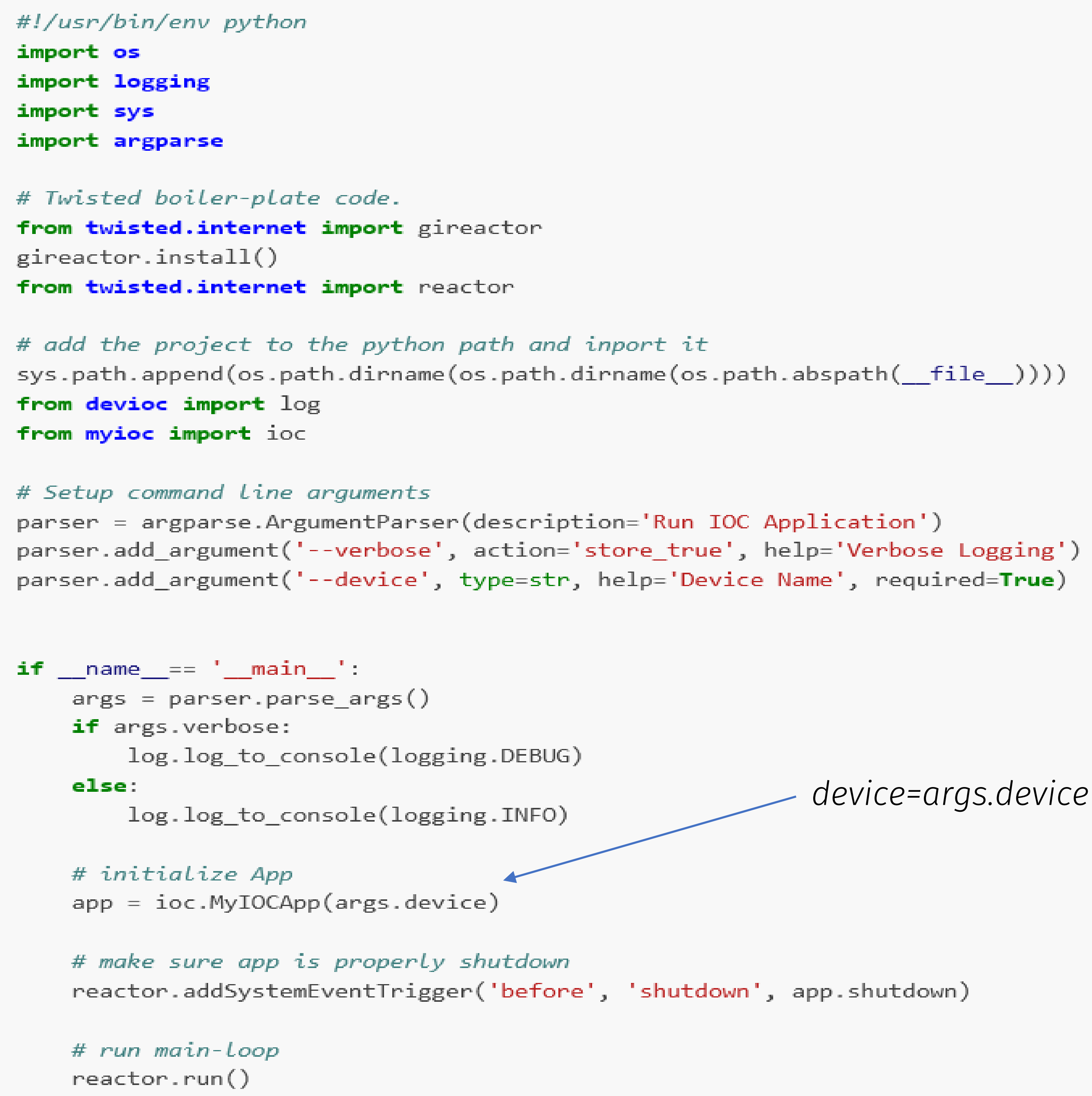
Dependencies:

- 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

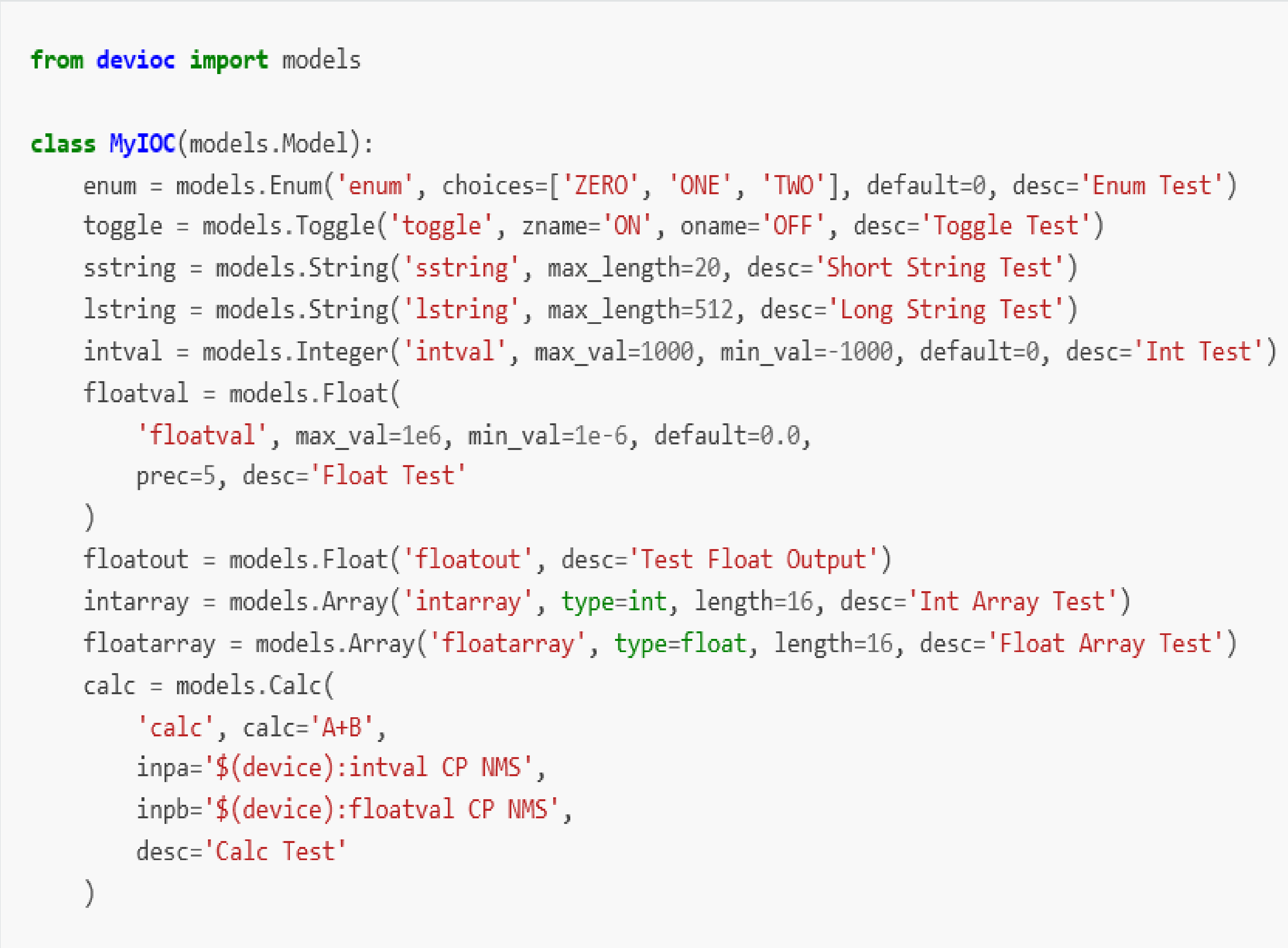


```python
#!/usr/bin/env python
import os
import logging
import sys
import argparse

# 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
```



