An Introduction to Using IGOR in the Quantum Devices Group

Tim Child
Johann Drayne
13th December 2021

Table of Contents

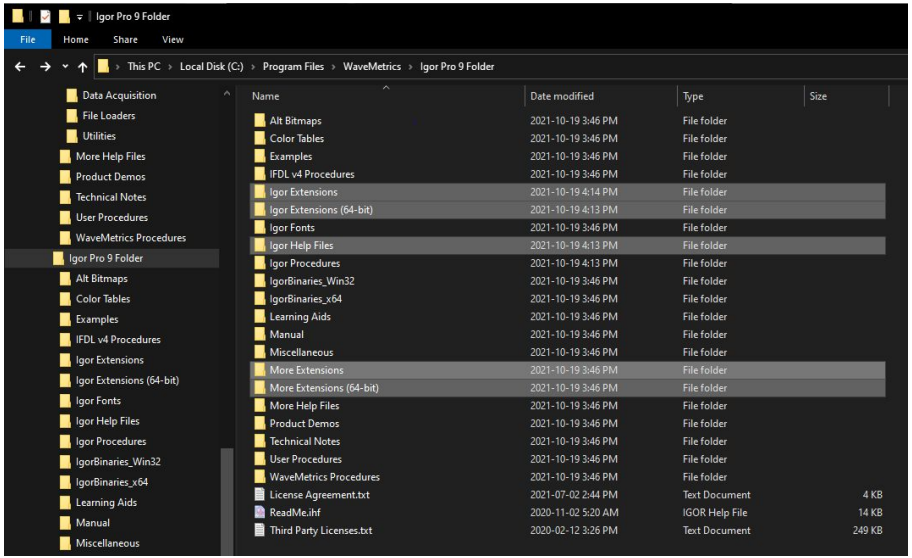
XOPs

On GitHub, go to ‘IgorAcq/XOP/’

Here you will see a list of XOP’s

- JSON-XOP-latest
- SOCKIT
- SQL
- VISA-ASYNC_7

Also need to copy into IGOR procedures.



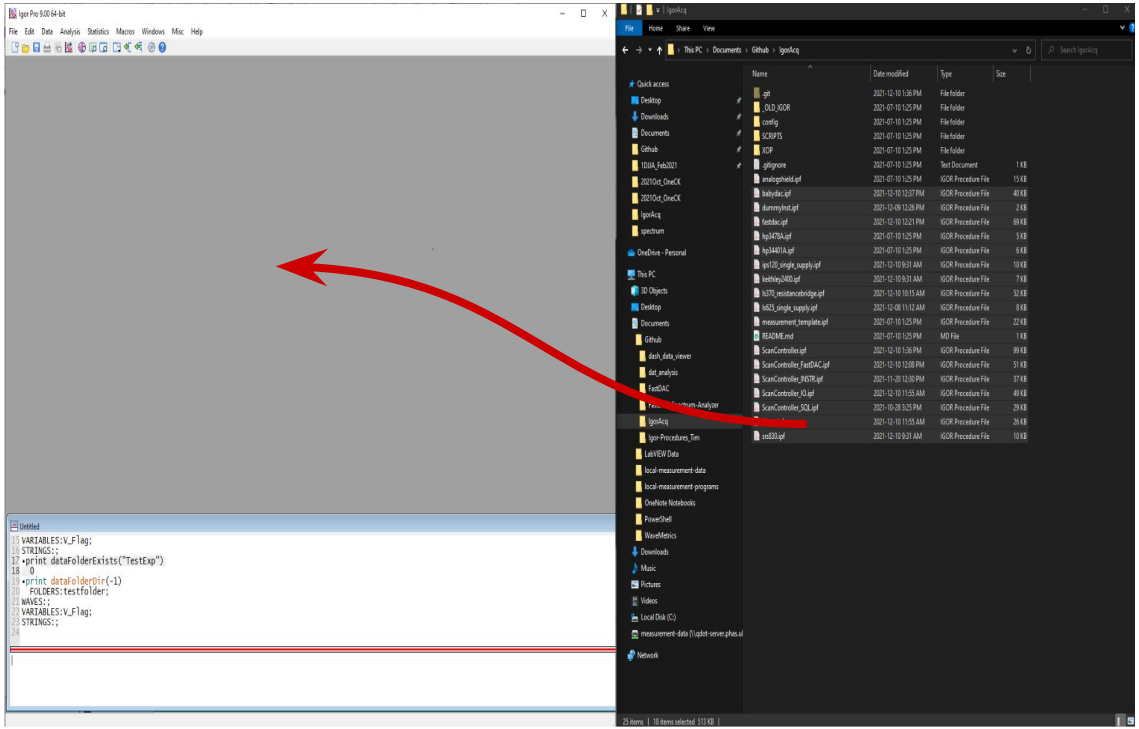
Adding .ipf files

Open up IGOR and go File/New Experiment

From C:\Users\folklab\Documents\Github\IgorAcq

Add the following .ipf files by drag and drop

- babydac.ipf
- fastdac.ipf
- ls370_resistancebridge.ipf
- ls625_single_supply.ipf
- ScanController.ipf
- ScanController_FastDAC.ipf
- ScanController_INSTR.ipf
- ScanController_IO.ipf
- ScanController_SQL.ipf
- Scans.ipf
- srs830.ipf
- keithley2400.ipf



