Testing: Areas of Concern

The following are areas of concern that should be tested. The range in brackets '[file.c:start-stop]' correspond to the lines of code in files recorder/main.c and recorder/recorder_helpers.c, which are in the directory:

```
/opt/ti-sdk-am335x-evm-07.00.00.00/example-applications-dt78xx-examples/
```

that should be further inspected and tested. These areas of concern are sorted by priority. It can be assumed for know that the dependency libraries (RingBuf.c, libaiff.c, and sunriset.c) function properly. Any bugs in these programs can be addressed as is necessary; they, if any, will crop up as functions in the recorder program are tested. These areas also apply to the version of the recorder program as of 2018-07-18 at 14h00.

1. Command line interface cases, that is setting [main.c:71-111]:

   1.1. Samples per file
      - 1.1.1. Check for correct change of samples_per_file

   1.2. Clock frequency (sample rate)
      - 1.2.1. Check for correct change of clk.clk_freq
      - 1.2.2. Check for correct change of buffer_object.sample_rate

   1.3. Sampling duration in days
      - 1.3.1. Check for correct change of duration_days

   1.4. Daemonise (on/off by default); run in background
      - 1.4.1. Check that program is running in background as daemon

   1.5. Number of buffers
      - 1.5.1. Check for correct change of num_buffers

   1.6. Auto trigger (on by default/off)
      - 1.6.1. Check the status of the trigger (auto if 0 or default/threshold if 1)

   1.7. Enabled channels
      - 1.7.1. Check that all channels that were to be enabled are enabled (chan_mask)
      - 1.7.2. Check that all channels that were to be disabled are disabled (chan_mask)

   1.8. Safety margin
      - 1.8.1. Check for correct change of safety_margin

1.9. Check that the program exits in error when a flag not of the set `{s, c, d, r, b, t, i, m}` is used

2. In general, check that all error checks that result in `goto _exit` have expected behaviour. These error checks are:

   2.1. Sample rate configuration [`main.c`136-142]

   2.2. Trigger configuration [`main.c`:147-156; `recorder_helpers.c: configTrig()`]

   2.3. Create channel mask [`main.c`:158-163]

   2.4. Channel configuration [`main.c`:165-173]

   2.5. Initialise AIO structures [`main.c`:175-181]

   2.6. Allocation of array buffer dynamic memory [`main.c`:183=194].
      2.6.1. That is, check that buffer dynamically allocated correctly (that is correct size). (Note that this uses a manufacturer provided function.)

   2.7. Allocation of circular buffer dynamic memory [`main.c`:196-202]

   2.8. Submission of buffers [`main.c`:242-247]

   2.9. Initialisation of ARM [`main.c`:249-254]

   2.10. Software start [`main.c`:256-262]

   2.11. Status check of active channels [`main.c`:266-273]

   2.12. AIFF file creation:
      2.12.1. Formatting of file [`main.c`:283-285]
      2.12.2. Opening of file [`main.c`:287-303]
      2.12.3. Starting of writing [`main.c`:305-308]
      2.12.4. Sampling writing [`main.c`:310-315; `main.c: writeBuffer()`]
      2.12.5. Finishing of writing [`main.c`:317-322]
      2.12.6. Closing of file [`main.c`:324-333]

3. Check that ‘Ctrl+C’ aborts the program at any and all stages.
   3.1. Especially check that during sampling for both when active and inactive; that is awake at night and asleep during day.

4. Circular buffer (`RingBuf.c` implementation) [`recorder_helpers.c: writeBuffer()`]; check that:

    4.1. 'Read' pointer that reads the circular buffer and writes to file lags the 'Write' pointer that reads from the linear buffer array and writes to the circular buffer by the number of channels; no accidental overwriting

    4.2. 'Write' makes one cycle after writing all samples per a file

    4.3. Cycle continues in the circular path (see `RingBuf.c`)

    4.4. Repeats in one cycle for `num_buffers` (per file)

    4.5. If `samples_per_chan * num_buffs * num_channels > 65536` samples, fatal error occurs [`recorder_helpers.c: checkFatal()`]

5. Day/night (asleep/awake) on/off cycling [`main.c`:214-237]:

    5.1. Disable `NIGHT_CYCLE` to make sure samples (on) regardless of time [`recorder_helpers.h:90`]

    5.2. On during time period when `present` within (`sunset-safety_margin`) through (`sunrise+safety_margin`) [`recorder_helpers.c: calcSunUpDown()`]

    5.3. Off during time period when `present` outside (`sunset-safety_margin`) through (`sunrise+safety_margin`) [`recorder_helpers.h: calcSunUpDown()`]

    5.4. Updates elapsed day after `present` time ahead of `sunrise` [`main.c:237`]

6. Miscellaneous `recorder_helpers.c` functions, check that:

    6.1. `ledIndicators()`:
        6.1.1. if `status` is 1 & `streaming` is 1 -> LEDs on (a digital output)

    6.2. `getTime()`:
        6.2.1. `pres_time` is equal to time gotten by testing file (within some margin of error)

    6.3. `getTimeEpoch()`:
        6.3.1. GMT Unix epoch time returned equal to time gotten (within some margin of error)

6.4. `timestamp()`:

- 6.4.1. String is unique

6.5. `getPresentTime()`:

- 6.5.1. Same as for `getTime()`

6.6. `createChanMask()`:

- 6.6.1. Created channel mask `chan_mask` same as that returned by `IOCTL_CHAN_GET`

6.7. `configChan()`:

- 6.7.1. All channels are configured

6.8. `initTrig()`:

- 6.8.1. All triggers are configured

6.9. `calcSunUpDown()`:

- 6.9.1. Calculated times same as pre-calculated and known times

6.10. `writeBuffer()`:

- 6.10.1. Measure delay/lag time
  - 6.10.1.1. Between each channel
  - 6.10.1.2. Between loops for saving files
  - 6.10.1.3. Scaling with the number of channels being recorded
- 6.10.2. Measure time it takes to write file
- 6.10.3. Produce complete series of files
- 6.10.4. Measure sample drop
  - 6.10.4.1. Loss of data
  - 6.10.4.2. Scaling with number of channels being recorded
- 6.10.5. Write pointer does not overwrite read pointer
- 6.10.6. Satisfactory fidelity
  - 6.10.6.1. Qualitatively: does recording sound like the test input (from a waveform generator)?
  - 6.10.6.2. Objectively: is the spectrum of the recorded waveform sufficiently equivalent to the spectrum of the input?
    - 6.10.6.2.1. Threshold the peaks
    - 6.10.6.2.2. Cross correlation?
    - 6.10.6.2.3. Self loop analog out to analog

6.11. `checkID()`:

- 6.11.1. Any arbitrary, non-zero length string successful as ID

6.12. `checkRate()`:

        6.12.1. Any arbitrary, positive, non-zero float successful as sampling frequency

6.13. `openStream()`:
        6.13.1. Confirm that opening a real stream is successful, and that opening a bogus stream is a failure

6.14. `openAIN()`:
        6.14.1. Given above, confirm that opening a real analog input is successful, and that opening a bogus analog input is a failure

6.15. `waitBuffering()`:
        6.15.1. No segmentation fault occurs after calling this function, regardless of size of buffer and number of channels