Publishing data over Bluelink on the MCU

We use Bluebox/Bluelink to communicate between the SOC and the MCU. Bluebox is the service applications talk to for publishing and listening, and Bluelink is the wire protocol, which is specified using flatbuffers[1].

Data Format

Any data transferred between the SOC and the MCU must be specified as a flatbuffer schema, which is compiled into C (MCU) and C++ (SOC) headers. To complicate it further, we use two different compilers to generate the appropriate headers. Flatbuffer schemas are stored in .fbs files, see [2] for a guide on how to write flatbuffer schemas.

In AOSP flatbuffer schemas are located in `vendor/oculus/software/libs/libbluemsgs` for the MCU, they are located in `merlot/flatbuffers`. Helper schemas go in the `common` directory, and messages should go into the appropriate directory in the `msgs` directory hierarchy.

Note it is very important to keep the two copies in sync. This is not done automatically and must be done manually by anyone updating or adding new schemas.

MCU

On the MCU we use uBluebox[4] and uBluelink, which are both written in and provide a C API. The generated headers should be put into `merlot/include/flatbuffers.` For Facebook internal systems running buck, they can be built using the **`make fbs`** target. Note this will not update the pre-built headers, but they can be found in `ovrsource/buck-out/gen/Firmware/projects/merlot/flatbuffers/fbs-c/`. This can be done running **`scripts/gen_libraries.py`**.

Manually compiling C headers

For non Facebook systems, eg. Quanta, it is necessary to download and install the flatcc compiler from [3]:

```
$ cd ~/
$ git clone https://github.com/dvidelabs/flatcc.git
$ flatcc/scripts/setup.sh -a merlot-flatcc
$ cd merlot-flatcc
$ scripts/build.sh
$ build/merlot-flatcc
```

You can now build the flatbuffers like this:

```
$ cd <path to AOSP>/vendor/oculus/software/libs/libbluemsgs
$ ~/merlot-flatcc/bin/flatcc -o <output dir> -I `pwd` msgs/sensors/alt.fbs
$ ls <output dir>
total 8
-rw-rw-r--. 1 jes jes 2293 Aug 27 12:25 alt_reader.h
-rw-rw-r--. 1 jes jes 2834 Aug 27 12:20 sensor_value_reader.h
```

For examples on how to use these headers, please look at
`merlot/src/sensor/{alt,als}.c`

SOC

On the SOC side, we use `commandrouter` as our Bluebox implementation. `commandrouter` runs as an Android service and provides `libcmdclient` as it's application interface. `libcmdclient` handles the Android IPC and provides a C++ interface with Java bindings to be used by applications.

C++ headers are automatically generated in the AOSP build, provided the flatbuffer file is added to `vendor/oculus/software/tools/bluecapture/Android.bp`

Sample code

`bluecapture` is a sample tool using the interface, which can be used as a reference for how to subscribe to Bluelink topics via `commandrouter`. As of this writing, it currently supports listening to the altimeter, topic `msgs.sensors.Alt,` and the ambient light sensor topic `msgs.sensors.Als`. The source code for bluecapture can be found in `vendor/oculus/software/tools/bluecapture`.

1. https://google.github.io/flatbuffers/index.html
2. https://google.github.io/flatbuffers/flatbuffers_guide_writing_schema.html
3. https://libraries.io/github/dvidelabs/flatcc#quickstart
4. https://docs.google.com/document/d/1hOw9S8LphZvGO-fH1o0MViXMr0sM9ktioDQjk8mYIFo/