Or manually add in IGOR: File/Open File/Procedure…

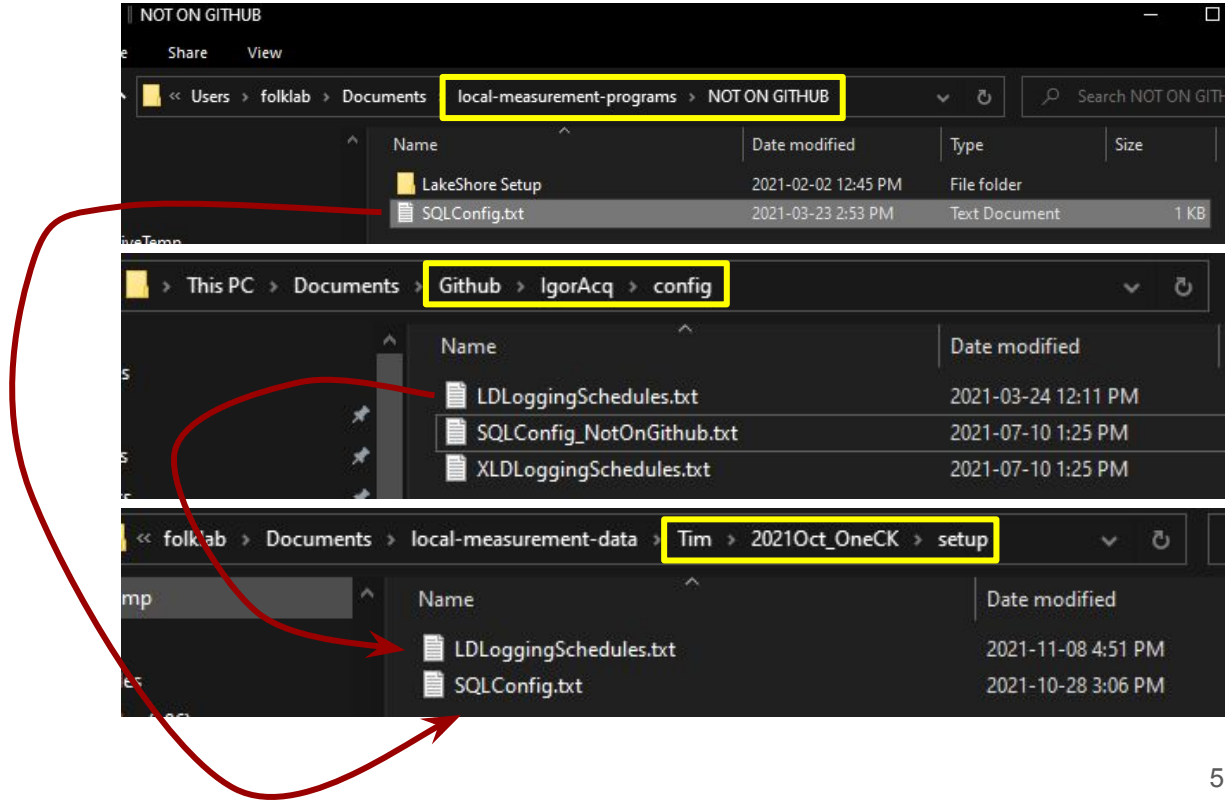
Adding .txt files

We also need to add some text files to a ‘setup’ folder in the experiment.

Two files to move:

- C:\Users\folklab\Documents\local-measurement-programs\SQLConfig.txt
- C:\Users\folklab\Documents\Github\IgorAcq/config/LDLoggingSchedules.txt

Add to experiment subfolder names ‘setup’



Defining File Paths

Local file path

C:\Users\folklab\Documents\local-measurement-data
\<your name>\<experiment>

Where:

<your name> is your name

<experiment> is the specific experiment name

Set paths in IGOR

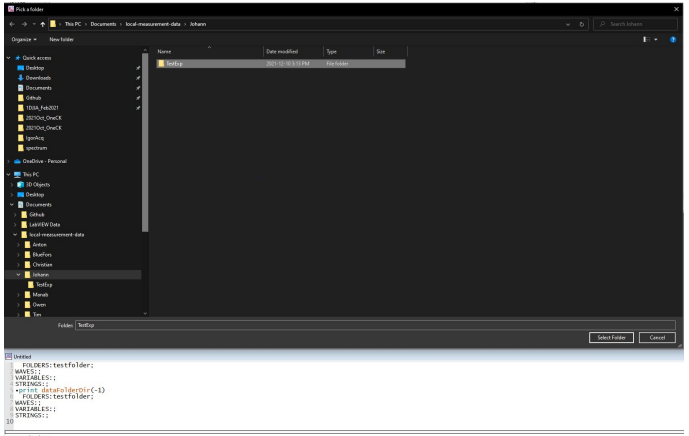
newpath data (Then select Local file path)

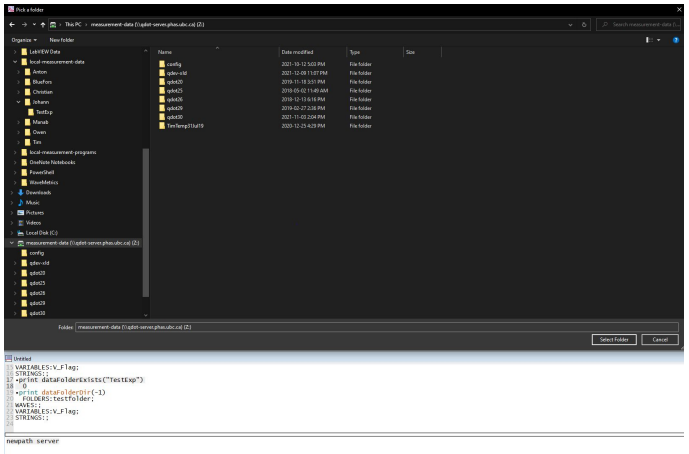
newpath server (Then select server drive e.g. Z:)