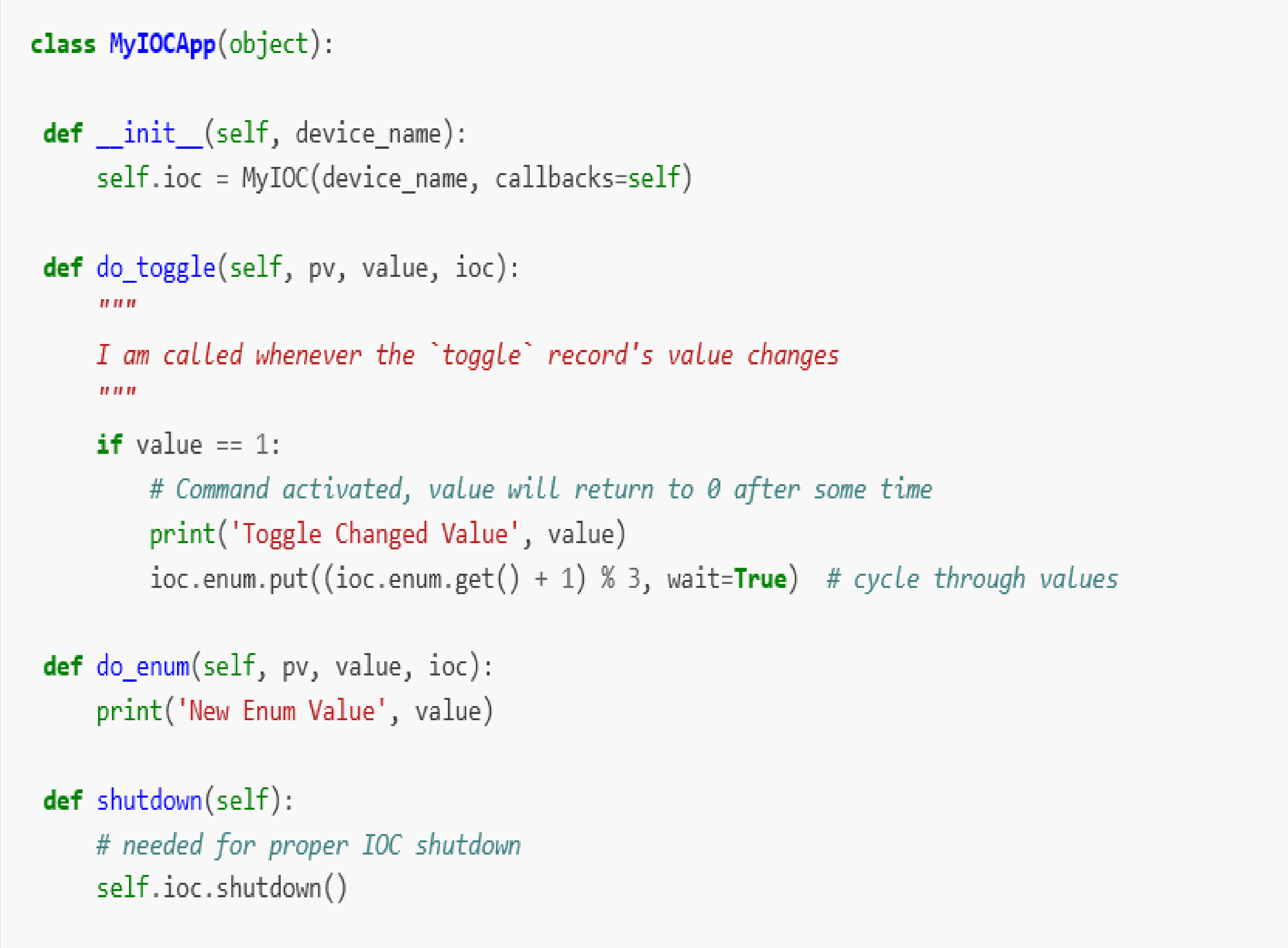
```python
from devioc import models

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 MyIOCApp(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()
```

