**Purpose:**

This document describes the purpose of the QLSMDECODE.PY file

**Background:**

When collecting and saving data to the SD Card, the QuickAI application saves data in a QLSM file.

This is based on the RIFF format.

More detail about the RIFF concept can be found here:

<https://en.wikipedia.org/wiki/Resource_Interchange_File_Format>

This format differs in the following way:

**High Level Differences & Notes:**

* The file consists of an array of equal sized blocks.
* Each block is exactly 4K bytes
* The 4K size was chosen to be a clean division of the FAT file system cluster allocation size.
* All blocks start with a 32bit tag, followed by a 32bit length (little endian)
* All things must be 32bit aligned in the file, with the exception of the JSON string object.
* All blocks are “padded” with ZEROs – Thus a tag=0, indicates end of data in this block.
* Important: A tag value 0 indicates “end of data” in the current block.

**Block 0 (the first block)**

* Uses the tag QLSM
* Contains a JSON tag followed by ZERO fill to the end of the block.
* The JSON object describes the capture configuration.

**Blocks 1 to N contain sensor data.**

* The TAG is SNSR
* The block contains 2 sub blocks with tags: HDR! And DATA
* The HDR! tag contains:
  + The 32bit SENSOR ID – which can be found in the JSON header
  + The Block Sequence Number (from 0 to N)
  + NOTE: If the capture system does not run out of memory the number sequence is 0 to N with no gaps. If the capture system runs out of buffer memory there will be gaps in the sequence numbers, the gaps indicate how many times the system did not have memory to handle a batch of sensor data.
  + The 64bit best-estimate time the data in this block begins in microseconds
  + The 64bit best-estimate time of the last sample in the block microseconds
    - Best estimate means or is caused by various buffering that occurs.
    - The S3 AI (and other) platforms often do not have a perfect matched clock.
    - Some sensors have their internal clocks that are synchronous with other clocks
    - Often timestamps are really 1mSec resolution
    - Sometimes frequencies do not divide, ie: 333.333333 hz rounding occurs
    - The exact moment (millisecond to microsecond resolution) when the capture DMA completes – and when the DMA interrupt is serviced and a timestamp can be determined may be off by some non-zero and non-deterministic amount of time
* The DATA tag contains:
  + Raw sensor data – in binary form.
  + The exact form is described in the specific JSON element

To convert the QLSM data into a CSV file, use this command:

python qlsmparser.py -d FILENAME.qlsm

The “-d” flag is for debug/verbose output

This will produce a CSV file for each sensor found in the data file.

Any AUDIO data will be converted into a standard Windows WAV file

Notes:

* Data files can become extremely large when converted to CSV files
* Using a 64bit version of python is strongly recommended
* Additional RAM is also very important