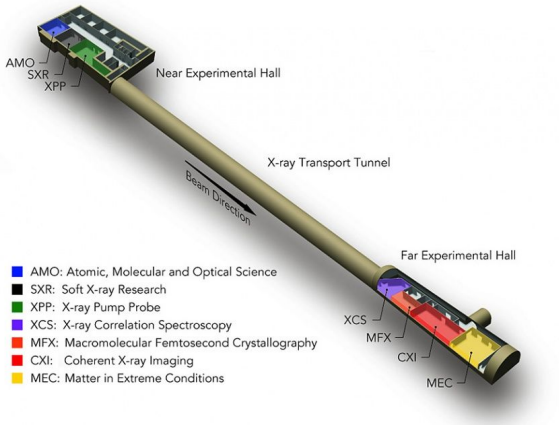
SC Timing System: BSA Plot and Diagnostic Tool

Drake Jha April 20th, 2023

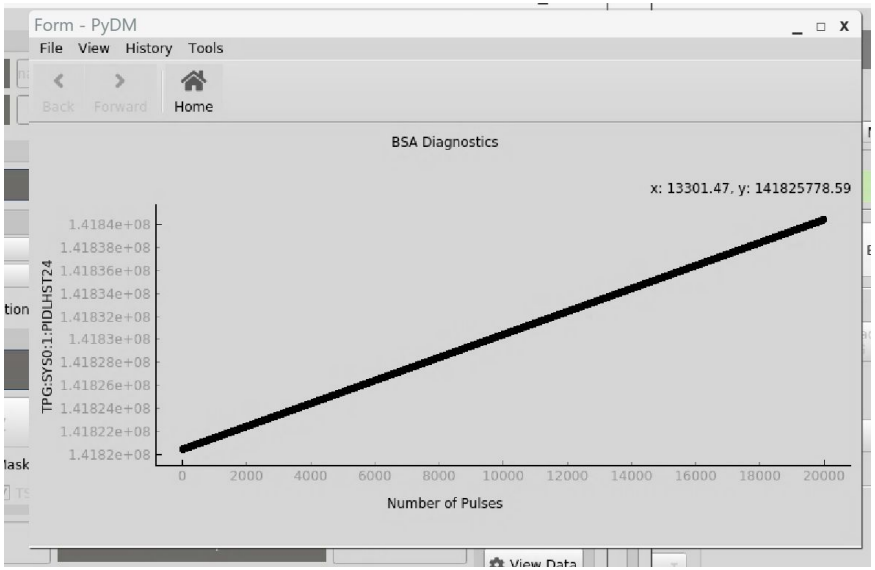


U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

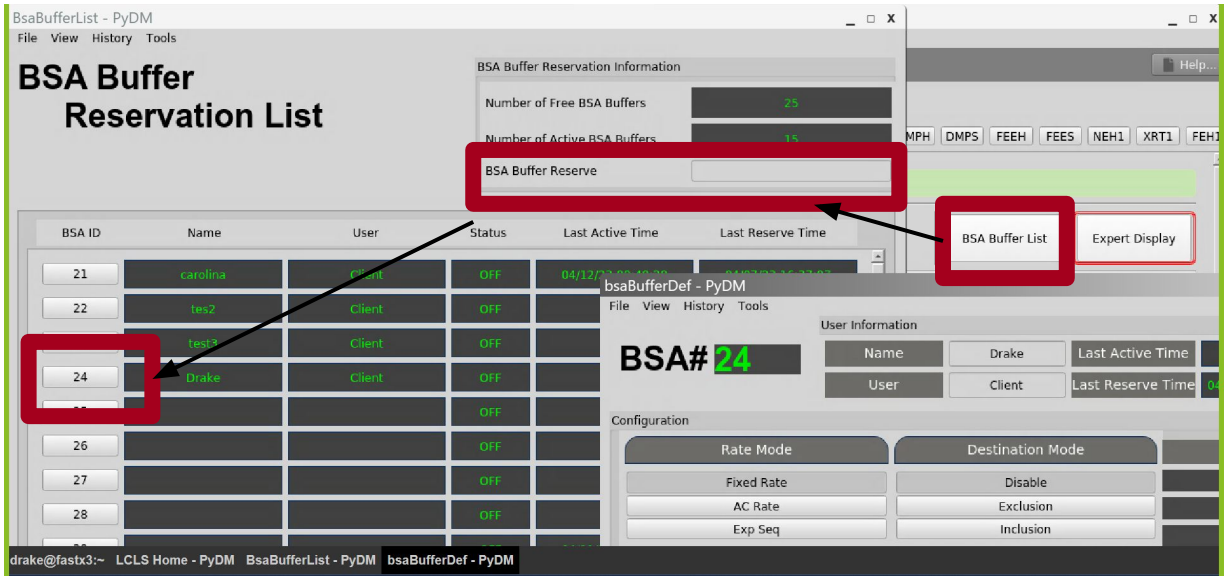
BSA Plot

- The BSA (Beam Synchronous Acquisition) reads pulse by pulse signals across many devices and publishes it in waveforms.
- lclshome & —> SC —> Event —-> BSA Buffer List —> (Click on a BSA ID number) —> View Data —> (select PV from drop-down menu) —> BSA plot will appear

