
# CAR HACKING

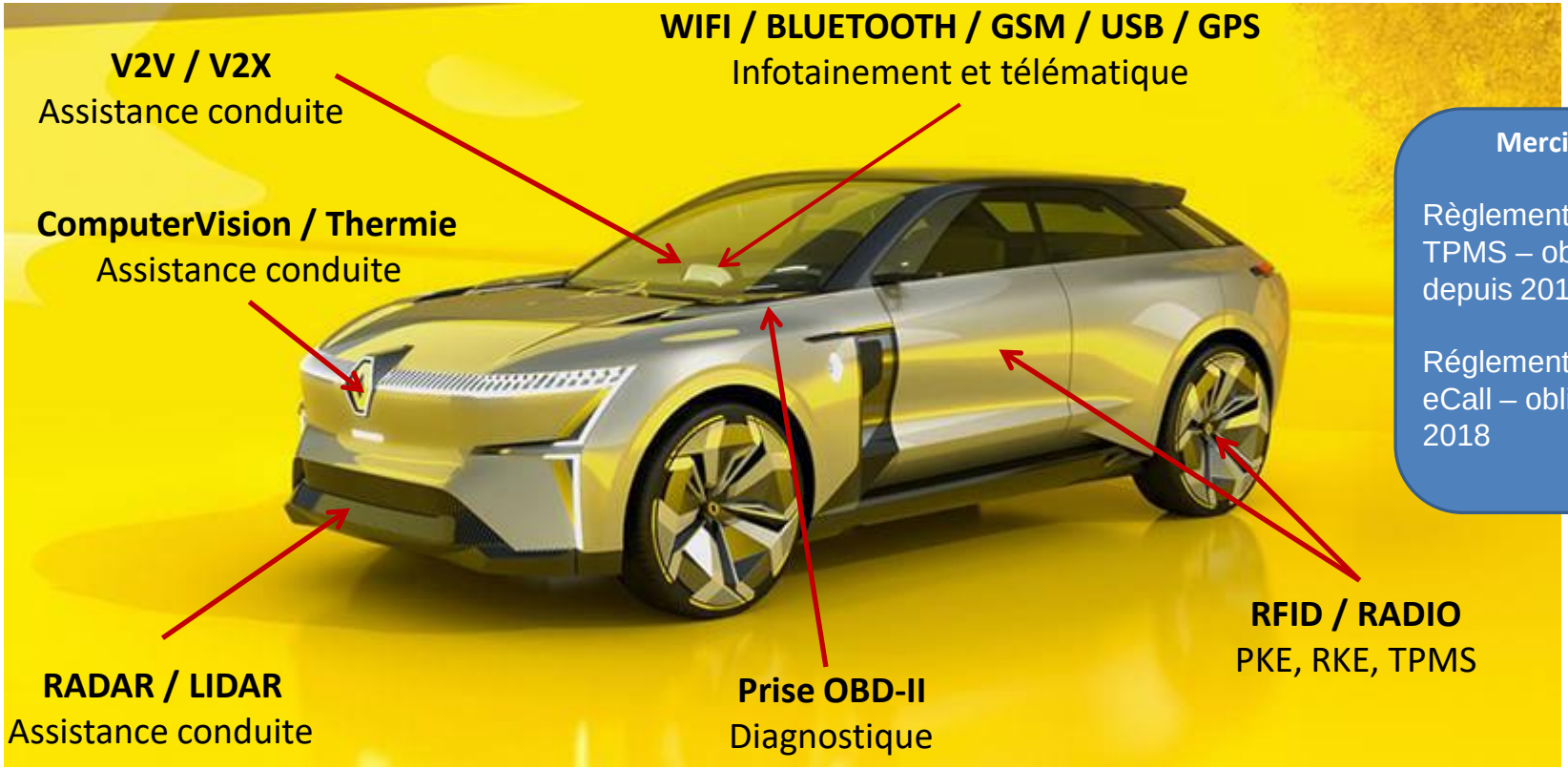
## Initiation au bus CAN

BARBHACK 2020

# Sommaire

- Une vaste surface d'attaque
- ECU 101
- Une voiture, différents réseaux
- Le connecteur OBD2
- Le bus CAN
- Trames CAN
- Boite à outils SW
- Boite à outils HW
  - Lab 1 : CAN Hello World
  - Lab 2 : filtre / cansniffer
  - Lab 3 : enregistrement / rejeu
- Protocole OBD-II & UDS
  - Lab 4 : requête OBD-II
  - Lab 5 : RoutineControl
- Protocole ISO-TP
  - Lab 6 : requête VIN
  - Lab 7 : SessionControl
- Documentation utile
  - Bonus : Hack a C-MhAckX

