# *RFWScopeDaq Software Guide*

## Contents



| Date | Version | Description | Author |
|---|---|---|---|
| August 2023 | 1-0 | A command line tool for continuously saving C100 scope-mode waveform records | Theo McGuckin |
| Sept 2023 | 1-1 | Updates after Summer 2023 C100 cavity changes | Theo McGuckin |

## Overview

The C100 field control chassis (FCC) input/output controllers (IOCs) are slated to be improved with additional buffers to present waveform records in near real-time. This software project is intended to save those records to disk on demand. This data is intended to be used in the development of AI models for predicting RF faults, similar to existing post-mortem analysis done by the certified software project rf_classifier.

# Software Location and Operation

The software has been added to the **Accelerator Software Group's CSUE application structure.**

Run **RFWScopeDaq -h** to output help file:

## Help:

- -h, --help show this help message and exit
- -z ZONE, --zone ZONE EPICS name of a zone to check. E.g. R1M
- -c CAVITY, --cavity CAVITY EPICS name of cavity to check. E.g., R1M1
- -e EMAIL [EMAIL ...], --email EMAIL [EMAIL ...] Space separated list of email addresses to report
- -q, --quiet Suppresses text output
- -n N_SAMPLES, --n-samples N_SAMPLES Number of samples to collect per cavity
- -o TIMEOUT, --timeout TIMEOUT How long each sample should wait for stable operations.
- -t TIME, --time TIME time to gather data in minutes, default = 5 minutes
- -d DIR, --dir DIR directory to save data, default = '/tmp/c100-rfw-scope-data/'
- -v, --version show program's version number and exit

-z or -c (a zone or single cavity) is a required argument

## Running the code:

**NOTE:** As of Summer 2023 OPS **channel access** to RF IOCs is required for software to make changes to RF settings (RXXX TRGS1, TGD1, WFSCOPper). It is therefore necessary to either have channel access open, or run the software as a user with channel access (such as alarms). Approval must be obtained (either from the on-duty OPS crew or via the PD through ATLis/SWList system) before taking data runs.

From an OPS-fiefdom linux (RHEL7, currently) machine (recommended: **opsrfaidaq0**), run command:

**RFWScopeDaq -z=<zone>|-c=<cavity>**

(if command can't be found then use full path: **/cs/prohome/bin/RFWScopeDaq**)

The **-z/--zone** (for a c100 zone) or **-c/--cavity** (for a single c100 cavity) option is required (ex. **RFWScopeDaq -z=R1O**). This will toggle the FCC from trip to scope mode, , collect waveforms/data for 5 minutes (default time), then restore the original configuration, and save the waveforms/data to a tab-separated-file (.tsv) on the local system (default base location: **/tmp/c100-rfw-scope-data/**)

**Note:** opsrfaidaq0 has a large-capacity dedicated filesystem (**/data/waveforms**) for storing data. However, as this directory is not available on other systems, and the program can be run from any OPS-linux system, the default location to save data is still set to **/tmp/c100-rfw-scope-data**.

## Code Structure

Below is a code-tree, trimmed down to remove standard CSUE structure (for easier reading):

```
./1-0
|-- bin
|   |-- cavity.py
|   |-- collect_data.py
|   |-- config.py
|   |-- main.py
|   |-- RFWScopeDaq.csh
|   `-- _version.py
|-- configure
|   |-- Makefile
|-- doc
|   `-- user_guide -> /cs/dvlhome/doc/r/RFWScopeDaq/user_guide/D1-0
|-- Makefile
`-- support
    |-- csueLib -> /cs/dvlhome/apps/c/csueLib/2-5
    |-- epics -> /usr/dvlepics/epics/R3.14.12.3.J0
    |-- Makefile
    `-- python -> /usr/csite/pubtools/python/3.11.2
```

### Code/function description

**main.py** – main software call.

- Reads **config.py** to get default configure and arguments
- Reads command line arguments
- Calls **buildDirectory** to build directory structure for saving data (in '/tmp/c100-rfw-scope-data/' by default)
- Calls **process_cavities** to use threaded processes to concurrently start data taking cycle on each cavity requested via **collect_data.py:run_cavity_job**

