HOWTO start the UHDAS GUI and collect data remotely (ssh only; no screen share)

Core commands and concepts are at the top, details follow.

NOTE: Examples of commandline output come from different sessions -- at present they only give a flavor of what to expect.

Core commands

(1) ssh in to the UHDAS computer on the ship

Do whatever you have to for that ship

(2) Start a **"screen session"**

(see below)

(3) In the "screen" session:

(4)

(a) Ensure that any graphics which show up will go to the console at sea. Run this:

```
export DISPLAY=:0.0
```

(b) It is **UP TO YOU** to make sure it is OK to quit the DAS etc.

(c) Quit the existing version:

```
DAS.py --quit
```

(d) Start DAS.py in the "screen session"

```
DAS.py --read_stdin
```

(e) Start a cruise (it is OK to re-use a cruise name)

```
start_cruise  test-2019-08-05_test1
```

(f) Start recording (annoying to have “start_logging” here)

```
start_logging
```

(g) Stop recording

```
stop_logging
```

(h) End cruise

```
end_cruise
```

(i) Quit the GUI

```
quit
```

How to tell what is happening with UHDAS without a GUI

Specifically, as different parts of UHDAS start up, they write messages to files.

The components of UHDAS are:

- “Flag” file is a file in `/home/adcp/flags` with the PID of the program running
- “Log files” are in `/home/adcp/log`
- Processes are found with “**`ps -ef | grep xxx`**”

status	action	DAS (or other) program	Flag file	Log file(s)	description
DAS is running	“Start the UHDAS GUI”	DAS.py	DAS.running	DAS_main.log	UHDAS GUI is up, standing by
Cruise is active	“Click Start Cruise; fill In the name”	DAS_while_cruise.py	DAS_while_cruise.running	DAS_while_cruise.log	Cruise directory was made; ready for acquisition
DAS is	“Click Start	ser_bin	DAS.logging	asc2bin.log	Acquisition

logging	Recording”	ser_asc (for serial) udp_asc (for udp)			processes spawned
processing is occurring	(happens after Recording starts)	DAS_while_logging.log	DAS_while_logging.running	(timers go Off for Various processes)	Web site figures are populated
Speedlog Is running	(happens after Recording starts)	DAS_speedlog.log	DAS_speedlog.running	DAS_speedlog.py	Web Speedlog should show values

(1)Look in /home/adcp/flags directory to see what is running

status	**action**	**Flag file**	**Description -- what SHOULD be happening**
DAS is running	Start UHDAS GUI	DAS.running	UHDAS GUI is up, standing by
Cruise is active	Start Cruise	DAS_while_cruise.running	Cruise directory was made; ready for acquisition
DAS is Logging and processing Is occurring Speedlog is running	Start Recording	DAS.logging DAS_while_logging.running DAS_speedlog.running	Acquisition processes spawned; Web site is update regularly Speedlog is running

(2)Look at processes to see what is running:

<u>Is the GUI up?</u>

```
ps -ef | grep DAS.py
```

If yes, you will get something like this:

- one long commandline (underlined)
- shows ‘yes’ DAS.py’ is running

```
adcp     13231 11354  2 16:53 pts/7    00:00:11 /usr/bin/python3
/usr/local/currents/bin/DAS.py --read_stdin
adcp     14354 10580  0 17:01 pts/1    00:00:00 grep DAS.py
```

Is a cruise active?

```
ls -l /home/adcp
```
if yes, there is a symbolic link 'cruise' pointing to cruise dir

If yes, you would see something like this:

yes, current cruise is called "rtest02"

```
lrwxrwxrwx  1 adcp efiring      18 Aug  6 16:53 cruise -> /home/data/rtest02
```

