RS Spectrometer User Guide

E. Barr 19/02/20

Overview

This document contains a general guide to the recording system for the Rohde Schwarz spectrometer. Covered are the setup of the recording system, the running of a capture and the inspection of output data.

Prerequisites

The user will need access to the following servers:

- EDD-dev (only for testing in the MPIfR)
- Fpgdev (production machine to be taken to the testing site)

Access can be requested from it-support@mpifr-bonn.mpg.de. When requesting access, the users should also ask that they are included in the “docker” user group (group number 999).

The user will also need the following:

- An SSH client that supports X window forwarding
- An X11 server (XQuartz for OSX, Xming for Windows)
- A VNC client (optional)

Setting up a VNC server (optional)

Tools are available on both EDD-dev and fpgdev to run a VNC server. TightVnc and ICEWM are both installed and users are free to use their desired setup. A possible setup is below:

In the users ~/.vnc/xstartup file, put the following:

```
#!/bin/sh
xrdb $HOME/.Xresources
xsetroot -solid grey
vncconfig --nowin &
xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
/usr/bin/icewm-session
```

A session can then be started using:

```
vncserver -geometry 2560x1440 :<desired session number>
```

It is up to the user to set their VNC password using “vncpasswd”.

[NOTE: Care must be taken that the VNC sessions do not interfere with the Ethernet capture required for the RS Spectrometer. To this end, it is recommended to taskset VNC servers to keep them away from the CPU cores that are used by the spectrometer. Details below.]

Preparing for a capture

Starting the FPGA

[Amit to add some note here]

Starting the capture container

The environment for running a capture is held in a Docker container. The image used for the container is called `fsw_spectrometer:latest`. This can be seen on both machines by using the `docker images` command.

Before starting a new capture container, the user should first check if one is already running. This can be done using `docker ps -a`.

```
$ docker ps -a
CONTAINER ID   IMAGE          COMMAND   CREATED          STATUS       PORTS   NAMES
fd7363ad5ed7   7d5dec37a3be   "bash"    39 hours ago     Up 39 hours
fsw_capture_container
```

If the status is “Up”, then there is no need to generate a new container and the container can be logged into using:
`docker attach fsw_capture_container.`

If the status of the container is “Exited”, then the user can attach to the container using:
`docker start -i fsw_capture_container`

If there is no container capture container at all in the list returned, then the user can start a new capture container using the following (different setups per server):

On EDD-dev:

```
docker run \
    -ti \
    --privileged \
    --name=fsw_capture_container \
    --device=/dev/infiniband/issm0 \
    --device=/dev/infiniband/issm1 \
    --device=/dev/infiniband/rdma_cm \
    --device=/dev/infiniband/ucm0 \
    --device=/dev/infiniband/umad0 \
    --device=/dev/infiniband/umad1 \
    --device=/dev/infiniband/uverbs0 \
    --ulimit=memlock=-1 \
    --ulimit=core=0 \
    --net=host \
    -v /data/FSW_SPECTROMETER/DATA/:/data/ \
    -v /data/FSW_SPECTROMETER/CONFIG/:/config/ \
    -w /root/RFI_chamber \
    fsw_spectrometer:latest bash
```

On FPGDev:

(TBD)

[NOTE: In the above configurations the -v flag denotes path mounts from the host system to the container. The user is free to set these to whatever they please although it should be noted that the new path name in the container is the path that should be set in the configuration files when running the capture.]

Running a capture

Once the user has started/attached to the capture container, they should find themselves in the `/root/RFI_chamber` directory (if not, navigate your way there). In this directory they will find the script that is used for the capture: `capture_data.py`

```
root@edd-dev:~/RFI_chamber# ./capture_data.py -h
usage: capture_data.py [-h] --config FILE [--dry-run] [--log-level LEVEL]
                       [--log-dir DIR]

Perform a data capture from the RFI chamber recording system

optional arguments:
  -h, --help          show this help message and exit
  --config FILE       The YAML measurement configuration file
  --dry-run           Do not send configuration requests to the spectrum
                      analyser only queries
  --log-level LEVEL   The logging level (DEBUG, INFO, WARNING, ERROR, CRITICAL)
  --log-dir DIR       A directory to output logs to, if no directory specified
                      logs will only go to stdout
```

The “`dry-run`” option should mostly be avoided as this is only to allow testing of the recording without explicit configuration of the Rohde & Schwarz spectrum analyser (here on referred to as the FSW).