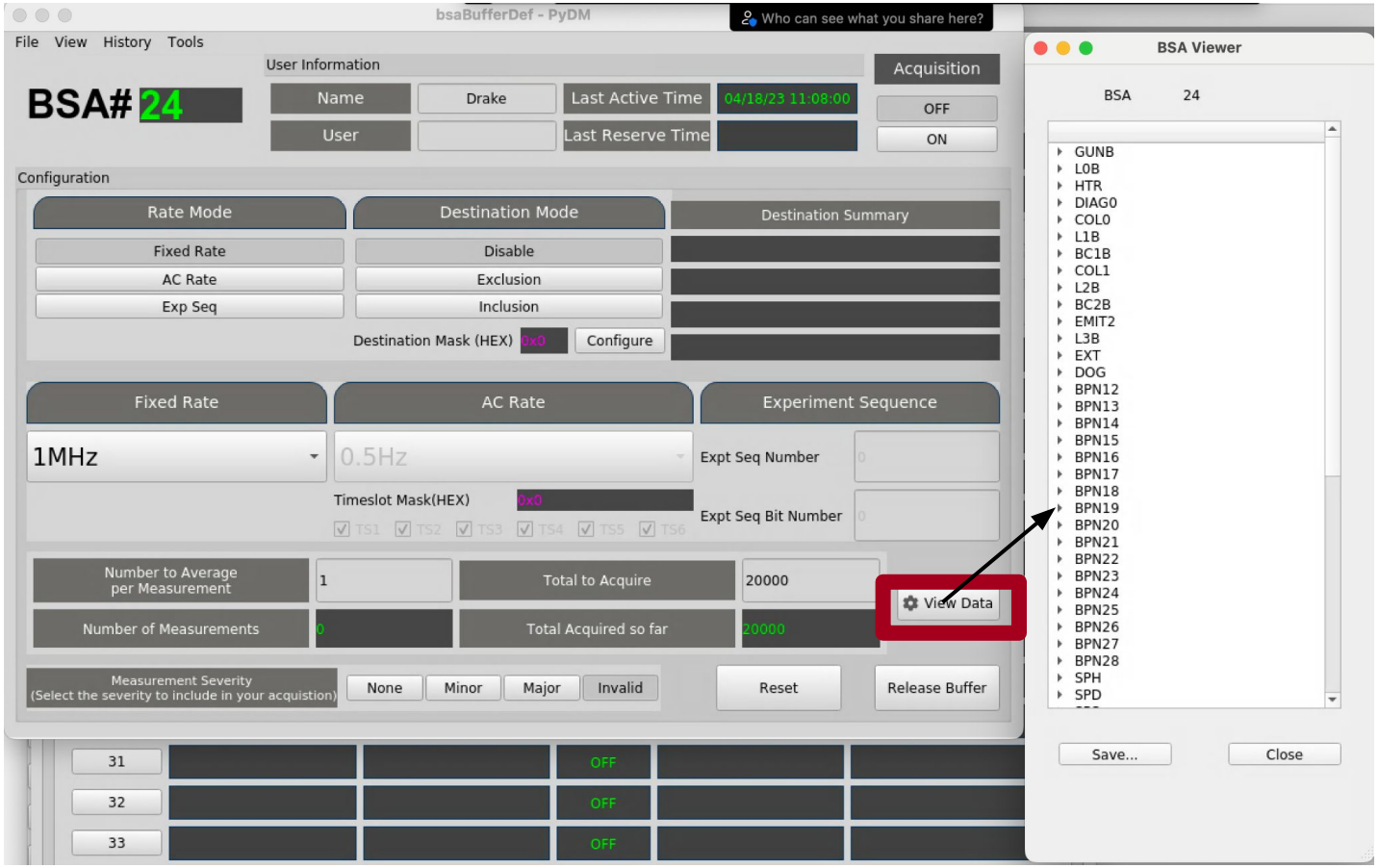


BSA Plot

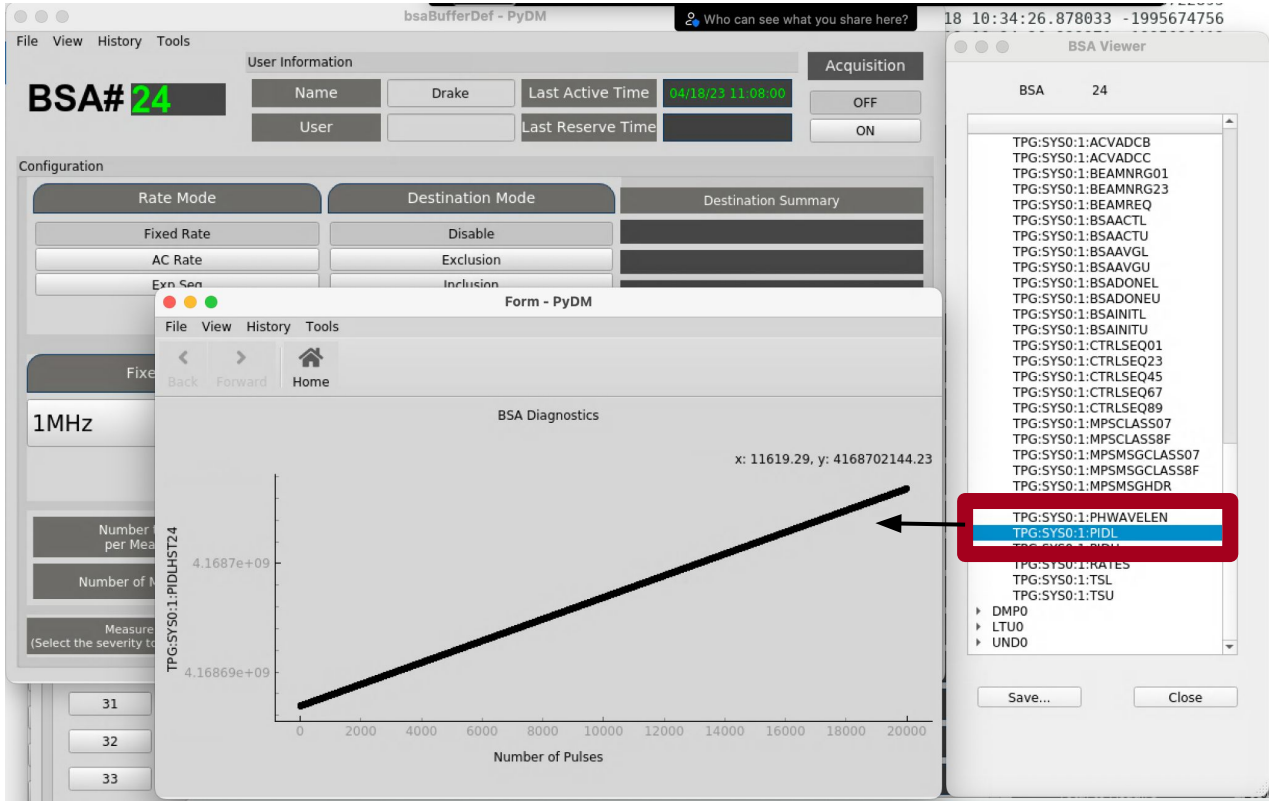
- From the BSA PyDM Interface, after collecting data, it is now possible to view a signal waveform in time.



