**Overview**

This takes input from a file(s) that define the CAN msg payload fields and the order and selection of payload fields for output in a comma separated file that can be used in MATLAB, spreadsheets, etc.

It uses the database files: CANID\_INSERT.sql and PAYLOAD\_TYPE\_INSERT .sql to look up CAN ids, and payload identification code number versus description.

Log files are inputted via STDIN, and output is to STDOUT.

Command line switches provide a few options for printing the input tables. Printing the table of csv selections produces a list of csv field position versus payload field description, which can be used in subsequent processing of the output.

**A. Compiling & Execution**

To navigate to the directory with cancnvtmatlab--

cd ~/GliderWinchCommons/embed/svn\_sensor/PC/cancnvtmatlab

The near the beginning of cancnvtmatlab.c are lines that can be cut & pasted to the command line for compiling and running. E.g. the following compiles and runs the conversion--

gcc -Wall cancnvtmatlab.c -o cancnvtmatlab && ./cancnvtmatlab csvlinelayout200321.txt csvfieldselect200321.txt < ~/GliderWinchItems/GEVCUr/docs/data/log200220-2.txt

In the above, csvlinelayout200321.txt, specifies the CANid versus payload, and csvfieldselect200321.txt specifies the csv field order and selection. The log file input is sent to stdin with ~/GliderWinchItems/GEVCUr/docs/data/log200220-2.txt which specifies the path/file.

Appending, e.g. “| tee log200220.csv” to the command would pipe the screen output to the screen and a file.

**B. Command line switch options**

Following ./cancnvtmatlab are the switch options. Entering the -h option produces the following “help”--

./cancnvtmatlab -h

Command line layout

./cancnvtmatlab <options> <path/file input file #1> <path/file input file #2> <path/file input file ...> e.g.

./cancnvtmatlab -t csvlinelayout.txt csvfieldselect.txt < ~/GliderWinchItems/dynamometer/docs/data/log190504.txt

options

h - help

t - print input tables

x - do not convert input data file

c - csv position lines between csv data lines

# - skip printing missing CAN ids

The -t prints the input tables, and -x skips converting the input data, so -tx prints the tables and quits. -t prints the tables, but the appear in the output before the csv data.

The -# option skips listing lines in the csv output that list the CAN msgs with CAN ids that are not in the specification input files.

**C. Payload field specification (T)**

Lines are identified by the first character on the line.

A # is a comment line and skipped.

A T in the character of the input line identifies it as a payload specification.

Note: Though two input files are being shown, one could combine them into one field. The lines in the input fields are identified by the first character rather than by the file name/type used on the command line. Since the payload field is a rather static type of data, and the csv selection might be change even though the payloads are the same, two files are being used.

The beginning of the example/test file has comments describing the columns.

# Each line in the following defines a payload field

#

# Each payload field has a unique identification number

#

# A CAN id msg can have many fields

#

# Each different payload arrangement has a unique code (PAYLOAD\_INSERT.sql)

# e.g. U16\_U16 defines two 16b unsigned integers, little endian in payload

#

# Several payloads have special arrangements, e.g. GPS lat/long/ht

#

# Current scheme does not handle payload definition conditional on a byte(s)

# e.g. the first byte of a function command specifies the following request.

#

# Column 1: Type of line: 'T' = Define CAN fields

# Column 2: index+1 in list array

# Column 3: CAN id (32b left justified format)

# Column 4: payload field number (1,2,3, etc.; 0 = not defined)

# Column 5: payload type code (see PAYLOAD\_TYPE\_INSERT.sql, and/or cancnvtmatlab.c #define)

# Column 6: Offset applied to CAN msg field before scaling

# Column 7: Scale applied after offset

# Column 8: payload ascii name

#

# Note: Time tick source triggering CSV line is identified as being the first entry

#

A some elaboration follows.

The number of CAN ids are limited by--

#define CANFLDLAYOUTSZ 512

Column 2 is merely a sequence number for an array.

Column 4 is the payload field number.

The number of payload fields are currently limited to five, I.e.

#define MAXNUMFIELDS 5

The current version doesn’t handle bit fields within a payload byte with a general scheme. The lat, lon, ht, number of satellites, etc., from the gps are handled with a special purpose field identification. Most payload field layouts can be handled by combinations of bytes. See the following for the specifications:

~/GliderWinchCommons/embed/svn\_common/trunk/db/PAYLOAD\_TYPE\_INSERT.sql

Even though CANID\_INSERT.sql has the payload type field, and given the CANID it would be possible to look up the payload type code number in PAYLOAD\_TYPE\_INSERT.sql, it was not done as a simplification of the routine, which is traded-off with the effort for constructing the input file.

For payloads of more than one byte a Little Endian payload is designated as Ux (unsigned) or Sx (signed), e.g. U8, U32, or S8, S32. Big Endian payloads such as used with the DMOC are I16 (“I” being an arbitrary prefix).

Example of PAYLOAD\_TYPE\_INSERT.sql payload specification --

('U8\_FF', 21, 5, ' [0]:[1]-[4]: uint8\_t, Full Float');

Specifies byte[0] as uint8\_t, followed by four bytes (Little Endian) float. The code number is 21, and it should have a dlc (payload count) of 5. The code in column 4 would 21.

Column 6 & 7: Offset is added to the decoded payload and then multiplied by scale.

**D. CSV line selections (S)**

“S” in the first character of the line signifies the line is a csv field selection line.

The first column is no longer used. Originally, this column specified the position in the csv line. Now, the position of the fields are simply in the order of the “S” lines as the are read-in.

The “id” column is the CAN filed id number from the payload field (“T” line).

The readings extracted from the payload fields are all converted to fp doubles (to save on the complication of having different variable types). Therefore, the format column is always a “f” type.