Record Types

devioc.models.Enum

devioc.models.BinaryInput

devioc.models.BinaryOutput

devioc.models.Toggle

devioc.models.Integer

devioc.models.Float

devioc.models.String

devioc.models.Array

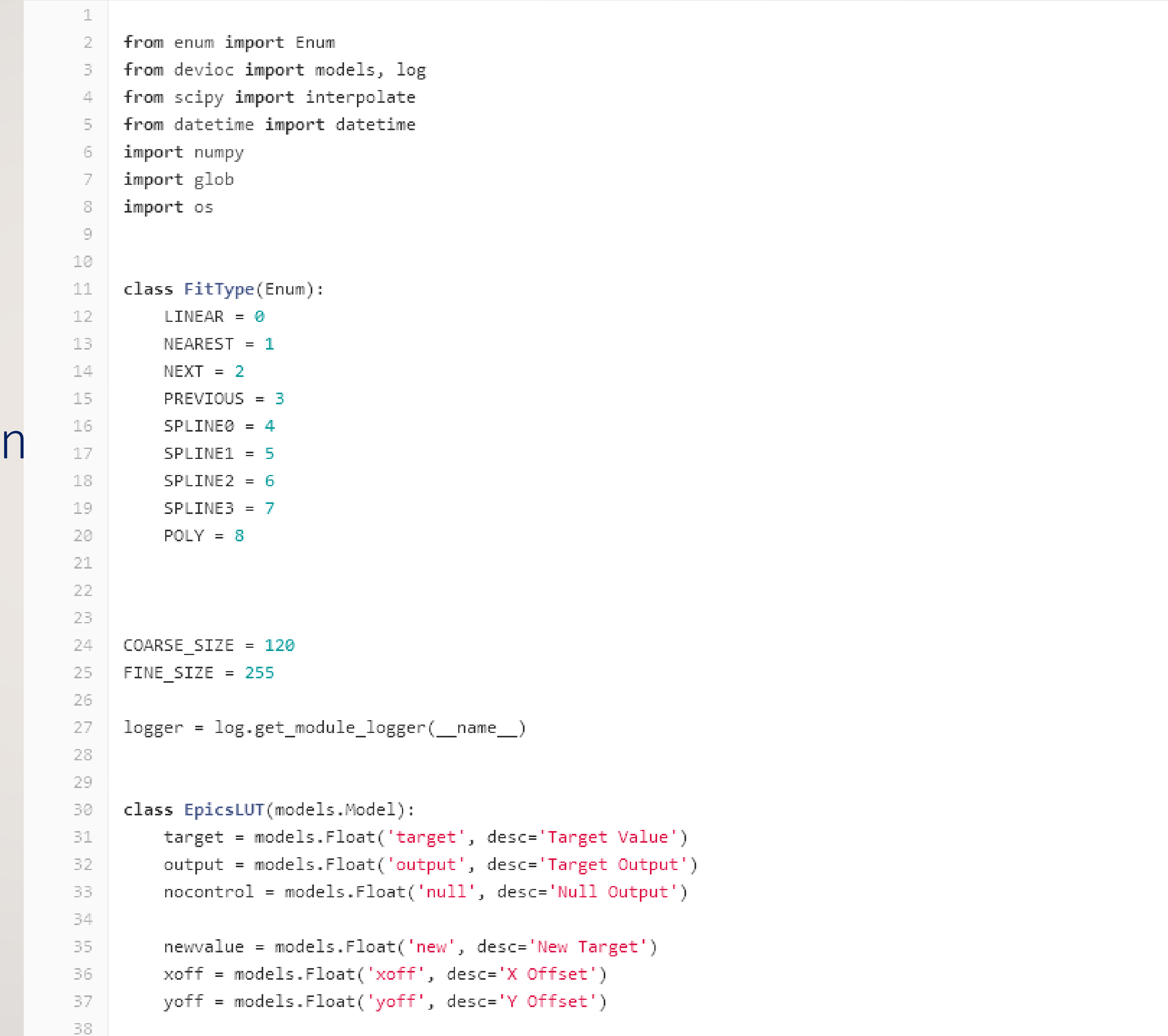
devioc.models.Calc

devioc.models.CalcOut

Examples

Look-Up-Table (LUT)

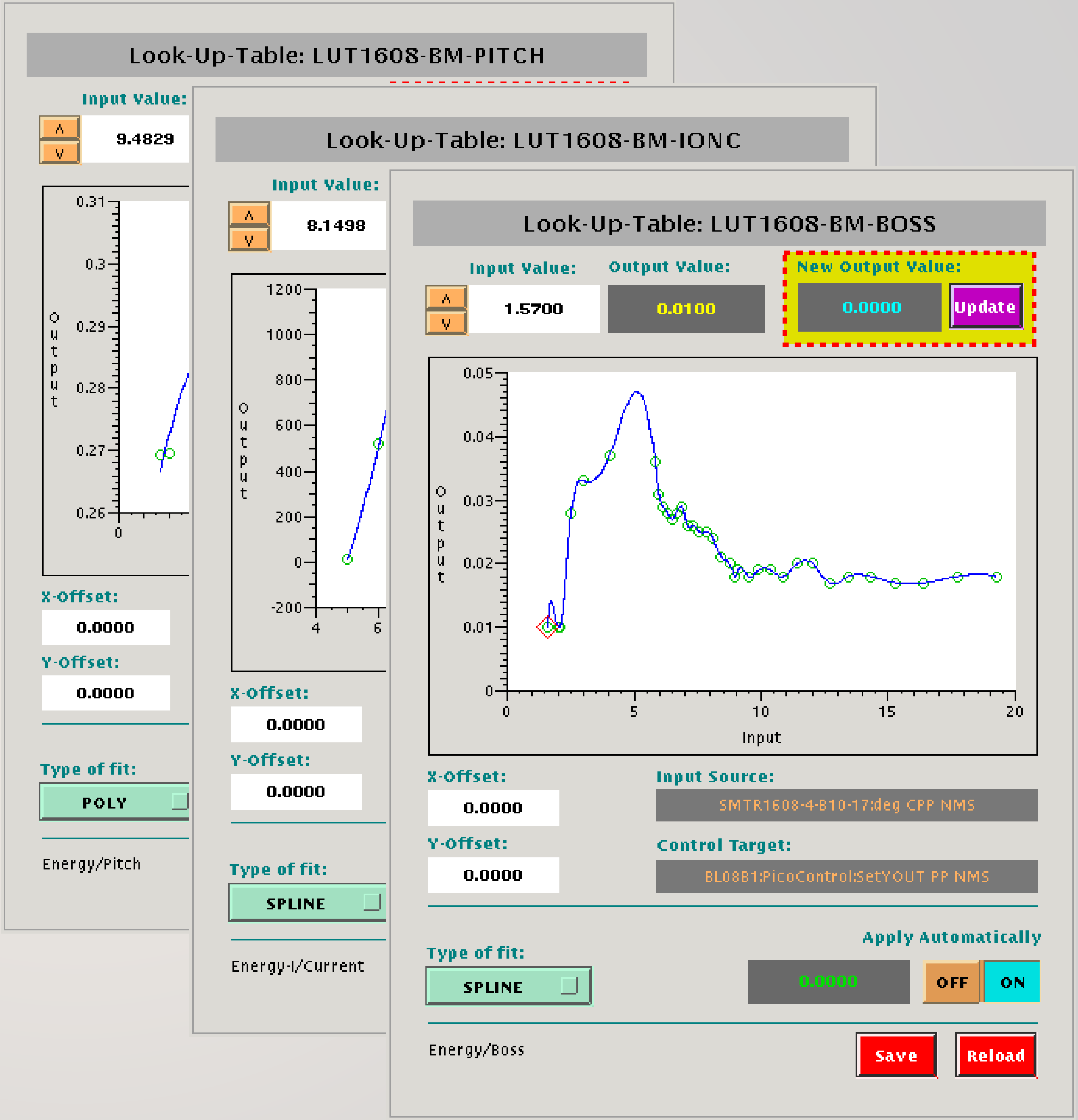
- Change **target** PV based on changes to **input** PV value usin a LUT.
- Supports interpolations
- Allow updating/saving values on the fly



```python

from enum import Enum
from devioc import models, log
from scipy import interpolate
from datetime import datetime
import numpy
import glob
import os


class FitType(Enum):
    LINEAR = 0
    NEAREST = 1
    NEXT = 2
    PREVIOUS = 3
    SPLINE0 = 4
    SPLINE1 = 5
    SPLINE2 = 6
    SPLINE3 = 7
    POLY = 8



COARSE_SIZE = 120
FINE_SIZE = 255

logger = log.get_module_logger(__name__)


class EpicsLUT(models.Model):
    target = models.Float('target', desc='Target Value')
    output = models.Float('output', desc='Target Output')
    nocontrol = models.Float('null', desc='Null Output')

    newvalue = models.Float('new', desc='New Target')
    xoff = models.Float('xoff', desc='X Offset')
    yoff = models.Float('yoff', desc='Y Offset')
```