Check if folder exists

print dataFolderExists(data)

prints 1 if path exists





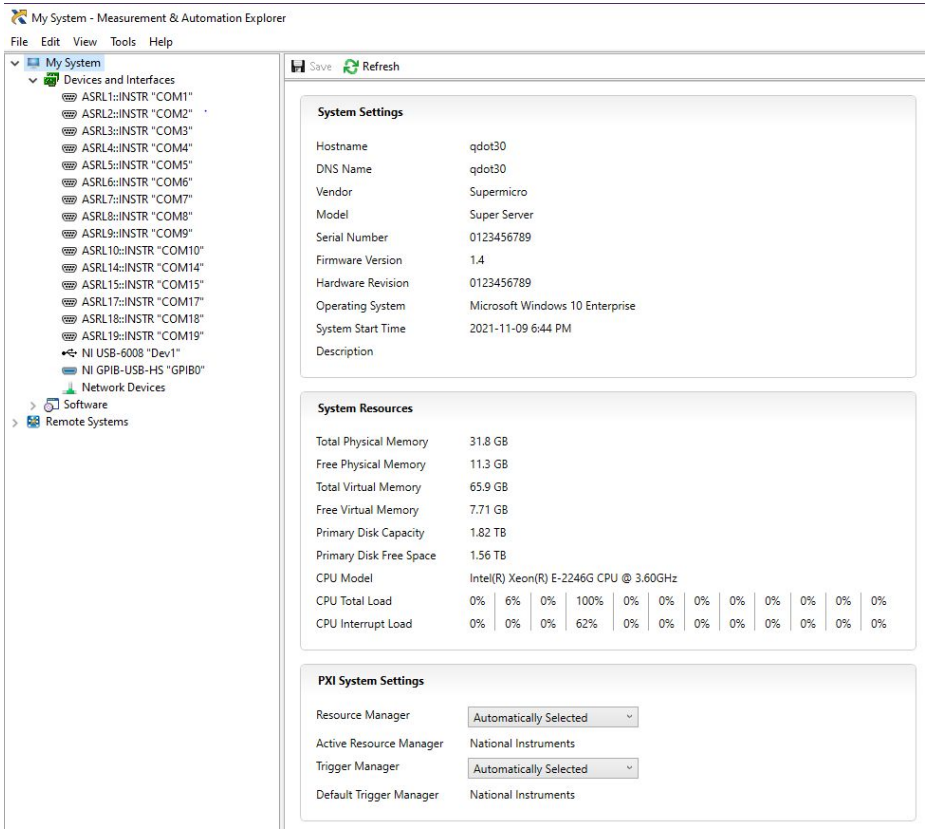
ASRL number

Open application NI MAX

Connect instrument and use ‘F5’ to refresh (note “Refresh” does not refresh the device list)

Or

listSerialports()
listGPIBinstr()



Instrument specific connections

Useful to set verbose = 1

FastDAC:
openFastDACconnection(instrID, visa_address, [verbose,numDACCh,numADCCh,master, optical])
E.g. openFastDACconnection("fd", "ASRL18::INSTR", numDACCh=8, numADCCh=4, master=1, verbose=1)

BabyDAC: Detailed BabyDAC information already on Discourse https://qdev-forum.phas.ubc.ca/t/babydac-wiki/102
openBabyDACconnection(instrID, visa_address, [verbose])
E.g. openBabyDACconnection("bd", "ASRL10::INSTR", verbose=1)

InitBabyDACs(instrID, boards, ranges, [custom])
E.g. InitBabyDACs(bd, "5,4", "55,55")

Magnet: (magnet values can be found in the comments in openLS625connection() for LD nad XLD)
openLS625connection(instrVarName, visa_address, amps_per_tesla, max_field, max_ramprate, [verbose, hold])
E.g. openLS625connection("magy", "ASRL15::INSTR", 55.2181, 1000, 155.058, verbose=1)

Remotely Controlling Fridge:
openLS370connection(instrID, http_address, system, [verbose])
E.g. openLS370connection("ls370", "10.18.101.12:49301/api/v1/", "bfsmall", verbose=1)

Checking Instrument Connections

FastDAC:
print getFDACStatus(instrID)
rampMultipleFdac(instrID,channels,setpoint)
print getFDACStatus(instrID

BabyDAC: https://qdev-forum.phas.ubc.ca/t/babydac-wiki/102

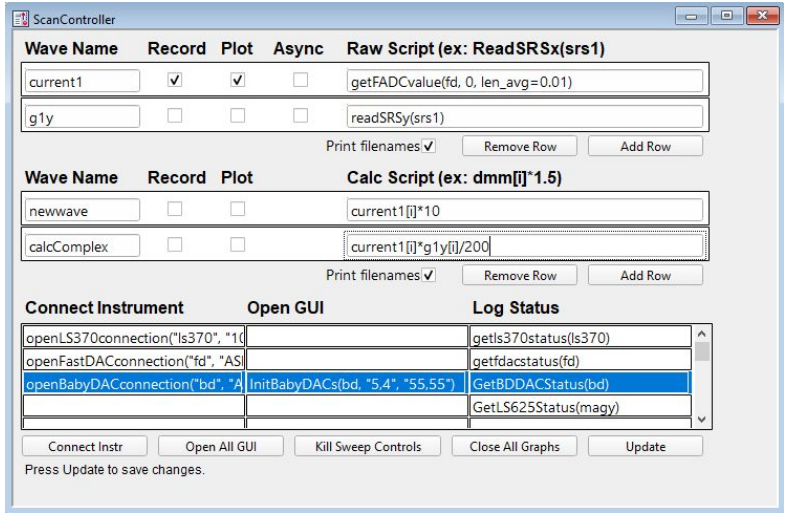
Magnet:
print getIs625status(magy)
setLS625rate(magy,50)
print getIs625status(magy)