Is acquisition occurring?

```
ps -ef | grep _bin # acquisition from ADCPs
ps -ef | grep _asc  # NMEA messages
```

yes, one ADCP and
2 ascii streams are being logged

```
ticurrents02:PY3(~)$ ps -ef | grep _bin
adcp     13321     1  6 16:54 pts/7    00:00:36 /usr/local/bin/ser_bin -y 2019
-P ttyUSB0 -b 38400 -d /home/data/rtest02/raw/wh300 -i
/tmp/SerialLogger/inpipe.ttyUSB0 -o /tmp/SerialLogger/outpipe.ttyUSB0 -T
1565110495 -f zzz -F -m 1 -H 2 -e raw -lE -c -O -I -Z tcp://127.0.0.1:38010

ticurrents02:PY3(~)$ ps -ef | grep _asc
adcp     13326     1  0 16:54 pts/7    00:00:00 /usr/local/bin/ser_asc -y 2019
-P ttyUSB2 -b 9600 -d /home/data/rtest02/raw/gyro -i
/tmp/SerialLogger/inpipe.ttyUSB2 -o /tmp/SerialLogger/outpipe.ttyUSB2 -T
1565110495 -f zzz -F -m 1 -H 2 -e hdg -c -Y2 $HEHDT
adcp     13329     1  0 16:54 pts/7    00:00:00 /usr/local/bin/ser_asc -y 2019
-P ttyUSB3 -b 9600 -d /home/data/rtest02/raw/gpsnav -i
/tmp/SerialLogger/inpipe.ttyUSB3 -o /tmp/SerialLogger/outpipe.ttyUSB3 -T
1565110495 -f zzz -F -m 1 -H 2 -e gps -c -Y2 $INGGA $GPGGA $PASHR
```

Is speedlog running?

```
ps -ef | grep DAS_speedlog.py
```
#will be here if running

If yes:

yes, DAS_speedlog.py is running

```
adcp     13338     1  2 16:54 pts/7    00:00:13 /usr/bin/python3
/usr/local/currents/bin/DAS_speedlog.py replace
adcp     15227 10580  0 17:05 pts/1    00:00:00 grep DAS_speedlog.py
```

Is processing occurring?

  # This is not the way to find out: processing occurs sporadically, not “always running”

Look in `/home/adcp/log/DAS_while_logging.log`, and
- Figures have recent dates

(3) Look in the /home/adcp/log directory to see what messages are there.

DAS_main.log

```
tail -300 /home/adcp/log/DAS_main.log # last 300 lines
less /home/adcp/log/DAS_main.log # page through using ‘less’
```

Entries include:

# starting the GUI
```
2018-12-07 20:34:44,801 INFO     DAS            Starting DAS.py
2018-12-07 20:34:44,817 INFO     DAS            DISPLAY is :0.0
```

# starting a cruise

```
2019-08-01 10:47:15,273 INFO     cruisesetup    Initializing yearbase from
current date: 2019
2019-08-01 10:47:17,183 INFO     cruisesetup    StartCruise, new, cruiseid is
ar35-05
```

# start logging (start recording)

```
2019-08-03 16:56:39,987 INFO     DAS            entering StartLogging, resume =
0, auto = False

2019-08-03 16:56:40,004 INFO     DAS            Commands for wh300:
```
  [commands snipped]
  [command block exists for each instrument]

```
2019-08-05 20:43:02,179 INFO     DAS            Logging started
```

# stop recording

```
2019-07-29 08:39:58,448 INFO     DAS            Logging stopped
```

# end cruise

```
2019-07-29 08:40:02,603 INFO     cruisesetup    EndCruise, cruiseid is ar35-04
```

# kill the UHDAS GUI

```
2019-07-29 08:40:07,105 INFO     DAS            Closing DAS GUI
```

DAS_while_cruise.log

- DAS_while_cruise.py controls two things:
    - Backups
    - Updating the ‘reports’ directory

DAS_while_logging.log

- DAS_while_logging.py controls:
    - `run_lastensq.py` (stages the 5-min averages and makes the 5-min profile plot)
    - `run_quick.py` (puts 5-min averages into database)
    - `run_3dayplots` (makes contour and vector plots on web page)
    - [many others]
    - Each of these has its own log file (eg. 3dayplots_os150bb.log
- These processes and their arguments AS CALLED are all listed in `/home/adcp/uhdas_tmp/repeaters.txt`