# Une vaste surface d'attaque




**Merci l'Europe !**

Règlement n°661/2009 **:** TPMS – obligatoire depuis 2014

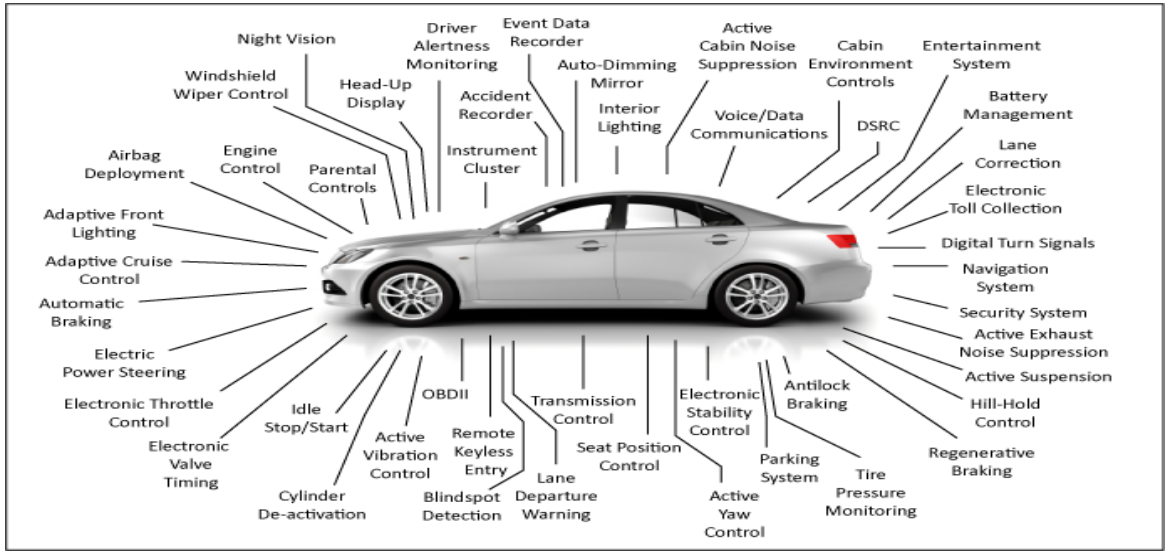
Réglement n°305/2013 : eCall – obligatoire depuis 2018

# ECU 101

**ECU** : **E**lectronic **C**ontrol **U**nit (calculateurs)

Un **ECU** se compose de **capteurs** et d'**actionneurs** et dialogue sur un ou plusieurs **réseaux** avec d'autres ECU

Les ECU sont souvent nommés via un trigramme.
Exemple : PCM (Power Control Module), BCM (Body Control Module), ABS, RCM (Restrain control Module)…