BSA Plot

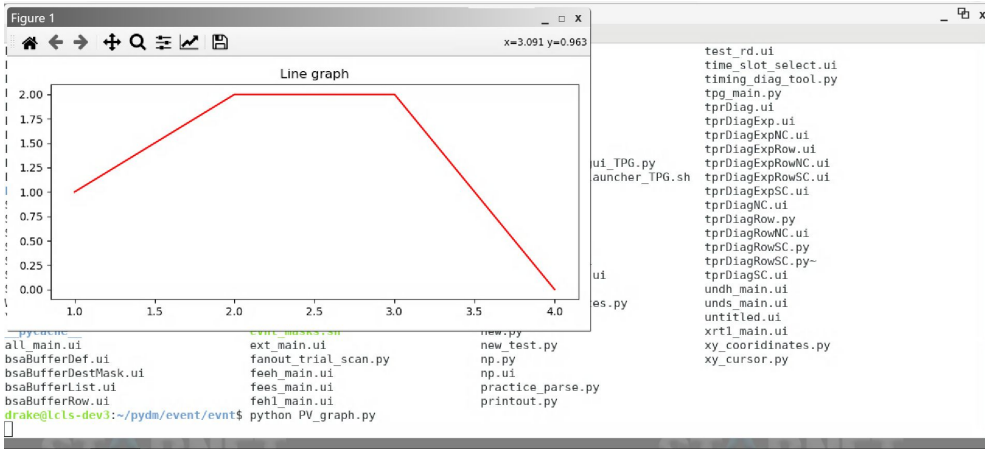

bsaBufferDef - PyDM
Who can see what you share here?
File View History Tools
User Information
BSA# 24
Name
Drake
Last Active Time
04/18/23 11:08:00
User
Last Reserve Time
Acquisition
OFF
ON
Configuration
Rate Mode
Destination Mode
Destination Summary
Fixed Rate
AC Rate
Exp Seq
Disable
Exclusion
Inclusion
Destination Mask (HEX)
0x0
Configure
Fixed Rate
AC Rate
Experiment Sequence
1MHz
0.5Hz
Expt Seq Number
Timeslot Mask(HEX)
0x0
TS1 TS2 TS3 TS4 TS5 TS6
Expt Seq Bit Number
Number to Average per Measurement
1
Total to Acquire
20000
View Data
Number of Measurements
0
Total Acquired so far
20000
Measurement Severity
(Select the severity to include in your acquistion)
None
Minor
Major
Invalid
Reset
Release Buffer
31
32
33
OFF
BSA Viewer
BSA 24
GUNB
L0B
HTR
DIAG0
COL0
L1B
BC1B
COL1
L2B
BC2B
EMIT2
L3B
EXT
DOG
BPN12
BPN13
BPN14
BPN15
BPN16
BPN17
BPN18
BPN19
BPN20
BPN21
BPN22
BPN23
BPN24
BPN25
BPN26
BPN27
BPN28
SPH
SPD
Save...
Close

BSA Plot


bsaBufferDef - PyDM
File View History Tools
BSA# 24
User Information
Name
Drake
Last Active Time
04/18/23 11:08:00
User
Last Reserve Time
Acquisition
OFF
ON
Configuration
Rate Mode
Destination Mode
Destination Summary
Fixed Rate
AC Rate
Disable
Exclusion
1MHz
31
32
33
Form - PyDM
File View History Tools
Back
Forward
Home
BSA Diagnostics
x: 11619.29, y: 4168702144.23
TPG:SYS0:1:PIDLHST24
4.1687e+09
4.16869e+09
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
20000
Number of Pulses
BSA Viewer
BSA 24
TPG:SYS0:1:ACVADCB
TPG:SYS0:1:ACVADCC
TPG:SYS0:1:BEAMNRG01
TPG:SYS0:1:BEAMNRG23
TPG:SYS0:1:BEAMREQ
TPG:SYS0:1:BSAACTL
TPG:SYS0:1:BSAACTU
TPG:SYS0:1:BSAAVGL
TPG:SYS0:1:BSAAVGU
TPG:SYS0:1:BSADONEL
TPG:SYS0:1:BSADONEU
TPG:SYS0:1:BSAINITL
TPG:SYS0:1:BSAINITU
TPG:SYS0:1:CTRLSEQ01
TPG:SYS0:1:CTRLSEQ23
TPG:SYS0:1:CTRLSEQ45
TPG:SYS0:1:CTRLSEQ67
TPG:SYS0:1:CTRLSEQ89
TPG:SYS0:1:MPSCLASS07
TPG:SYS0:1:MPSCLASS8F
TPG:SYS0:1:MPSMSGCLASS07
TPG:SYS0:1:MPSMSGCLASS8F
TPG:SYS0:1:MPSMSGHDR
TPG:SYS0:1:PHWAVELEN
TPG:SYS0:1:PIDL
TPG:SYS0:1:RATES
TPG:SYS0:1:TSL
TPG:SYS0:1:TSU
DMP0
LTU0
UND0
Save...
Close

How I constructed the program

- Started by making a PyQt graph that could plot data from two arrays in .py file:



- Then I needed to incorporate the .py and .ui files together, at the beginning I did a lot of the PyDM functionalities in the python file
- Also, I needed to get the data from the process variables (PVs) as macros

How I constructed the program

- Received different parts of the PVs as macros from a related display or the command line, then parsed and constructed them, then used ‘caget’ function to get value:

```
class PV_plot(Display, QtWidgets.QMainWindow):

    def __init__(self, parent=None, args=None, macros=None):

        super(PV_plot, self).__init__(parent=parent, args=args, macros=None)
        pv_parts = ''
        counter = len(macros)

        for key in macros:

                if not key == 'BSA_ID':

                        pv_parts += (macros[key])

                        if counter > 1:

                                pv_parts += ':'

                counter -= 1

        y_channel = pv_parts + 'HST' + macros['BSA_ID']
        x_channel = 'BSA:SYS0:1:' + macros['BSA_ID'] + ':CNT'
        pv_lst = [y_channel, x_channel]
        pv_values_lst = epics.caget_many(pv_lst, timeout=0.1)
        y_values = pv_values_lst[0]
        x_values = [1]
-- INSERT --
```

How I constructed the program

- Tapped into attributes of PyDMWaveformPlot to do some of the functionalities manually :

```
self.curve.redrawCurve()

self.PyDMWaveformPlot.addAxis(self.curve, 'Axis 1', 'left', y_channel)
self.PyDMWaveformPlot.addChannel(redraw_mode=1, yAxisName='Axis 1')
self.PyDMWaveformPlot.addCurve(self.curve, y_axis_name = 'Axis 1')
self.PyDMLabel_4.setText(x_channel)
```