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



Look-Up-Table: LUT1608-BM-PITCH

Input Value: 9.4829

X-Offset: 0.0000

Y-Offset: 0.0000

Type of fit: POLY

Energy/Pitch

Look-Up-Table: LUT1608-BM-IONC

Input Value: 8.1498

X-Offset: 0.0000

Y-Offset: 0.0000

Type of fit: SPLINE

Energy-I/Current

Look-Up-Table: LUT1608-BM-BOSS

Input Value: 1.5700

Output Value: 0.0100

New Output Value: 0.0000

X-Offset: 0.0000

Y-Offset: 0.0000

Input Source: SMTR1608-4-B10-17:deg CPP NMS

Control Target: BL08B1:PicoControl:SetYOUT PP NMS

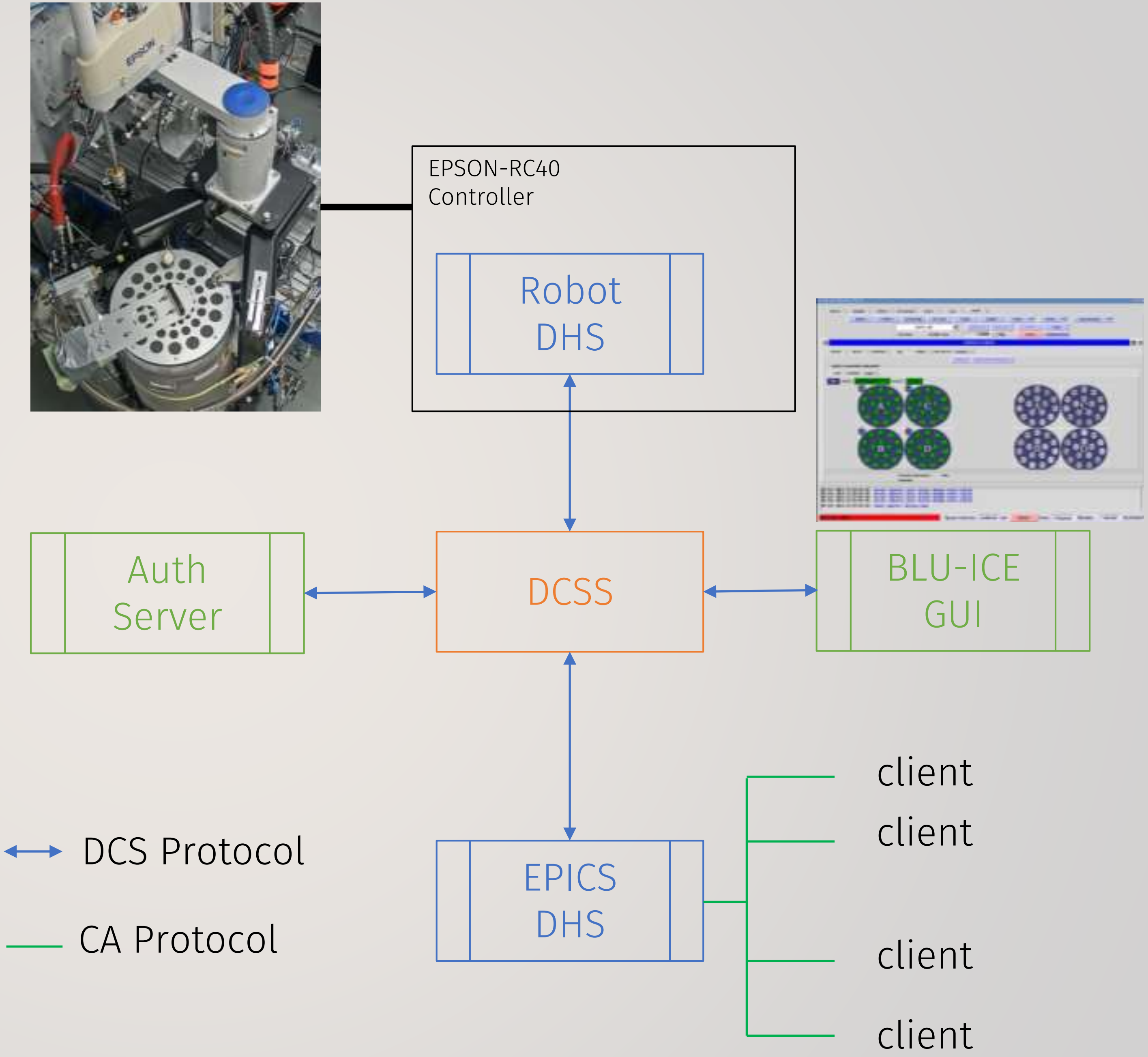
Apply Automatically: 0.0000

Type of fit: SPLINE

Energy/Boss

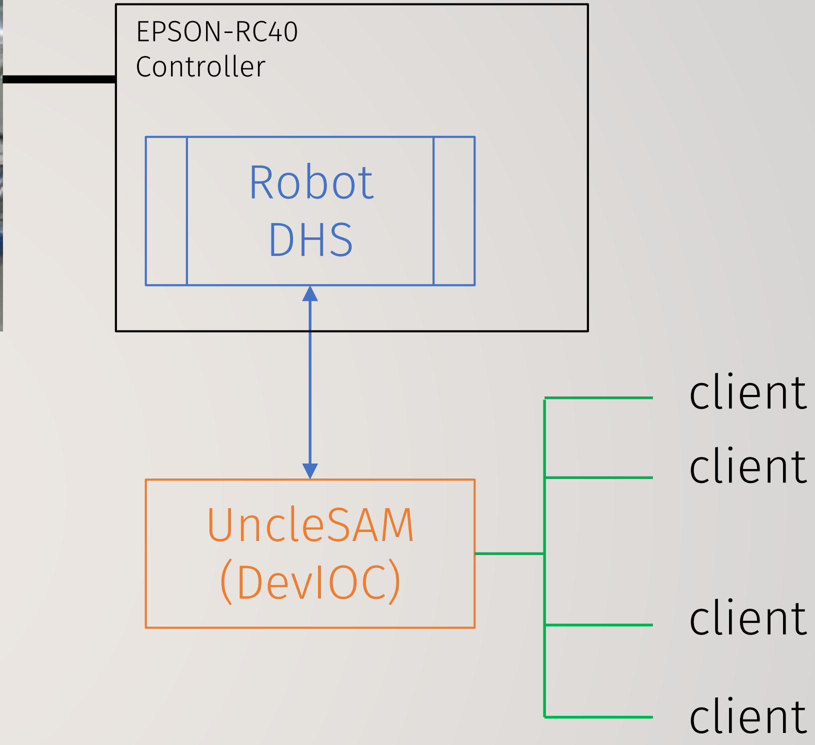
Stanford Auto Mounter (SAM)