Status Output: We will address how to make use of this logging in post analysis.
{"LS625 Magnet Supply":{"variable name":"magy", "field mT":-0.03079, "rate mT/min":49.984}}

Types of ScanControllers

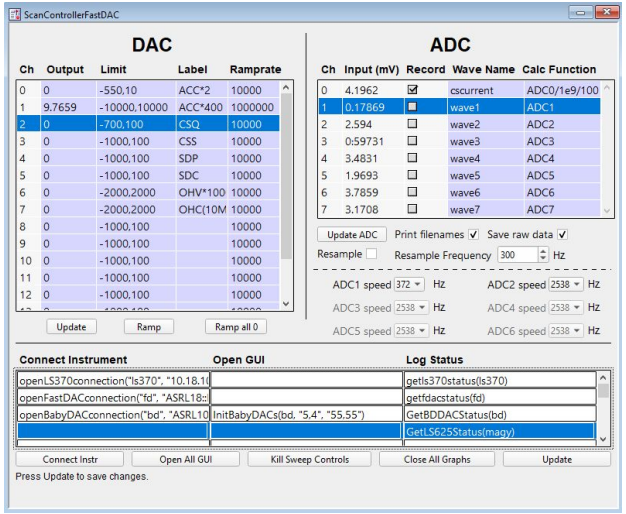
ScanController

- Original controller for Slow scans



ScanControllerFastDAC

- Equivalent of ScanController for FastDAC scans



ScanController

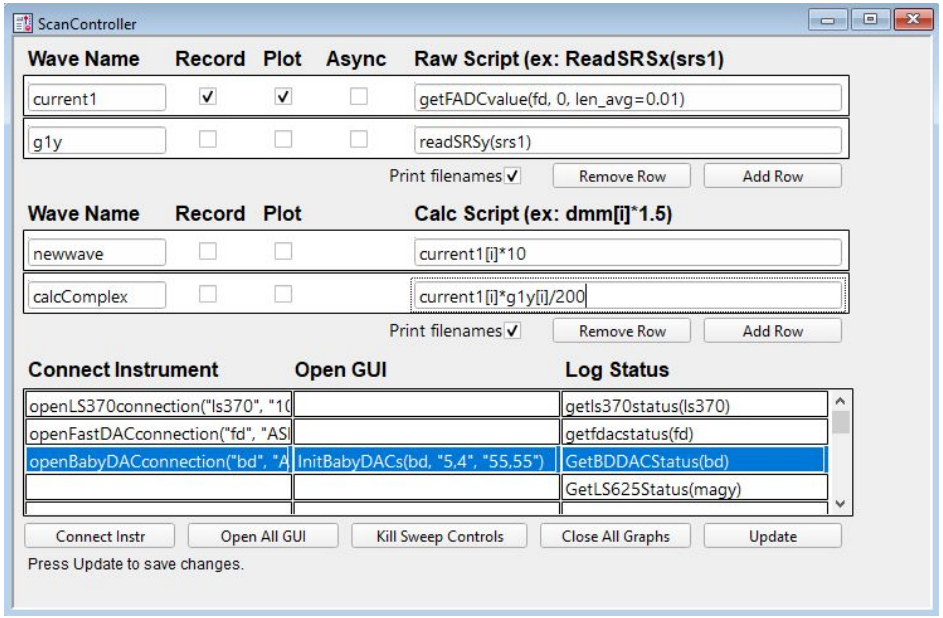
Open up the window: InitScanController()

Raw Script
FastDAC: getFADCvalue(fdid, channel, [len_avg])
BabyDAC: ReadBDadc(instrID, channel, board_number)
Lockin: readSRS(srs1)
etc

Calc Script
This will perform the mathematical operation specified.
"[i]" is used to update the corresponding index of calc wave

Connect Instrument
Copy + Paste the function for connecting to an instrument
Change to verbose = 0

Log Status (makes SweepLogs)
Logging functions collect meta data about each scan.
FastDAC: getFDACStatus(instrID)
BabyDAC: GetBDDACStatus(instrID)
370: getLS370Status(instrID)
625: GetLS625Status(instrID)



ScanControllerFastDAC - DAC

Open up the window: InitScanController()

Ch
This corresponds to the number on the FastDAC DAC labels

Output
The voltage output in mV on this FastDAC channel

Limit
The voltage limits on DAC channel.
Scan will abort if FastDAC tries to ramp outside these limits.