- …

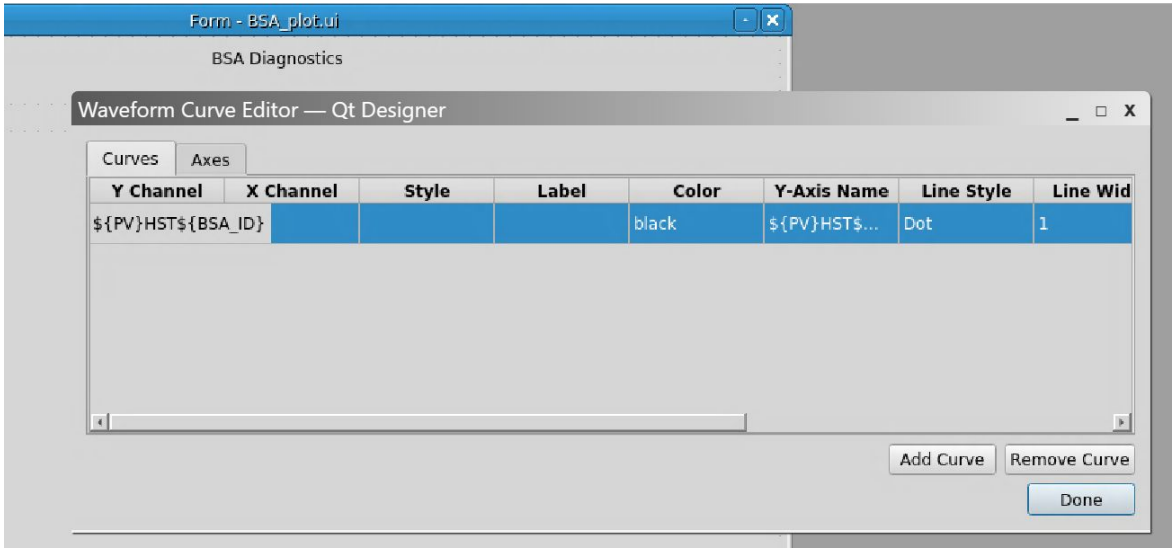
```
self.curve = widgets.waveformplot.WaveformCurveItem(color=QColor(0, 0, 0))
arr_1 = numpy.array(x_values)
self.curve.receiveXWaveform(arr_1)
```

- Then transitioned to receiving data and plotting Waveform in Designer

```
def ui_filename(self):

    return 'BSA_plot.ui'
```

How I constructed the program

- For most of the project I was trying to use Scatterplot, but at the end, I switched to using the Waveform Plot because the Scatterplot could only plot scalar values
- Used PV channel address in PyDm:



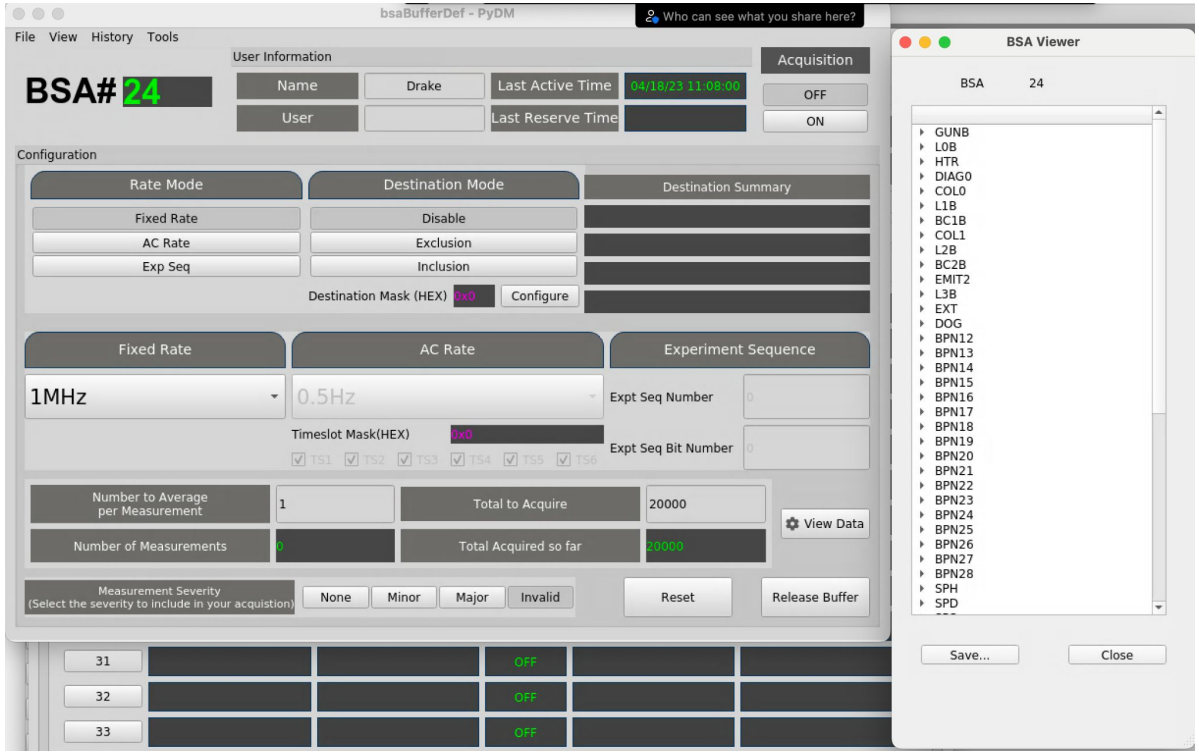
How I constructed the program

- Another big part of this project was making the mouse coordinate feature on the plot:


```
    proxy = pg.SignalProxy(self.PyDMWaveformPlot.scene().sigMouseMoved, rateLimit=60, slot=self.mouseMoved)
    proxy.signal.connect(self.mouseMoved)

def ui_filename(self):


def ui_filepath(self):

        return path.join(path.dirname(path.realpath(__file__)), self.ui_filename())

def mouseMoved(self, evnt):