The most critical option is the --config. This is the YAML file that will dictate the configuration of FSW and the recording parameters. An example can be found in `defaults.yaml` in the `/root/RFI_chamber` directory. As noted above, the output directories specified in this configuration file should be the paths as seen inside the container, not the paths on the host (e.g. we want data in `/data/FSW_SPECTROMETER/DATA/` on the host, but we have mounted this to the path `/data/` in the container, therefore we should set the output path in the configuration file to `/data/`)

Logging information will always be put to stdout/stderr. If the user also wishes to preserve the logs, it is recommended to use the `--log-dir` argument to set a directory to which an additional log file will be written.

Once the user has generated their desired configuration, a capture can be started using the following (here specifying only info level logging and requesting a written log file to also be produced in the `/data/` directory):

```
$./capture_data.py --config <config file> --log-level=info
--log-dir=/data/
```

[NOTE: The configuration does not need to specify any measurements, but can instead just be used to specify the list of SCPI commands required to initialise the FSW. An example of such a configuration file can be found at /root/RFI_chamber/init.yaml. The command to apply the configuration is the same as that shown above.]

Viewing captured data

Manual inspection

To view the data manually, navigate to the output directory (either inside the capture container or on the host).

```
(py3) ebarr@edd-dev:/data/FSW_SPECTROMETER/DATA$ ls
2020-02-19T09:26:38_rfi_chamber.log
LowRange_500-3600MHz_100Hz_6sec_1114.4_1582110078392.rfi
LowRange_500-3600MHz_100Hz_6sec_1114.4_1582110078392.bin
LowRange_500-3600MHz_100Hz_6sec_704.8_1582106689794.rfi
LowRange_500-3600MHz_100Hz_6sec_704.8_1582106689794.bin
```

The “.rfi” files contain a simple ASCII header:

```
$ cat LowRange_500-3600MHz_100Hz_6sec_1114.4_1582110078392.rfi
Data:
Center Frequency in Hz: 1114400000.0
Bandwidth in Hz: 512000000.0
Number of Channels: 524288
Frequency Spacing: uniform
Integration time in milliseconds: 20000.768
```

```
Unique Scan ID: 1582110078392
Timestamp: 1582110078392
User Friendly Name: LowRange_500-3600MHz_100Hz_6sec
Data Acquisition System: Spectromonster
Chamber Calibration: default
Antenna Calibration: default
Cable Calibration: default
LNA Calibration: default
Background Data: unknown
Chamber Type: reverb
```

The “.bin” files contain binary host-order (little-endian) single precision floating point numbers. These are intended to be viewed by the GUI viewing tool, however if a quick look is needed either IPython on GNUPlot can easily be used:

GNUPlot:

```
$ gnuplot
gnuplot> set term x11 enhanced
Terminal type set to 'x11'
Options are ' nopersist enhanced'
gnuplot> plot 'LowRange_500-3600MHz_100Hz_6sec_1114.4_1582110078392.bin' binary
array=524288 format='%float' with line lt -1 lw 1
```

[Note: the user should check the corresponding .rfi file as the number of channels “array=<nchans>” must be manually set].

IPython:

```
$ ipython --pylab
In [1]: from matplotlib.pyplot import *
In [2]: ion()
In [3]: import numpy as np
In [4]: x =
np.fromfile("LowRange_500-3600MHz_100Hz_6sec_1114.4_1582110078392.bin",dtype="float32")
In [5]: plot(x)
Out[5]: [<matplotlib.lines.Line2D at 0x7f918e84c748>]
```

Viewing with MVIS

(TBD)