- 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

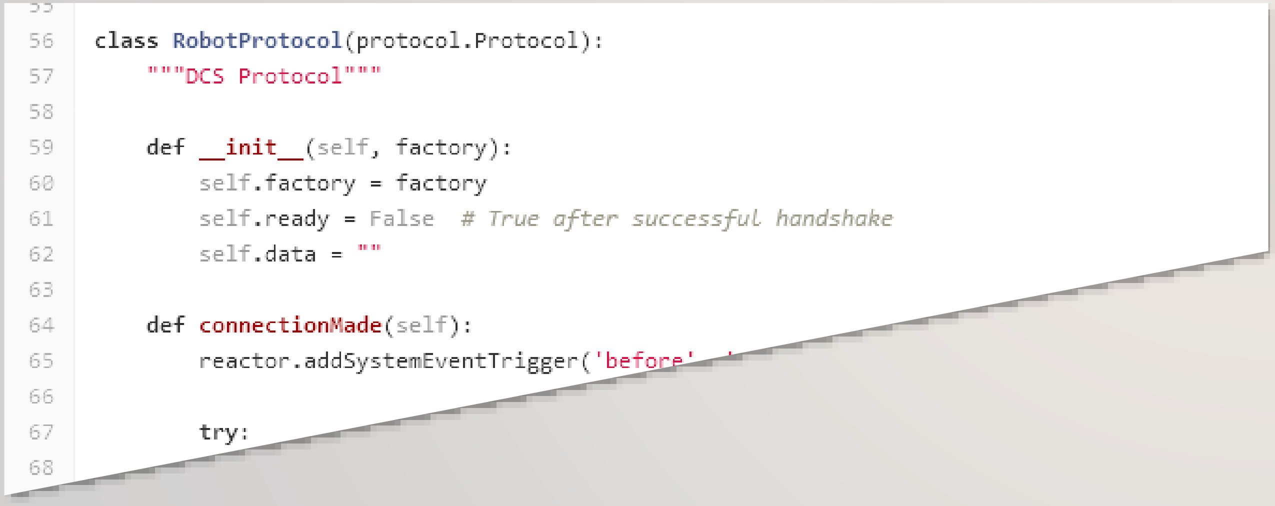


⟷ DCS Protocol

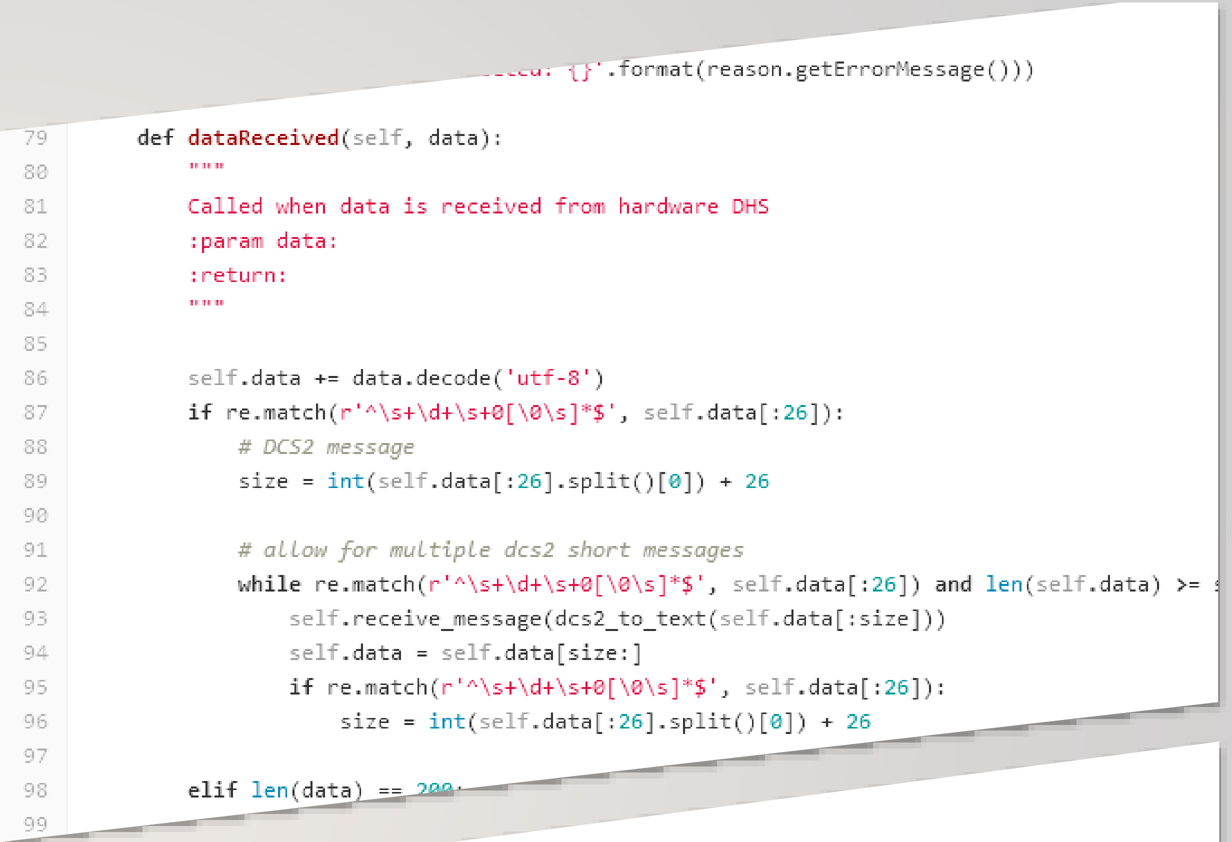
— CA Protocol

unclesam/dcs.py

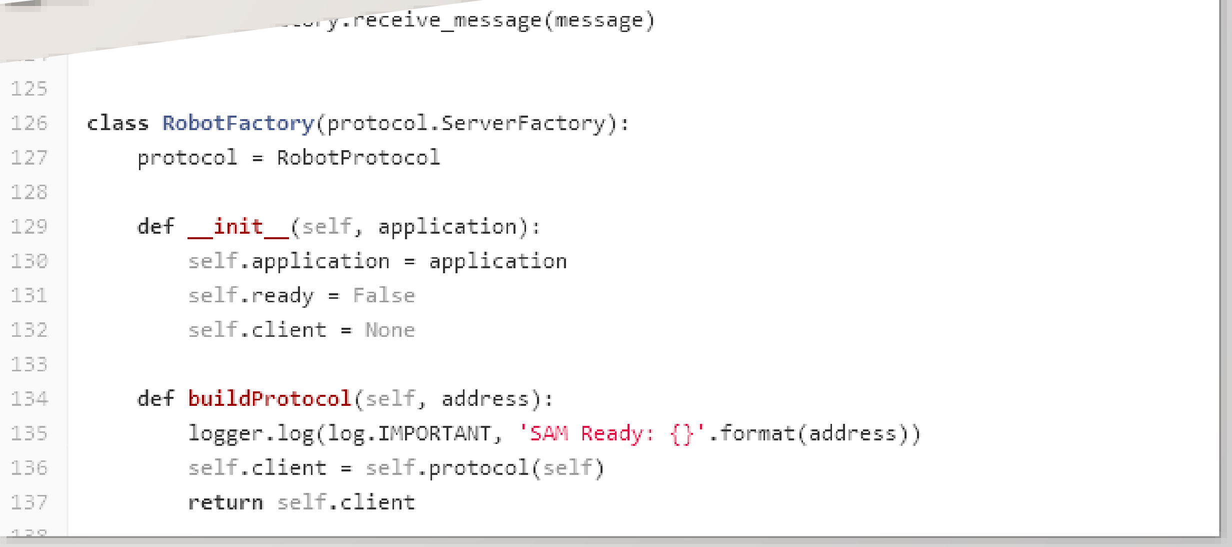
- Implementation of DCS Protocol using Twisted Framework



```python
class RobotProtocol(protocol.Protocol):
    """DCS Protocol"""

    def __init__(self, factory):
        self.factory = factory
        self.ready = False  # True after successful handshake
        self.data = ""

    def connectionMade(self):
        reactor.addSystemEventTrigger('before'

        try:

```

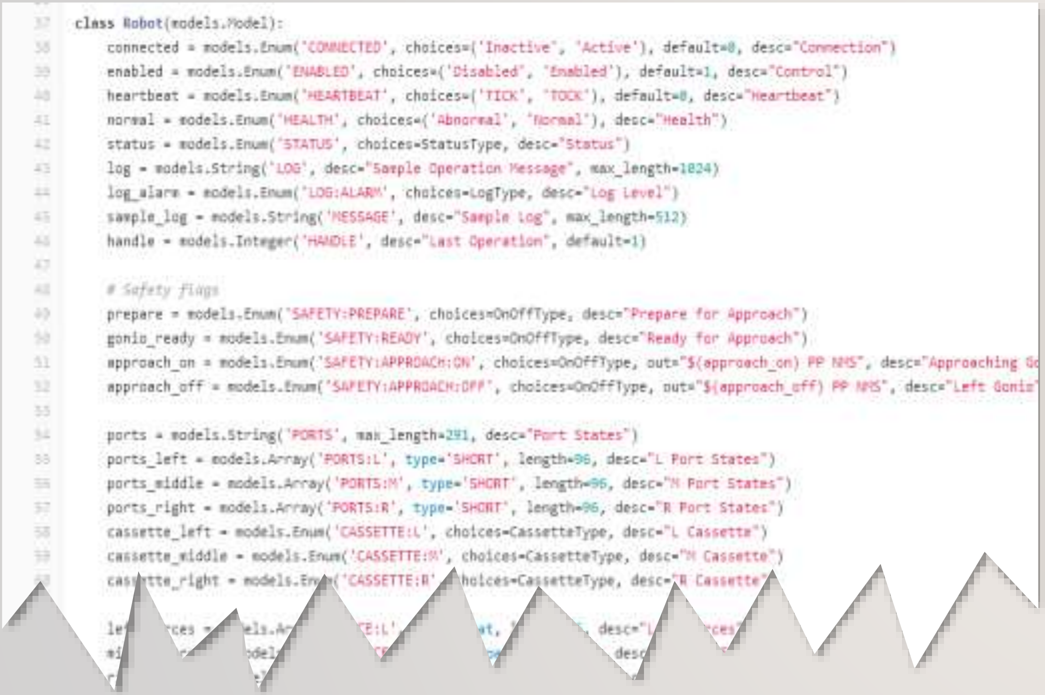


```python
                  {}'.format(reason.getErrorMessage()))

    def dataReceived(self, data):
        """
        Called when data is received from hardware DHS
        :param data:
        :return:
        """

        self.data += data.decode('utf-8')
        if re.match(r'^\s+\d+\s+0[\0\s]*$', self.data[:26]):
            # DCS2 message
            size = int(self.data[:26].split()[0]) + 26

            # allow for multiple dcs2 short messages
            while re.match(r'^\s+\d+\s+0[\0\s]*$', self.data[:26]) and len(self.data) >= s
                self.receive_message(dcs2_to_text(self.data[:size]))
                self.data = self.data[size:]
                if re.match(r'^\s+\d+\s+0[\0\s]*$', self.data[:26]):
                    size = int(self.data[:26].split()[0]) + 26

        elif len(data) == 200:

```



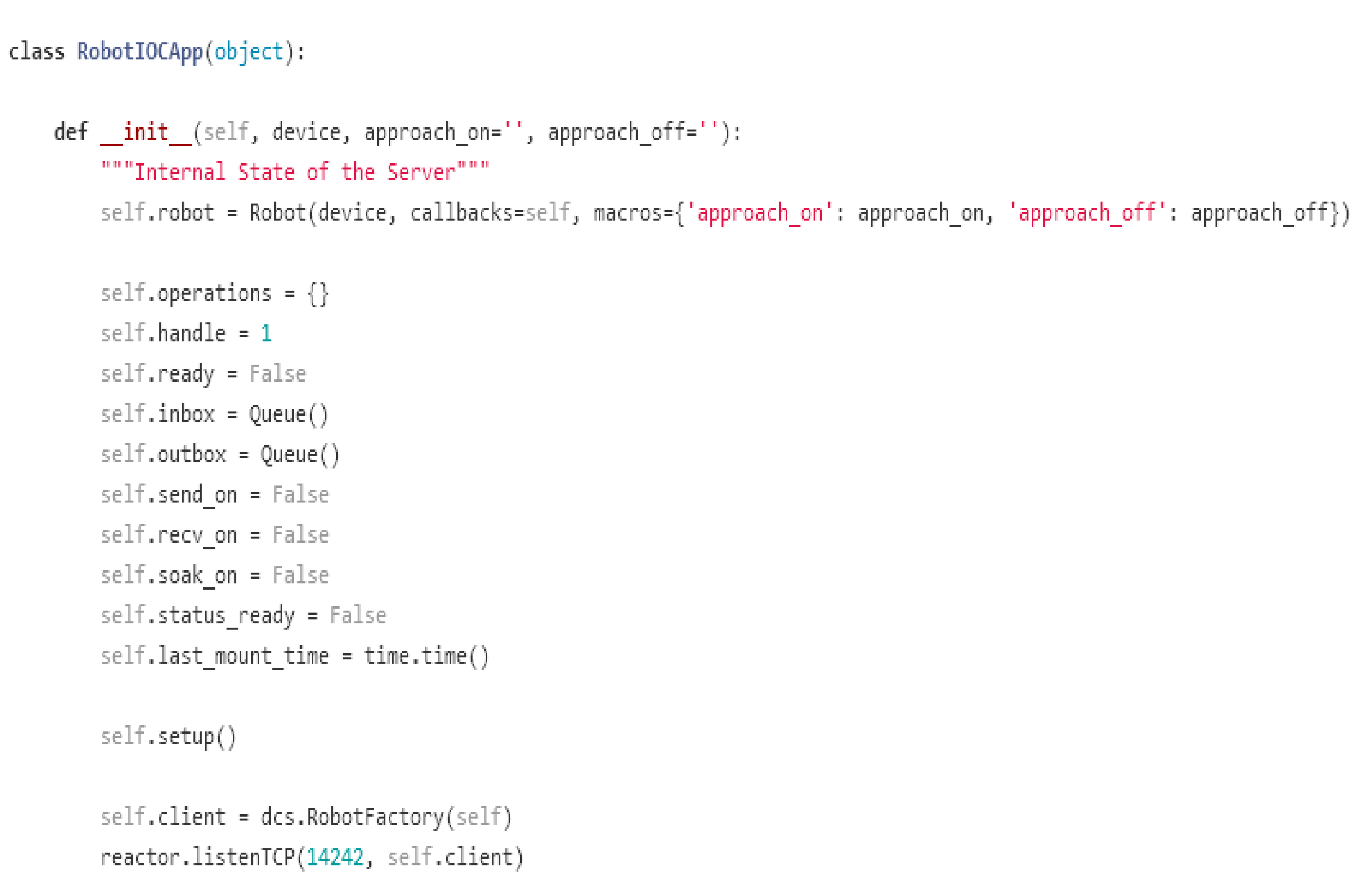
```python
          ory.receive_message(message)


class RobotFactory(protocol.ServerFactory):
    protocol = RobotProtocol

    def __init__(self, application):
        self.application = application
        self.ready = False
        self.client = None

    def buildProtocol(self, address):
        logger.log(log.IMPORTANT, 'SAM Ready: {}'.format(address))
        self.client = self.protocol(self)
        return self.client
```