        mousePoint = self.PyDMWaveformPlot._curves[0].mapFromScene(evnt)
        self.PyDMLabel.setText('x' + ': ' + str(round(mousePoint.x(), 2)) + ', y: ' + str(round(mousePoint.y(), 2)))
-- INSERT --
```

File Network

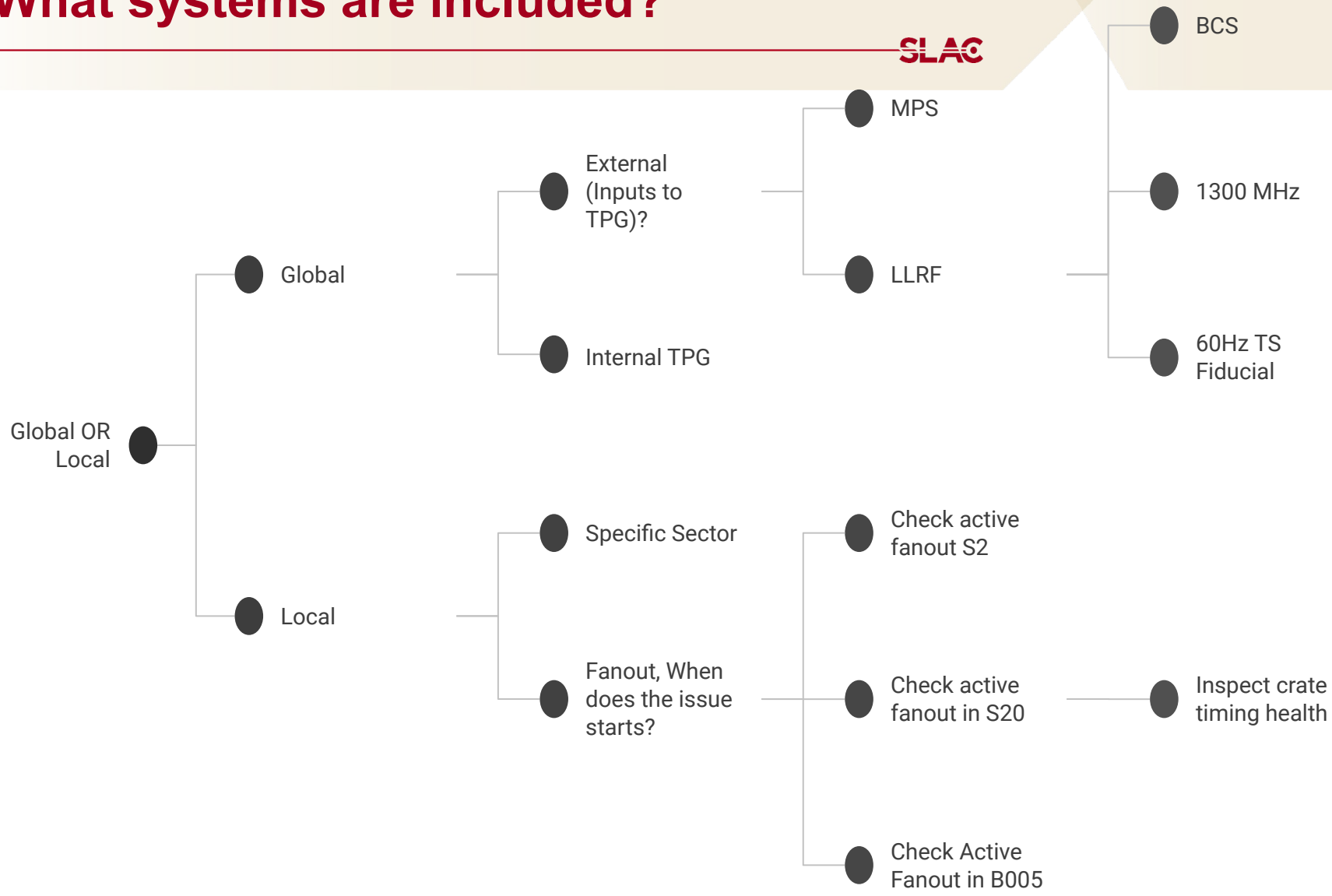
- A lot of testing was done to verify the connection between displays. The ‘View Data’ button calls the directory service through python code and makes a list of BSA PVs in a Python-QT widget list.