DAS_speedlog.log

- Lists speedlog parameters and says when it is starting and stopping
- See “speedlog” Troubleshooting HOWTO

term*.log

Each instrument gets a new log file every time the UHDAS GUI starts.
Example:

```
arcurrents01:(log)$ ls -l term*
-rw-r--r-- 1 adcp efiring  9748 Jul 29 08:40 termos150log2019_173_56340.txt
-rw-r--r-- 1 adcp efiring  8952 Aug  5 20:42 termos150log2019_212_38819.txt
-rw-r--r-- 1 adcp efiring 19572 Jul 29 08:40 termos38log2019_173_56340.txt
-rw-r--r-- 1 adcp efiring  8989 Aug  5 20:42 termos38log2019_212_38819.txt
```

```
-rw-r--r-- 1 adcp efiring 13728 Jul 29 08:40 termwh300log2019_173_56340.txt
-rw-r--r-- 1 adcp efiring  3432 Aug  5 20:42 termwh300log2019_212_38819.txt
```

Each file has

- A wakeup message (eg. model and firmware)

```
Ocean Surveyor Broadband/Narrowband ADCP
Teledyne RD Instruments (c) 1997-2008
All rights reserved.
Firmware Version: 23.17
```

- A dialog with the instrument saying the commands that were sent
- A dump of all the commands and the instrument values

Other notes:

- Warnings are stored in files ending in *.warn
- Older files are moved out of /home/adcp/log to /home/adcp/morgue (for later examination)

Details about “screen”

When you run a “screen” session you are in a bash shell as if you had logged in, but several things are different:

(1) You can “detach” from a screen session and you can “reattach” to a screen session
(2) If you “detach” from the screen session, any process that you started will KEEP RUNNING, whereas other mechanisms (“nohup” and backgrounding) do not seem as reliable for this.
(3) Screen uses control key combinations so watch out! For example **control-a** followed by another letter (or just using other control sequences) is the way to make “screen” do things, so if you are editing with emacs or using emacs shortcuts (which use control-a) you will be surprised. Screen uses other commands too, but typing normal bash commands won’t get you into trouble.

    Control sequences are often written in two ways

    `control-a`

^a
“Screen” control sequences do NOT have a space, but they are normally written that way for easier reading. The following are the same:

```
control-a ?                 # how we write it
^a?                         # what you type
```

(4) You can see all the commands that can be run inside screen by typing

```
control-a ?
```

(5) Here are some common scenarios:
   (a) Start a screen session

```
screen
```

   (b) Detach from a screen session

From within a screen session type

```
control-a d
```

   (c) Ask what screen sessions exist

```
screen -ls
```

In this example the results are:

```
There are screens on:
     15760.pts-3.moli      (08/05/2019 09:49:29 AM)    (Detached)
     15754.pts-3.moli      (08/05/2019 09:49:20 AM)    (Detached)
2 Sockets in /run/screen/S-jules.
```

   (d) Reattach to an existing screen session
   If there is only one existing session,

```
screen -r
```

If there are multiple screens (as above), choose one:

```
screen -r 15760.pts-3.mol
```

(e) Answer the question: Am I in a screen session?
One simple way to test is to ask for help.

```
control-a ?
```

Answer: If you get a screen full of instructions, you are in “screen”
Answer: If you get a ‘?’ then you are in a bash shell, not s a screen session

(6) **Advanced**: you can use screen to monitor serial data:

(a) Syntax:

```
screen /dev/ttyUSB3 4800
```

(b) If there is something coming in, you will see it

(c) To QUIT (which is probably what you want to do)

```
control-a \
```

(then it asks if you want to do that; type the letter `y` for YES)

(d) ERROR: If your serial feeds are having trouble, check if you used “detach” instead of quit

(i) check the files in /var/lock and see if you have a lock file there. A file with this name would indicate a process is running which is using that serial port (such as ‘screen’). The file contains the process ID so you can tell what is using the port.

```
/var/lock/LCK..ttyUSB3
```

(ii) check the existing screen sessions to find out if one is still running. If you are still showing data in a ‘screen session,

1) Reattach to it
2) Use `control-a \` to EXIT rather than detaching

J.Hummon
2019-08-05