# Une voiture, différents réseaux

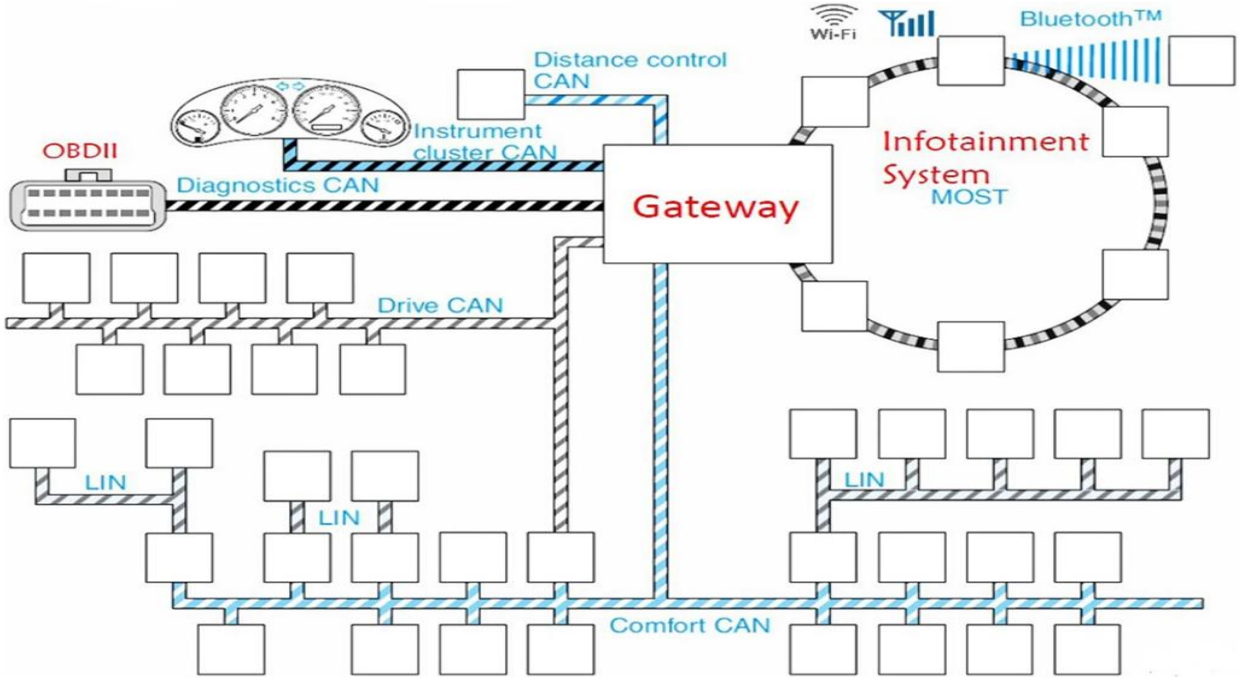
Les ECU sont interconnectés sur un ou plusieurs réseaux :

- **CAN**
  Controller Area Network

- **LIN**
  Local Interconnect Network

- **MOST**
  Media Oriented
  Systems Transport

- **FLEXRAY**




Il est courant de trouver 3 à 4 réseaux CAN différents dans un véhicule, filtrés ou non par une gateway

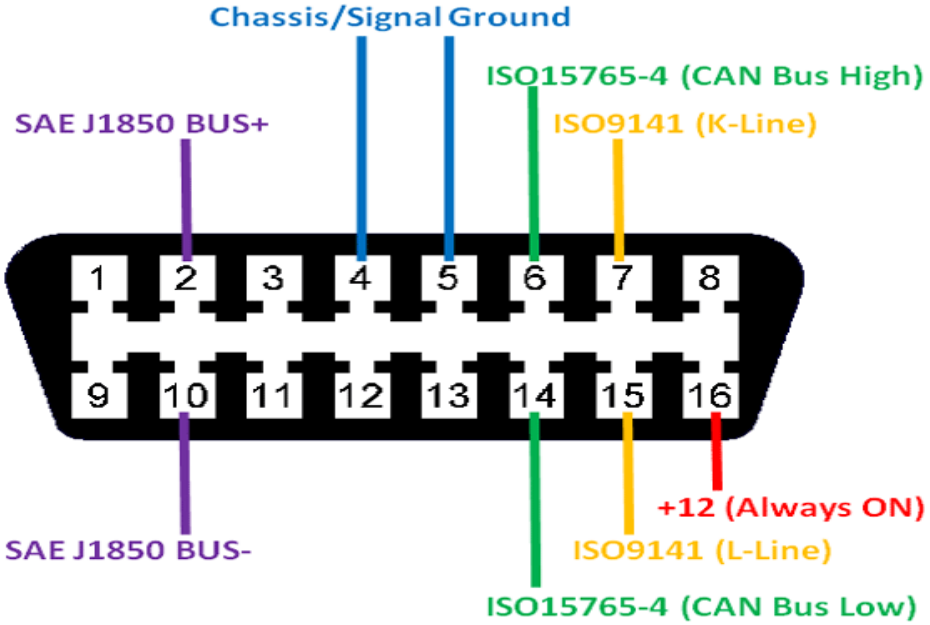
# Le connecteur OBD2

Le connecteur **OBD2** (On Board Diagnostic) est présent dans chaque véhicule, pour les besoins de diagnostic.

