ACTIA *Automotive*

MEMO

Date :	02/02/17
From :	Actia
Subject :	Tests commands for Generating/ Reading/ Confronting programs

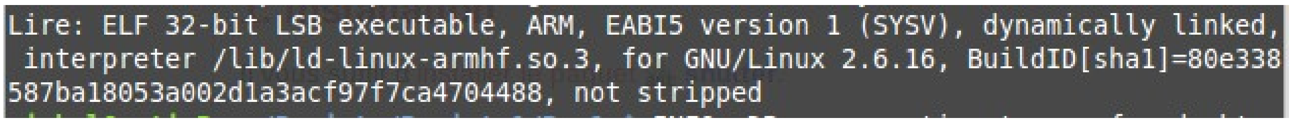
I) Foreword

There are 3 programs : Generating frame / Reading frame and one to confront these frame between them.
The generation and the reception is done through the *sockets can* thanks to *CAN bus.*

Before, executing these later we have to respect an order :

- *sourcing* (Generate a binary for ARM architecture / change the target) → Setting up SDK
  - Go to Wiki projet for imx6 , then click on *Setting up a generated SDK*
  - Run the source command : ***source ~/SDK/PCM_SDK/environment-setup-cortexa9hf-vfp-neon-oe-linux-gnueabi*** ***(in my case)***
- *compile* with **make** (to place oneself in the right directory)

Result expected after compilation :



```
Lire: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked,
 interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 2.6.16, BuildID[sha1]=80e338
587ba18053a002d1a3acf97f7ca4704488, not stripped
```

« Lire » : to depend the programs

II) Program 1 (Generating frame)

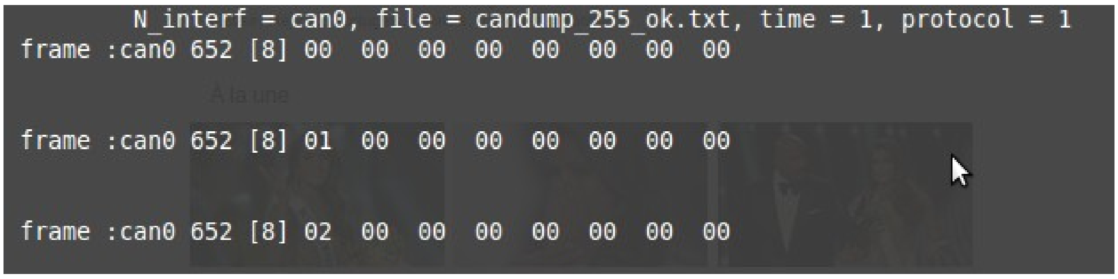
For this one, we have to respect instructions :

- *There are arguments* to pass to the function :
  - [-i] < CAN interface >
  - [-f] < file name > → file which allow to generate frame
  - [-t] < gap in microsecond >
  - [-z] < ptronbr > (default : CAN_RAW) → so not need to put this item
  - [-S] : if in the file, there are timestamps frame
  - [-h] : help

- Now we can *compile our program* :

  → Run the command (in the right directory) : **./Generer -i** can0 **-f** (file.txt) **-t** (.. in usec)
  → Add **-S** if the file contains *timestamp*

Result expected : (just one end)

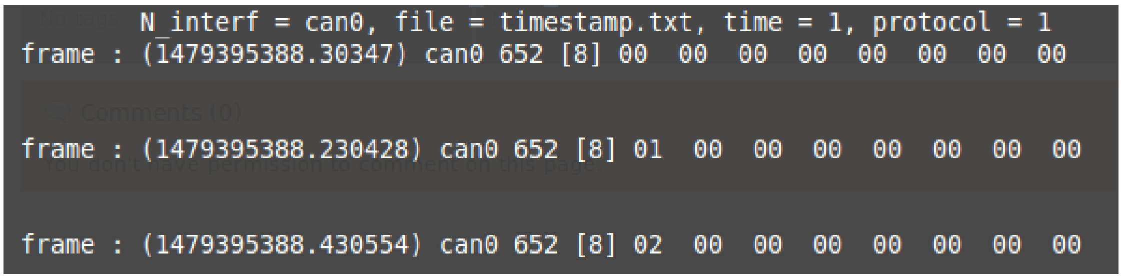


```
        N_interf = can0, file = candump_255_ok.txt, time = 1, protocol = 1
frame :can0 652 [8] 00  00  00  00  00  00  00  00

frame :can0 652 [8] 01  00  00  00  00  00  00  00

frame :can0 652 [8] 02  00  00  00  00  00  00  00
```