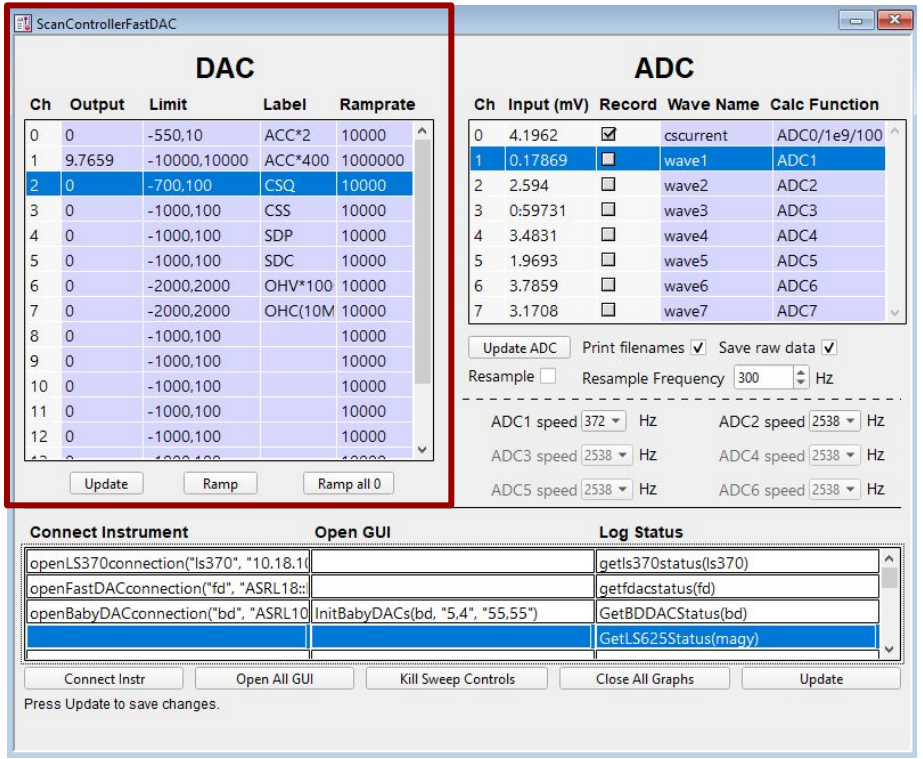
Label
When running FastDAC scans, you can refer to the channel by its number or by the label you define. This is useful if you have many different gates.

Ramprate
Sets Ramprate in mV/s for each channel (for everything except scanning)

Update
Clicking Update will ask each of the channels what they are at and will update the output.

Ramp
This will ramp each of the channels to the values specified in Output

Ramp all 0
This ramps every channel down to zero



ScanControllerFastDAC - ADC

Input
The voltage INPUT in mV on this FastDAC channel

Record
Checking record will record the inputs and save them under the variable Wave Name

Wave Name
The name of variable to refer to the specific input values

Calc Function
This will perform the mathematical operation specified.
ADC0/1000 this function:

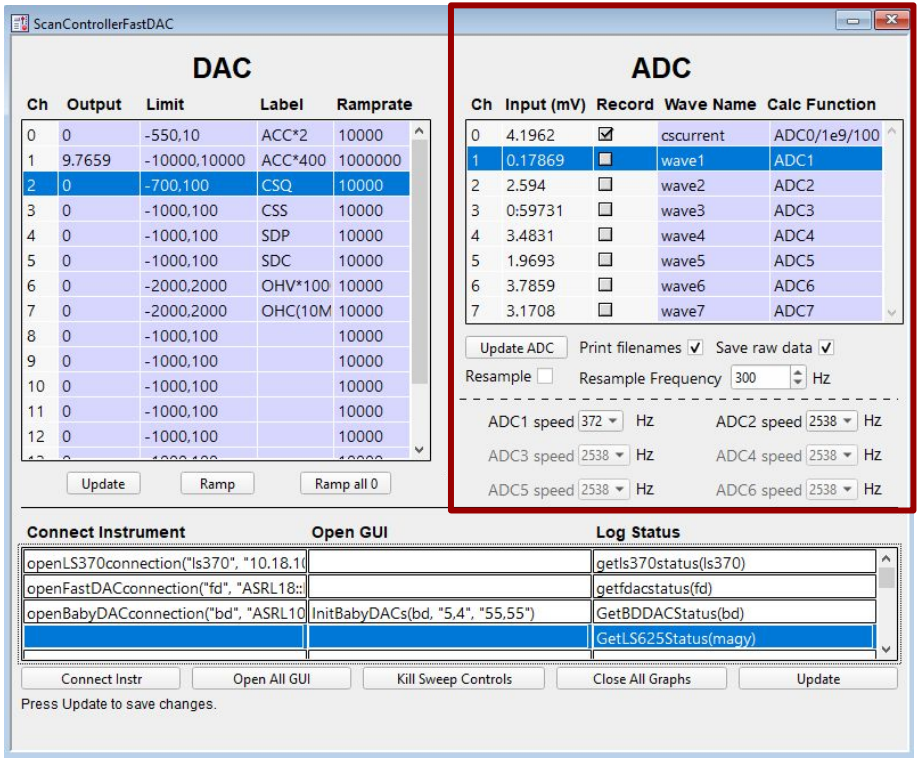
- Takes voltage readings from raw wave ADC0
- Convert mV to V by dividing by 1000
- Current amplifier is set at 1e9 amplification so V => nA in this case (but may need to do further calculation to convert to nA)

Update ADC
Clicking Update ADC will read the current value from the ADC channels

Save Raw Data
Checking Save Raw Data will save the raw data. We recommend to keep this checked (HD space is a negligible cost).

Resample (Resample Frequency)
Checking Resample will resample the raw data to the requested frequency.
I.e. Lowpass filters then downsamples to reduce datapoints significantly while maintaining info below resample frequency.

