Embedded Software Exercise

# Goal

The exercise goal is to design and implement a [CAN](https://en.wikipedia.org/wiki/CAN_bus) messages generator.

# Target

Design and implement SW that emulates transmission of a desired scenario into CAN bus.

The emulation should be done by writing the generated stream of CAN messages to a file with the transmission timestamp. The required resolution is 100[uSec].

# Learning

In order to be able to design and implement this solution, the following topics should be learned

* [DBC file structure](http://socialledge.com/sjsu/index.php/DBC_Format)
* [CAN bus message structure](https://en.wikipedia.org/wiki/CAN_bus#Frames)

# Details

* Implement the following messages (Taken from dbc file of Honda CRV):

*BO\_ 464 WHEEL\_SPEEDS: 8 VSA*

*SG\_ WHEEL\_SPEED\_FL : 0|15@0+ (0.01,0) [0|250] "kph" EON*

*SG\_ WHEEL\_SPEED\_FR : 15|15@0+ (0.01,0) [0|250] "kph" EON*

*SG\_ WHEEL\_SPEED\_RL : 30|15@0+ (0.01,0) [0|250] "kph" EON*

*SG\_ WHEEL\_SPEED\_RR : 45|15@0+ (0.01,0) [0|250] "kph" EON*

*BO\_ 446 BRAKE\_MODULE: 3 VSA*

*SG\_ BRAKE\_PRESSED : 4|1@0+ (1,0) [0|1] "" XXX*

*BO\_ 380 POWERTRAIN\_DATA: 8 PCM*

*SG\_ ENGINE\_RPM : 23|16@0+ (1,0) [0|15000] "rpm" EON*

* + The implementation can be naïve, i.e. direct coding without developing a mechanism to parse DBC file and create structures automatically from the dbc file
* Messages frequencies:

|  |  |
| --- | --- |
| Message | TX\_FREQUENCY |
| WHEEL\_SPEEDS | 100Hz |
| BRAKE\_MODULE | 10Hz |
| POWERTRAIN\_DATA | 25Hz |

* Input parameters to the program
  + STRESS\_FACTOR = 1.0 - 20.0
    - This parameter is used as multiplier to the above table values, in order to perform stress test
    - Default value: [1.0]
  + DRIVE\_SESSION\_TIME = 1.0 sec – 30 min
    - Drive session emulation time. The program should run and generate the messages in a cyclic way with the TX\_FREQUENCY \* STRESS\_FACTOR \* DRIVE\_SESSION\_TIME in seconds, until the duration is reached.
    - Default value - [10 minutes]
* Assign a data pattern for each message (e.g. wheel speed, RPM)
  + The payload of the messages should be randomly changed in a logical way (gradual change in values) within the values range and resolution as defined in dbc
* Measure and analyze: jitter for the message rate ratio / transmission period
  + The average deviation of the message timestamp from the expected timestamp. Units: [mSec]
  + Try to optimize your SW so that this number will be minimal

# Implementation guidelines

* Programming language - C or C++
* Thread model - single threaded
* Use of "fancy" libraries or OS functions is not allowed
* Use of floating point is not allowed (fixed point is allowed)
* Variables size – up to 32 bit
* Assume low power MCU
* Monitor jitter and bandwidth per each message type

# Exercise deliverables

* The source code
* The output file containing lines with each including the following structure:
  + time stamp, message type, message data
* A summary report containing:
  + Number of sent messages
  + Jitter
  + Bandwidth
  + Any warnings/errors