**collect_data.py**

- Set **cavity** to **scope mode** (via **cavity.scope_mode**)
- Then goes into a loop for duration, waits for signals and collects them (via **cavity.get_waveforms**) into a dataframe
- Writes dataframe to file (via **writeFiles**)

**config.py** – config file with defaults for:

- C100 **zones**
- **Channels** to save
- **Duration** to gather data
- **Basedir** to save data in
- **Email** individuals when run complete

- **Verbose** mode on or off

**cavity.py**

- Contains functions to initialize cavity, change state, cycle through sequencer, monitor state, get waveforms and exit cleanly

## A Sample configuration for daily data runs

**Summer 2023:**

Data runs to be performed as user alarms on opsrfaidaq0 with data stored in /data/waveforms/DAILY/YYYY_MM_DD/RXX/RXX1

A cronjob has been configured on opsrfaidaq0 as user alarms:

```
# run cronjob to take data for each zone for five minutes daily
0 6 * * * /cs/prohome/bin/RFWScopeDaq -z=R1N -t=5 -d=/data/waveforms/DAILY/  >> /tmp/c100n.log
0 6 * * * /cs/prohome/bin/RFWScopeDaq -z=R1O -t=5 -d=/data/waveforms/DAILY/  >> /tmp/c100n.log
0 6 * * * /cs/prohome/bin/RFWScopeDaq -z=R1P -t=5 -d=/data/waveforms/DAILY/  >> /tmp/c100n.log
5 6 * * * /cs/prohome/bin/RFWScopeDaq -z=R2N -t=5 -d=/data/waveforms/DAILY/  >> /tmp/c100s.log
5 6 * * * /cs/prohome/bin/RFWScopeDaq -z=R2O -t=5 -d=/data/waveforms/DAILY/  >> /tmp/c100s.log
5 6 * * * /cs/prohome/bin/RFWScopeDaq -z=R2P -t=5 -d=/data/waveforms/DAILY/  >> /tmp/c100s.log
```

**Spring 2022:**

As part of the long term C100 data gathering project, it was requested that a cronjob be setup to run the program on daily basis, and then process and copy the data over to another location as well as log any output/errors. This involved setting up a wrapper script and cronjob on an OPS system (initially opsl77, should now be **opsrfaidaq0**):

*0 9 * * * wrapper.csh R1O >> /tmp/c100n.log 2>&1*

The script “wrapper.csh” handled error checking on input, as well as post-processing of the data once the program was done running:

```
#!/bin/csh

# req ops - the hard way
set tmpfile=`/cs/prohome/bin/reqExec ops`
source $tmpfile
rm -f tmpfile

# make sure a zone was included
if ( "$1" == "" ) then
      echo "missing zone (ex. \"R1O\")"
      exit
endif

# set signal to check RF on
#signal="${1}5RFONr"
set signal="${1}XKDOUT1r"
# check value
set return_value=`caget -t $signal`
```

```
# check if a value was returned (signal not mistyped)
if ( "$return_value" != "" ) then
    # if value 7 = all cavities off, skip; otherwise run script
    if ( $return_value != 7 ) then
        # run script
        /cs/prohome/bin/RFWScopeDaq-z=$1 -t=10
        # tar up the data and move it to c100 dir
        tar -cvf /usr/opsdata/waveforms/c100/$1_`date +"%Y_%m_%d"`.tar /tmp/c100-rfw-scope-data/$1*
        # zip it up to save space
        gzip /usr/opsdata/waveforms/c100/$1_`date +"%Y_%m_%d"`.tar
        # rm tmp files
        rm -rf /tmp/c100-rfw-scope-data/$1*
        # copy file over to CUE for Monibor
        scp -p /usr/opsdata/waveforms/c100/$1_`date +"%Y_%m_%d"`.tar.gz  labl1:/group/felteam/C100_DATA/
    else
        echo "Zone turned off, skipping"
    endif
else
    echo "Signal ${1}XKDOUT1r unreachable, skipping"
endif
exit
```