ADC Speed
This is the frequency that ScanControllerFastDAC will read values from the FastDAC ADC

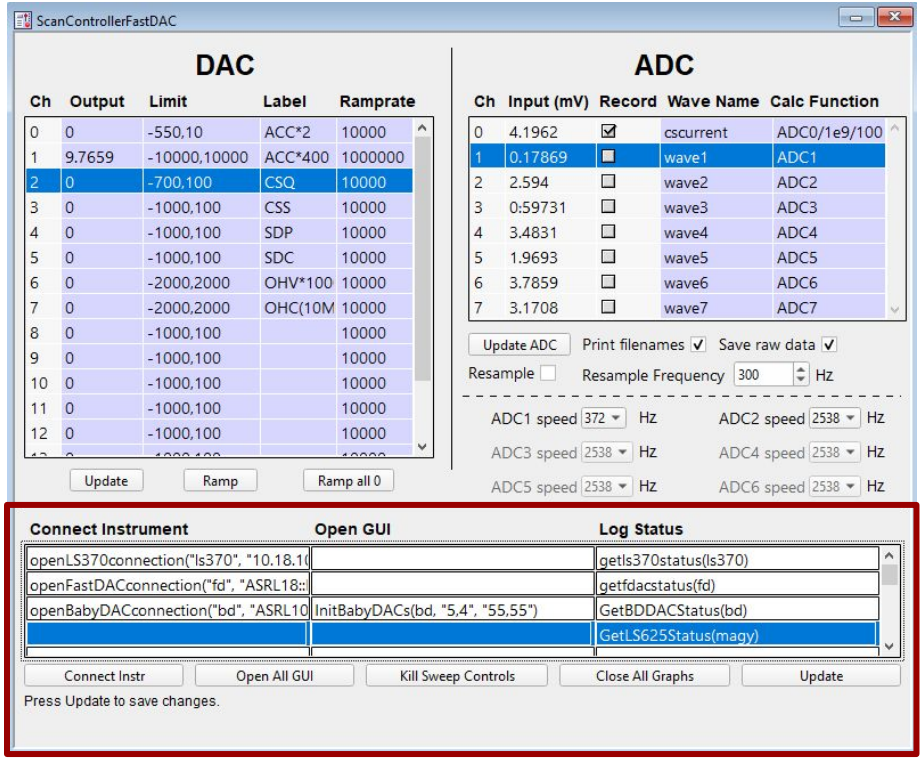


ScanControllerFastDAC - Connecting and Logging

Connect Instrument
Copy + Paste the function for connecting to an instrument
Change to verbose = 0

Open GUI
This will bring up the GUI interface with an instrument if it has been written.
Mainly used for the BabyDACs
InitBabyDACs(bd, "5,4", "55,55")

Log Status
Logging functions collect meta data about each scan.
FastDAC: getFDACStatus(fd)
BabyDAC: GetBDDACStatus(bd)
LS370: getLS370Status(ls370) ← Fridge Temps
LS625: GetLS625Status(magy) ← Mag Field



Calibrating the FastDAC

1. CalibrateFDAC(instrID) e.g. CalibrateFDAC(fd)
2. Using a BNC, connect:
   ADC 0 to DAC 0
   ADC 1 to DAC 1
   ADC 2 to DAC 2
   ADC 3 to DAC 3
3. CalibrateFADC(instrID) e.g. CalibrateFADC(fd)

BabyDAC Basic Scan (found in ‘scans.ipf’)

ScanBabyDAC(instrID, start, fin, channels, numpts, delay, ramprate, [y_label, comments, nosave])

instrID: Name of the instrument when you open a connection
start: Starting voltage in mV
fin: Finishing voltage in mV
channels: Channels to sweep with these setting.
The channel can be specified by providing the channel number or label set in ScanController BabyDAC Window E.g. "0" or "ACC*2"
numpts: Number of points in scan. This is historic from the BabyDAC scans.
delay: This is the delay between finishing the ramp to the starting voltage and taking the first data point. This can normally be set to be quite short e.g. 1s
ramprate: This is the rate to ramp to the starting voltage. This can normally be set quite high e.g. 10000

Optional
y_label: The y label on the plotted graph.
comments: These comments are saved in the hdf file and can be usedful during analysis
nosave: The output will be saved as default under a new dat. However specifying nosave=1 will not save the scan

EXAMPLES
ScanBabyDAC(bd, -10, 10, "OCSB*1000", 10, 0.001, 1000, y_label="current", comments="testing", nosave=1)

FastDAC Basic Scan (found in 'scans.ipf')

ScanFastDAC(instrID, start, fin, channels, [numpts, sweeprate, ramprate, delay, starts, fins, x_label, y_label, comments, use_AWG, nosave])

sweeprate: This is the rate at which the FastDAC will sweep from the starting voltage to the finishing voltage. In units of mV/s
The time taken for a scan can be calculated by: Time of Scan = (fin - start)/sweeprate
delay: This is the delay between finishing the ramp to the starting voltage and taking the first data point. This can normally be set to be quite short e.g. 1s
starts: This is used if you want to ramp multiple channels at the same time.
This can be done with the syntax "<channel1-start-mV>, <channel2-start-mV>" where <channel1-start-mV> is the starting voltage for channel 1.

fins: Similar to starts
Fins is the ending voltage of multiple channels "<channel1-end-mV>, <channel2-end-mV>" where <channel1-end-mV> is the ending voltage for channel 1.
x_label: The x label on the plotted graph should be set to the channel that is being ramped but this argument provides control if necessary
y_label: The y label on the plotted graph.

