Robustness test :

I) canCheck

All the tests will be done on the card (PCM) .

The objective of canCheck is to check if all can frames are well received and if all data are correct : no missing element . This program will read the data in a text file .

a) can frames

Here is a procedure to follow :

Open terminal and connect to the card :

- *Bring can0 interface up @500kbps*
  ip link set up can0 type can bitrate 500000

- *In a window, launch command* :
  canCheck -f file.txt
  → add **-S** if file with timestamps is present
  → add **-D** for debug
  → file.txt is file containing the data to be compared

canCheck integrated a function which compare data from one frame to another. This is made possible by a counter which increments and then its' easier to check bytes by bytes.

canCheck should return an error if :

- data are not incremented correctly
- data are in random mode

Or return 0 , if all is correct !

canCheck -f candump_225_ok.txt

Expected result	Test
Return 0	**OK**

In debug Mode :
canCheck -f candump_225_ok.txt -D

Expected result	Test
Frame : can0 652 [8] 00 00 00 00 00 00 00 00 **Frame : can0 652 [8] 01 00 00 00 00 00 00 00** **Frame : can0 652 [8] 02 00 00 00 00 00 00 00** **Frame : can0 652 [8] 03 00 00 00 00 00 00 00** **Return 0**	**OK**

a) Isotp frames

ISOTP frame is a protocol allows be useful for the transport of messages that exceed the eight bytes maximum payload of CAN frames.

Here is a procedure to follow :

Open terminal and connect to the card :

- *Bring can1 and can2 interface up @500kbps*
  ip link set up can0 type can bitrate 500000
  ip link set up can1 type can bitrate 500000

- *Start isotp server (in one window)*
  isotprecv -s 701 -d 700 can1 -z 9 -l

  → *isotprecv* allows to receive only data
  → we can't see interface name, Id, data length
  → But we can use candump or wireshark to see the entire frame

- *Start a sniffer in an other window :*
  candump can1 can0 -z 7 > file.txt &

- *Start isotpgen*
  isotpgen -s 700 -d 701 can0 -z 9 -D i -T i -n 255

We have different options for this function : *See below*

- **-D** : send PDU with len bytes
  - : enter a number *1<nbr<4095*
  - *'**i**' :* incremented length
  - '**r**' : random length

- **-T** : Data (payload) generation mode
  - *'**i**' :* incremented data
  - '**r**' : random data

We can juggle between the different options
isotpgen -s 700 -d 701 can0 -z 9 -D r -T I
isotpgen -s 700 -d 701 can0 -z 9 -D 4095

Expected result with **isotpgen -s 700 -d 701 can0 -z 9 -D i -T I -n 255** Test

00 **01 00** **02 00 00** **03 00 00 00** **04 00 00 00 00** **05 00 00 00 00 00** **06 00 00 00 00 00 00** **07 00 00 00 00 00 00 00** **[…] [..]**	**OK**

Now, open the file.txt :

Test

can1 700 [2] 01 00 **can1 700 [3] 02 01 00** **can1 700 [4] 03 02 00 00** **can1 700 [5] 04 03 00 00 00** **can1 700 [6] 05 04 00 00 00 00** **can1 700 [7] 06 05 00 00 00 00 00** **can1 700 [8] 07 06 00 00 00 00 00 00** **can1 700 [8] 10 08 07 00 00 00 00 00** **can0 701 [3] 30 00 00** **can1 700 [3] 21 00 00** **[..] [..]**	**OK**

Explanation :

As soon as, the message is longer than 7 bytes, we could see the first frame with the data length.
Then we have, the flow control frame from the receiver (701). And the consecutive frame (with a loop from 21 to 2F)

Then we can launch canCheck with the this file and check if all is well received

canCheck -f isotp.txt -i