The ankle exo is now transmitting data in bytes. I've got a program set up with RealTerm to record that data in hex format: a pair of hex numbers is equivalent to one byte.

First, you probably want to take a look at the translation of the data from hex. I used a hex -> ASCII online tool to do this:

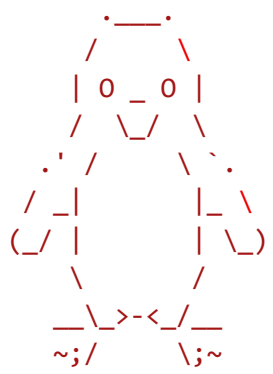
Ascii (String)

```
Startup received!


I ,shin_cal_tot,shin_cal_gyr,shin_cal_acc,shin_cal_mag |
foot_cal_tot,foot_cal_gyr,foot_cal_acc,foot_cal_mag |
shin_qw,shin_qx,shin_qy,shin_qz | foot_qw,foot_qx,foot_qy,foot_qz |
shin_ax,shin_ay,shin_az | foot_ax,foot_ay,foot_az
F ,time | val
L ,time | val
M ,time | potL_cur,potR_cur | potL_adj,potR_adj | targetPotL,targetPotR |
motorCmdL,motorCmdR
LOG,time | msg
ERR,msg


LOG,242263 | Motors started.
I □0□0□0□0□0□0□0□0'1□0□1□0□1o0□1□0&1□0□1ø0□1□0□1]0□1□
```

As before, the data starts by transmitting a set of headers; this is encoded in ASCII and can be directly translated. **This set of headers should be the first couple lines in the output CSV** (each on its own line).

Next comes the actual data. This will always start with an ASCII message of "LOG,TIME | Motors started." Boring. Just write that line to the CSV. Note: all LOG messages will be in ASCII format and have no specified length.

Note: unless specified as unsigned, assume all values are signed. For a 16-bit (2 byte) number, check if it is greater than (2^16) / 2 and subtract 2^16 from it if so.



See below for information on indicators…

Indicators

Message Type	Message Indication (→ASCII bytes←)	Expected Bytes (after indicator)	Byte Translation Key
IMU	→\nI ←	36	Byte num: comment 0: uint8 0-3 1: uint8 0-3 2: uint8 0-3 3: uint8 0-3 4: uint8 0-3 5: uint8 0-3 6: uint8 0-3 7: uint8 0-3 8-9: Float * 10^4 10-11: Float * 10^4 12-13: Float * 10^4 14-15: Float * 10^4 16-17: Float * 10^4 18-19: Float * 10^4 20-21: Float * 10^4 22-23: Float * 10^4 24-25: Float * 10^2 26-27: Float * 10^2 28-29: Float * 10^2 30-31: Float * 10^2 32-33: Float * 10^2 34-35: Float * 10^2
Motor	→\nM ←	20	Byte num: comment 0-3: uint32 4-5: uint16 6-7: uint16 8-9: int16 10-11: int16 12-13: uint16 14-15: uint16 16-17: uint16 18-19: uint16
FSR	→\nF ←	6	Byte num: comment 0-3: uint32 4-5: uint16
Load	→\nL ←	6	Byte num: comment 0-3: uint32 4-5: uint16
Log	→\nLOG,←	Undefined	ASCII