FastDAC Basic Scan (found in 'scans.ipf')

Sweeping One Channel

```
ScanFastDAC(fd, -10, 10, "ACC*400", sweeprate=20, ramprate=10000, delay=0, comments="testing",  nosave=1)
```

We will scan FastDAC labelled as fd

- On channel "ACC*400"
- From -10mV to 10mV
- A sweeprate of 20 in this instance will be a 1 second scan. (10 - (-10))/20 = 1
- Ramprate specifies how fast to ramp to starting voltage -10mV
- Once we reach -10mV the FastDAC will not delay and it will start sweeping immediately

Sweeping Multiple Channels

```
ScanFastDAC(fd, -100, 100, "OHV*1000, OHC(10M)", sweeprate=10, starts="-100, 70", fins="100, -70", comments="Checking ratio for 0 bias", nosave=1)
```

We will scan FastDAC labelled as fd

- We will be ramping channels "OHV*1000, OHC(10M)"
- "OHV*1000" from -100mV to 100mV
- "OHC(10M)" from 70mV to -70mV
- A sweeprate of 10 in this instance will be a 20 second scan. (100 - (-100))/10 = 20

List of scan template names (found in ‘scans.ipf’)

BabyDAC scans

ReadVsTime
ReadVsTimeUntil
ScanBabyDACUntil
ScanBabyDAC2D
ScanBabyDACRepeat
ScanBabyDAC_SRSAmplitude

FastDAC scans

ReadVsTimeFastdac
ScanFastDAC
ScanFastDacSlow
ScanFastDAC2D
ScanFastDACRepeat
FDacSpectrumAnalyzer

ScanBabyDAC_SRSAmplitude is just an example function to show how you can change some other instrument in the y-axis

‘Pseudo code’ edit of ScanBabyDAC to also ramp lock-in

```
function ScanBabyDAC_SRSAmplitude(babydacID, srsID, startx, finx, channelsx,
numptsx, delayx, rampratex, starty, finy, numptsy, delayy, [comments, nosave])

        // Reconnect instruments
    sc_openinstrconnections(0)

    // Initialize ScanVars
    InitBDscanVars()

    // Check software limits and ramprate limits
    SFbd_pre_checks()

    // Ramp to start without checks because checked above
    SFbd_ramp_start()

    // Let gates settle
    sc_sleep()

    // Make waves and graphs etc
    initializeScan()
```

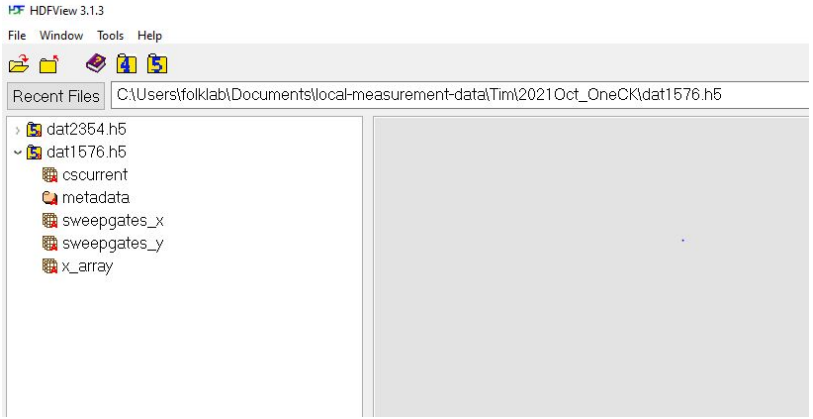
```
    // main loop
    variable i=0, j=0, setpointx, setpointy
    do
        setpointx = S.startx
        setpointy = starty + (i*(finy-starty)/(S.numptsy-1))
        RampMultipleBD(S.instrID, S.channelsx, setpointx, ramprate=S.rampratex, ignore_lims=1)
        SetSRSAmplitude(srsID,setpointy)
        sc_sleep(S.delayy)
        j=0
        do
            setpointx = S.startx + (j*(S.finx-S.startx)/(S.numptsx-1))
            RampMultipleBD(S.instrID, S.channelsx, setpointx, ramprate=S.rampratex, ignore_lims=1)
            sc_sleep(S.delayx)
            New_RecordValues(S, i, j)
            j+=1
        while (j<S.numptsx)
        i+=1
    while (i<S.numptsy)

    // Save by default
    if (nosave == 0)
        EndScan(S=S)
    else
        dowindow /k SweepControl
    endif
end
```

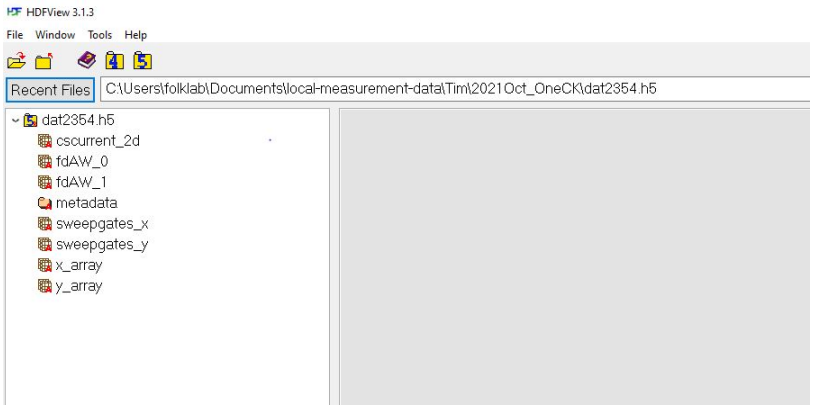
Manual inspection of hdf file

- Click on file in C:\Users\folklab\Documents\local-measurement-data\<your name>\<experiment>
- Or open application ‘HDFViewer’ and ‘open file’