SC Diagnostic Tool

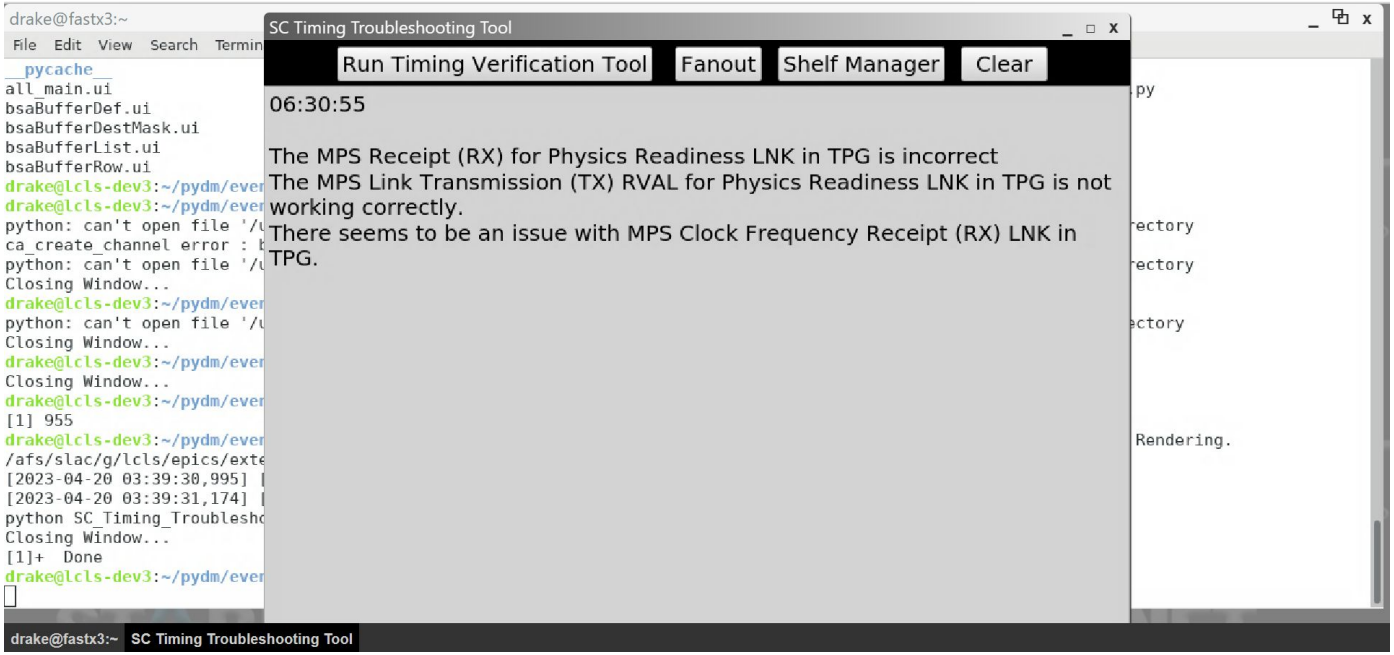
- Tool developed to give a diagnostic report back regarding instabilities in the timing system, starting panel:

What systems are included?



SC Timing Verification Tool: Example:

- Checks the input and outputs to the SC Timing System. The program then delivers a message indicating if there is an issue and provides a clue to fix the issue.



- Coming Soon… UI to query timing signal health on shelf managers and TPRs

How I Constructed the Program

- The program is four .py files:
  - SCTimingVerificationTool.py
  - SC_Shelf_Manager_PV_Test_Panel.py
  - SC_Fanout_PV_Test_Panel.py
  - SC_Timing_Troubleshooting_Tool.py
- First underwent checks in the TPG system using ‘caget’ to get the values of the PVs

```
        TPG_pv_lst = ['TPG:SYS0:1:COUNTBRT', 'TPG:SYS0:1:MPSLNK:PHYREADYRX', 'TPG:SYS0:1:MPSLNK:PHYREADYRX.RVAL', 'TPG:SYS0:1:MPSLNK:PHYREAD
YTX', 'TPG:SYS0:1:MPSLNK:PHYREADYTX.RVAL', 'TPG:SYS0:1:RATETXCLK', 'TPG:SYS0:1:MPSLNK:RXCLOCKFREQ', 'TPG:SYS0:1:MPSLNK:TXCLOCKFREQ']

        caget_many_values_TPG = epics.caget_many(TPG_pv_lst, timeout = 0.1)

        if not caget_many_values_TPG[0] == 910000:
                print('The TPG might not be working properly as the Frequency readback is not = 910000, please contact: C. B. Mattison')

        if not caget_many_values_TPG[1] == 'Ready':
                print('The MPS Receipt (RX) for Physics Readiness LNK in TPG is incorrect')

        if not caget_many_values_TPG[2] == 1.0:
                print('The MPS Link Receipt (RX) RVAL for Physics Readiness LNK in TPG is not working correctly.')

        if not caget_many_values_TPG[3] == 'Ready':
                print('The MPS Link Transmission (TX) RVAL for Physics Readiness LNK in TPG is not working correctly.')

        if not caget_many_values_TPG[4] == 1.0:
                print('The MPS Transmission (TX) RVAL for Physics Readiness LNK in TPG is incorrect')

        if not 184.0 <= caget_many_values_TPG[5] <= 190.0:
                print( 'The Transmission (TX) RATE Clock in TPG is out of range')

        if not caget_many_values_TPG[6] == 250000100:
                print( 'There seems to be an issue with MPS Clock Frequency Receipt (RX) LNK in TPG.')
```