Result expected :With timestamp



```
        N_interf = can0, file = timestamp.txt, time = 1, protocol = 1
frame : (1479395388.30347) can0 652 [8] 00  00  00  00  00  00  00  00

frame : (1479395388.230428) can0 652 [8] 01  00  00  00  00  00  00  00

frame : (1479395388.430554) can0 652 [8] 02  00  00  00  00  00  00  00
```

II) Program 2 (Reading frame)

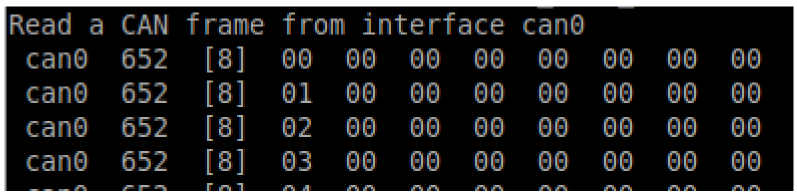
For this one, we have to respect instructions :

- *Options :*
  - [-i] < CAN interface >
  - [-w] < way > path where the user want to create the file
  - [-h] : help

- Now, we *can compile our program* :

  → Run the command (in the right directory) : **./Lire -i** can0 **-w** (/way …)

Result expected : (just one end)



```
Read a CAN frame from interface can0
 can0  652  [8]  00  00  00  00  00  00  00  00
 can0  652  [8]  01  00  00  00  00  00  00  00
 can0  652  [8]  02  00  00  00  00  00  00  00
 can0  652  [8]  03  00  00  00  00  00  00  00
```

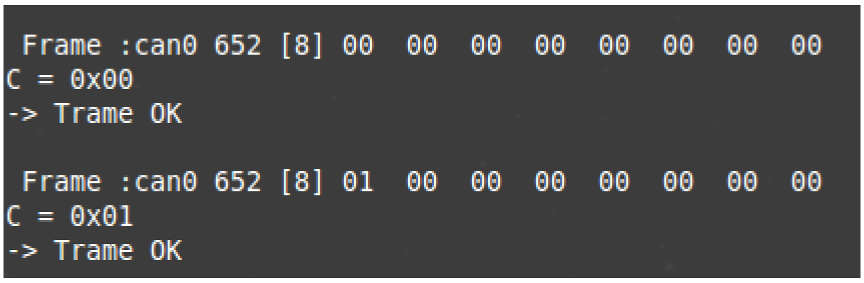
III) Program 3 (Confronting frame)

For this one, we have to respect instructions :

- *Options :*
  - [-f] < file name >
  - [-S] : with or without timestamp
  - [-h] : help

- Now, we *can compile our program* :
  → Run the command (in the right directory) : **./Com -f** file.txt
  → Add -S if the file contains timestamps

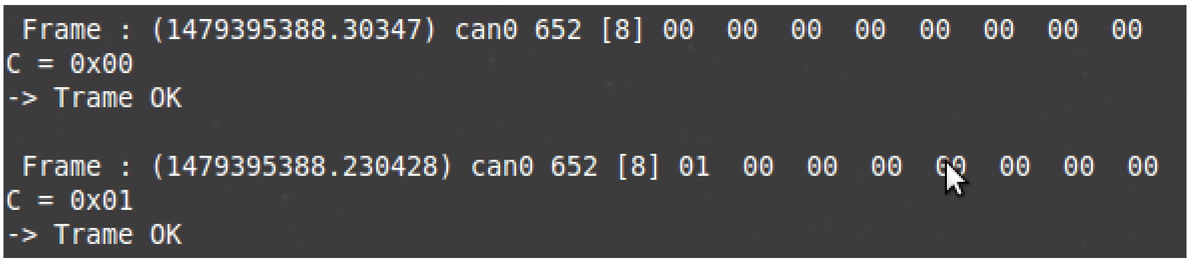
Result expected : (just one end)



```
 Frame :can0 652 [8] 00  00  00  00  00  00  00  00
C = 0x00
-> Trame OK

 Frame :can0 652 [8] 01  00  00  00  00  00  00  00
C = 0x01
-> Trame OK
```

Result expected :With timestamp



```
 Frame : (1479395388.30347) can0 652 [8] 00  00  00  00  00  00  00  00
C = 0x00
-> Trame OK

 Frame : (1479395388.230428) can0 652 [8] 01  00  00  00  00  00  00  00
C = 0x01
-> Trame OK
```