1D Scan

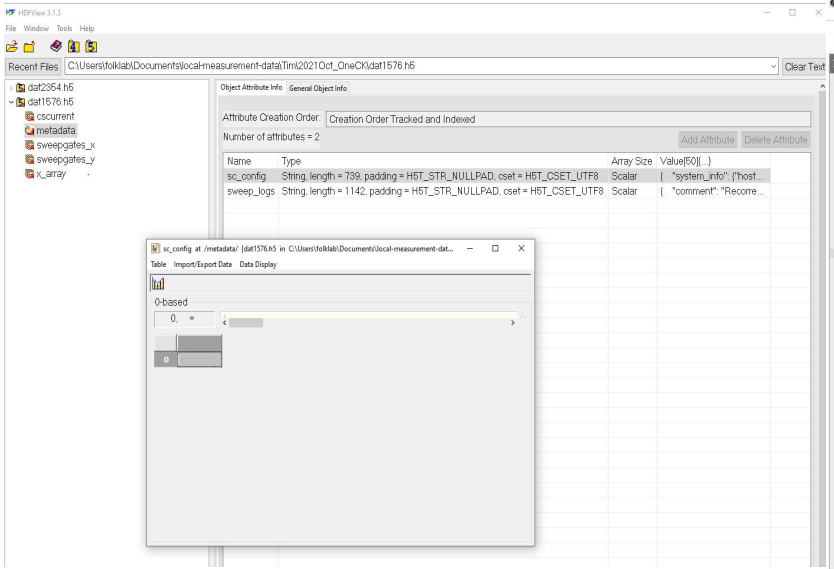


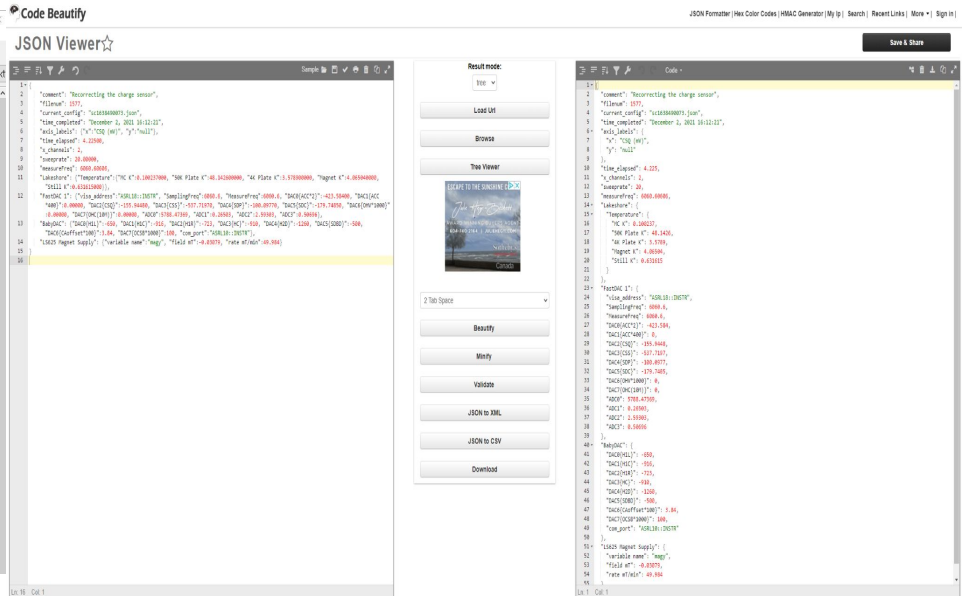
2D Scan



Manual inspection of sweeplogs

1. Click on ‘metadata’
2. Click on ‘sweeplogs’
3. Copy (ctrl-c) the cell
4. Go to a json viewer website
5. https://codebeautify.org/jsonviewer
6. Paste (ctrl-v) and click ‘beautify’





Using Python to parse hdf files

```
import h5py

import json

with h5py.File(<path-to-file>) as f:

      cscurrent = f['cscurrent '][:]

      config = f['metadata'].attrs['sc_config ']

      sweeplogs = f['metadata'].attrs['sweep_logs']

      sweepgates_x= f['sweepgates_x'][:]

      x_array= f['x_array'][:]


d = json.loads(sweeplogs)

fdac1f = d['FastDAC 1']['MeasureFreq']
```

1. We open the file with the path name, this can be relative or absolute
2. We ask for an attribute of 'metadata' in this case we want 'sweep_logs'
3. Turn the sweep_logs into json format
4. Pull the 'MeasureFreq' from 'FastDAC 1'

Using Regex to find hdf file path from datnum

```
import re

import os

datnum = 42

regex = re.compile(r'.*dat' + str(datnum) + '.h5$')

path = <path-to-folder>

filename = []

for root, dirs, files in os.walk(path):

    for file in files:

        if regex.match(file):

            if full_path == 1:

                filename.append(path + '/' + file)

            else:

                filename.append(file)
```

1. Set a regex expression that will check if the string contains 'dat<dat-num>.h5'
2. Look at all the files in the folder that is provided in path
3. If the filename matches the regex expression, append to filename list

Generally filename should only be 1 element long, but sometimes you may find that you have multiple files with the same dat number. So checking all the files which match the regex is good practice to test for duplicates.

Instrument not responding correctly

print fd, bd, magy, bd_window_resource
    1 2 2 1

Here we see that bd and magy are equal to the same variable and IGOR is getting confused.

Possible solution

- killvariables fd, bd, magy, bd_window_resource
- killvisa()
- Click 'Connect Instr' in the ScanController or ScanControllerFastDAC window

Check if each variable is now equal to a unique integer
print fd, bd, magy
    1 2 3

Bytes left in FastDAC buffer

Sometimes if bytes are 'left'/'stuck' in the FastDAC buffer, a function will abort with the error

[ERROR] "getFDACOutput": Bad response:

Possible solution

- clearfdacBuffer(fd)