unclesam/ioc.py



```
class Robot(models.Model):
    connected = models.Enum('CONNECTED', choices=('Inactive', 'Active'), default=0, desc="Connection")
    enabled = models.Enum('ENABLED', choices=('Disabled', 'Enabled'), default=1, desc="Control")
    heartbeat = models.Enum('HEARTBEAT', choices=('TICK', 'TOCK'), default=0, desc="Heartbeat")
    normal = models.Enum('HEALTH', choices=('Abnormal', 'Normal'), desc="Health")
    status = models.Enum('STATUS', choices=StatusType, desc="Status")
    log = models.String('LOG', desc="Sample Operation Message", max_length=1024)
    log_alarm = models.Enum('LOG:ALARM', choices=LogType, desc="Log Level")
    sample_log = models.String('MESSAGE', desc="Sample Log", max_length=512)
    handle = models.Integer('HANDLE', desc="Last Operation", default=1)

    # Safety flags
    prepare = models.Enum('SAFETY:PREPARE', choices=OnOffType, desc="Prepare for Approach")
    gonio_ready = models.Enum('SAFETY:READY', choices=OnOffType, desc="Ready for Approach")
    approach_on = models.Enum('SAFETY:APPROACH:ON', choices=OnOffType, out="$(approach_on) PP NMS", desc="Approaching Go
    approach_off = models.Enum('SAFETY:APPROACH:OFF', choices=OnOffType, out="$(approach_off) PP NMS", desc="Left Gonio"

    ports = models.String('PORTS', max_length=291, desc="Port States")
    ports_left = models.Array('PORTS:L', type='SHORT', length=96, desc="L Port States")
    ports_middle = models.Array('PORTS:M', type='SHORT', length=96, desc="M Port States")
    ports_right = models.Array('PORTS:R', type='SHORT', length=96, desc="R Port States")
    cassette_left = models.Enum('CASSETTE:L', choices=CassetteType, desc="L Cassette")
    cassette_middle = models.Enum('CASSETTE:M', choices=CassetteType, desc="M Cassette")
    cassette_right = models.Enum('CASSETTE:R', choices=CassetteType, desc="R Cassette")
```

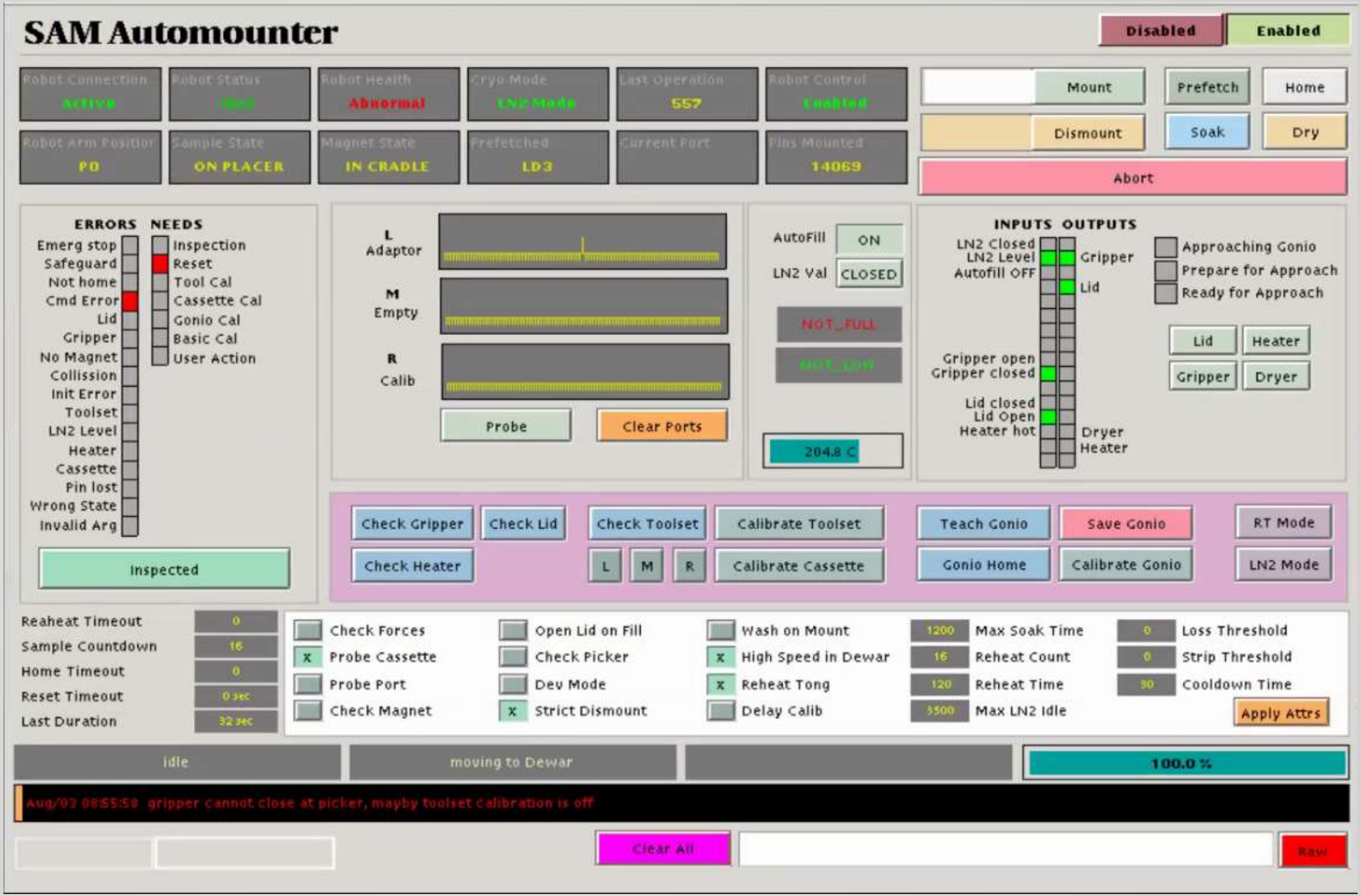
~100 Records



```
class RobotIOCApp(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

Robot Connection: Active
Robot Status: Idle
Robot Health: Abnormal
Cryo Mode: LN2 Mode
Last Operation: 557
Robot Control: Enabled
Robot Arm Position: P0
Sample State: ON PLACER
Magnet State: IN CRADLE
Prefetched: LD3
Current Port:
Pins Mounted: 14069

AutoFill: ON
LN2 Val: CLOSED
NOT_FULL
NOT_LOW
204.8 C

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

□ Check Forces
☒ Probe Cassette
□ Probe Port
□ Check Magnet
□ Open Lid on Fill
□ Check Picker
□ Dev Mode
☒ Strict Dismount
□ Wash on Mount
☒ High Speed in Dewar
☒ Reheat Tong
□ Delay Calib

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

idle | moving to Dewar | | 100.0 %

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

Thanks

?