**SOLAR IN-CAR RASPBERRY PI 3**

**Config.txt**

The config.txt file defines how the CAN messages will be handled and how to allocate memory in the logging script. There are three numbers needed to know how much memory is needed for allocation.

1. NUM\_STM\_IN\_SYS: [ ]

* Defines the number of files that will be created
* Assumes you are making a different file per STM (sensor hub)
* Allows multiple files for a single STM

1. STM\_BUFFER\_SIZE: [ ]

* Defines the STM buffer size in bytes
* Once the STM buffer is full then the buffer will be emptied into the Raspberry Pi 3’s log file
* Recommended: make the buffer size larger than the number of entries in your log file. This will have a higher probability of fully populating the entry line.

1. MAX\_NUMBER\_SENSORS: [ ]

* Defines the max number of sensors in the entire system
* If this is less than the max number of sensors then there is a chance the system will segmentation fault if a large number of sensors are associated with one file

Once the above is defined, then the sensors are defined.

"SENSORS"

1. "FILE=[]" which will name the output log file (Ex: STM1 will turn into log file "STM1.csv")
2. "NAME=[]" which will tell the log how to label the columns
3. "ID=[]" which tells the program which message ids are associated with sensors
4. "MSGFORMAT=[]" which tells the program how to use the values for the data

THIS ORDER IS IMPORTANT

* Fill in [] with the value you want
* FILE will be interpreted as a string
* NAME will be interpreted as a string
* ID will be interpreted as an integer
* MSGFORMAT has multiple inputs

MSGFORMAT:

Each element is separated by a ',':

[SensorName, ValueType, ByteIndex[MSB], ByteIndex[MSB-1],...]

Ex: if there was a value that we wanted the first byte to be just a signed byte of data:

[POWER,int8\_t,1]

if we wanted two bytes then we would if we want the second byte to be most significant:

[POWER,int16\_t,2,1]

SensorName MUST match the string used in NAME

valid ValueTypes:

* int8\_t,uint8\_t
* int16\_t,uint16\_t
* int32\_t,uint32\_t
* float

if you want to separate one ID into multiple columns then you would:

NAME=[,,,]

MSGFORMAT=[],[],[],[]

Using the NAMEs in the SensorName field make multiple formats

Ex:

NAME=[ORIENTATION\_X,ORIENTATION\_Y,ORIENTATION\_Z]

MSGFORMAT=[ORIENTATION\_X,int16\_t,2,1],[ORIENTATION\_Y,int16\_t,4,3],[ORIENTATION\_Z,int16\_t,6,5]

Make sure the separate values are all the same ValueType

Related functions: SVT\_CAN::init\_log()

**CAN Messages**

CAN messages are 8 bytes long with an identifier and the data. We separate the data into the first byte and the ladder 7 bytes. The first byte for every message will tell the logging script how to interpret the last 7 bytes based on the config.txt file. The team did not use the identifier number since the number is the message’s priority to get onto the CAN bus.

CAN MSG: { Sensor ID | D[1] | D[2] | D[3] | D[4] | D[5] | D[6] | D[7] } { CAN ID (static) }

**Logging**

The files are populated one line at a time. The CAN messages are saved in the file buffers. When the buffers have filled up, the buffer is emptied and common timestamps are logged. The sensor values populate the log files based on the config.txt file.

PROBLEM: If there are two sensor values with the same time stamp and the same sensor ID then the last received value will be the only value logged.

Related functions: SVT\_CAN::store\_canframe()

**UDP Messages**

As soon as a CAN message is received, the message is put into the correct buffer and sent out as a UDP message to the pit server. There is no handshake with the respective IP so if there is no connection it will attempt to send a message but nothing will happen and the Raspberry Pi 3 will continue to log data.

Related functions: SVT\_CAN::parse\_canframe\_struct()

Idle/gathering data: 0.752 A

Bootup: 1.039 A

3.76 W – 5.5 W