How I Constructed the Program

- Used PyQT to construct the UI's of the panels:

```
def __init__(self):

        QMainWindow.__init__(self)
        QDesktopWidget.__init__(self)

        x = round(self.width()*1.25)
        y = round(self.height()*1.25)
        self.setMinimumSize(QSize(x, y))
        # self.setWindowTitle("SC Fanout PV Tool")
        self.setStyleSheet("background-color: black")

        self.editorOutput = QPlainTextEdit(self)
        self.editorOutput.resize(x, int(y*0.93))
        self.editorOutput.move(0, int(y*0.07))

        self.button_run = QtWidgets.QPushButton(self, clicked=self.runCommand)
        self.button_run.setText("Run Fanout")
        self.button_run.move(int(x*0.5591), int(y*0.01))
        self.button_run.adjustSize()
        self.button_run.setStyleSheet("background-color : lightgrey")

        self.button_tpg_script = QtWidgets.QPushButton(self, clicked=self.run_TPG_script_command)
        self.button_tpg_script.setText("Scan")
        self.button_tpg_script.resize(int(x*0.145), self.button_run.height())
        self.button_tpg_script.move(int(x*0.76005), int(y*0.01))
        # self.button_tpg_script.adjustSize()
```

```
def __init__(self):

        QMainWindow.__init__(self)
        QDesktopWidget.__init__(self)

        x = round(self.width()*1.25)
        y = round(self.height()*1.25)
        self.setMinimumSize(QSize(x, y))

        self.setWindowTitle("SC Shelf Manager PV Value Identification Tool")
        self.setStyleSheet("background-color: black;")

        self.editorOutput = QPlainTextEdit(self)
        self.editorOutput.resize(x, int(y*0.93))
        self.editorOutput.move(0, int(y*0.07))

        self.button_run = QtWidgets.QPushButton(self, clicked=self.runCommand)
        self.button_run.setText("Run Shelf Manager")
        self.button_run.move(int(x*0.56), int(y*0.01))
        self.button_run.adjustSize()
        self.button_run.resize(int(x*0.25), self.button_run.height())
        self.button_run.setStyleSheet("background-color : lightgrey")

        self.PV_input_box = QPlainTextEdit(self)
        self.PV_input_box.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
        self.PV_input_box.resize(int(x*0.553), self.button_run.height())
        self.PV_input_box.move(int(x*0.005), int(y*0.01))
```

How I Constructed the Program

- Updating the command line environment was a big challenge for me, but it was needed so the IPMI commands could be read from shelf manager

```
def update_env2(self, script_path):

        if not '--child' in sys.argv:

                os.execl('/bin/bash', '/bin/bash', '-c', 'source %s; %s %s --child' % (script_path, quote(sys.executable), quote(sys
.argv[0])))
```

- Also, I had to parse the message from shelf manager PV. Instead of using the Python function, I made my own recursive method:

```
        def parse(self, msg):

                msg = msg.split('\n')
                msg.pop()

                parsed_segment = ''

                for string in msg:

                        if not string[-2] == 'D':

                                for i in range(len(string)-1, -1, -1):

                                        if len(string) - i < 0:

                                                break

                                        if string[i] == '!':

                                                counter = i
                                                original_i = i

                                                while not string[counter:counter + 4] == 'sioc':

                                                        counter -= 1

                                                del_str = ''
-- INSERT --
```

What I Learned From These Projects

- Python: EPICS (Channel Access Control System Module), PyQt (Python Graphical User Interface Module)
- GIT and CVS (version control systems)
- User interface (UI) experience
- Controls software and timing
- Accelerator complex hardware operations
- Data structures
- Macros
- PyDM
- Programming and problem solving concepts
- Communication and collaboration skills

Thank You/Acknowledgments

Special Thank You to Carolina Bianchini Mattison, Matt Gibbs and Ryan McClanahan