**Attention**, si les broches 4, 5, 6, 14, 12 et 16 sont toujours respectées, chaque constructeur fait un peu sa sauce.

Exemple : Ford
Pin 3-11 : CAN MS
Pin 12-13 : ICAN

**Emplacements habituels :**

# Le bus CAN

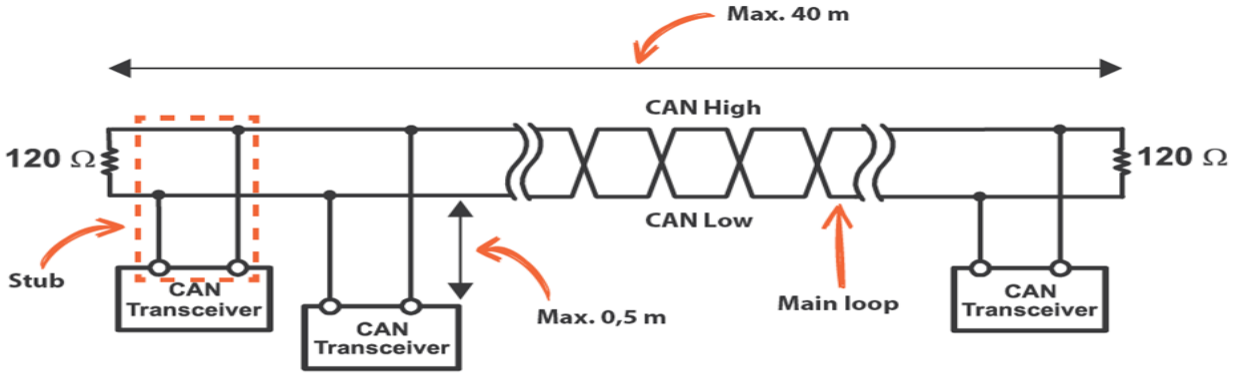
**Bus CAN** : liaison série asynchrone half-duplex, normalisé par la norme ISO 11898

Physiquement, c'est une paire torsadée avec un fil **CAN-High** et un fil **CAN-Low**, reliant différents ECU et terminé de part  et d'autres par des résistances de **120 Ohms**.

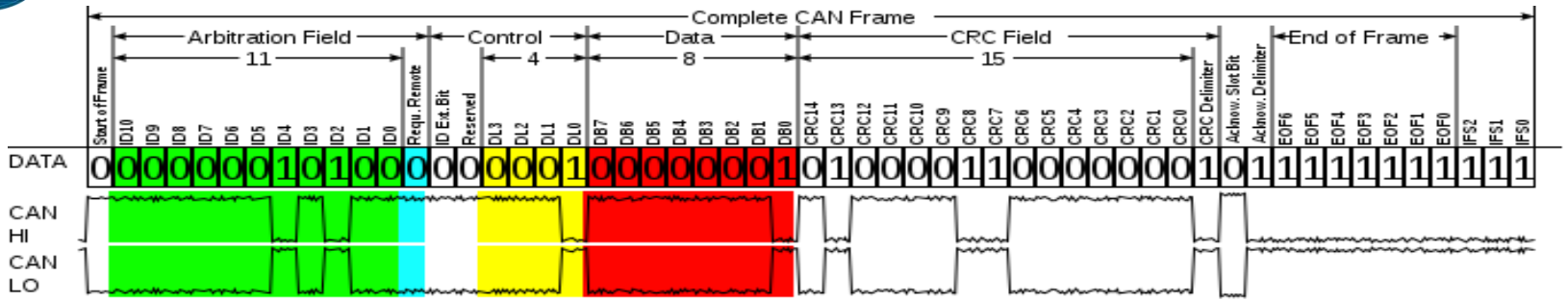
Le signal est une **tension différentiée** entre **CAN-H** et **CAN-L**.

Chaque message est **broadcasté** sur le bus, avec une gestion de priorité via **l'Arbitration ID.**

| Signal | CAN-H | CAN-L |
|---|---|---|
| 1 | 2.5V | 2.5V |
| 0 | 3.5V | 1.5V |



# Trames CAN




**4 types de trames**: data, error, remote et overload.

**Arbitration ID :** 11 bits (0x0 – 0x7FF) ou 29 bits (extended ID 0x0 – 0x1FFF FFFF)
**Data:** 1 à 8 octets

**Exemple d'une lecture de trames can, via candump :**

```
$ candump can0
can0  123   [8]  DE AD BE EF 01 02 03 04
```

**Arbitration ID** **Données**

# Boite à outils SW

**CAN-Utils**
https://github.com/linux-can/can-utils

Regroupe les commandes essentielles :
candump, cansend, cansniffer

**Socket CAN - slcan**
https://github.com/linux-can/can-utils

Permet de monter une interface CAN avec des équipements gérant le protocole LAWICELL

**Python-CAN**
https://python-can.readthedocs.io/en/master/

Librairie CAN pour Python

**SaavyCAN**
https://www.savvycan.com/

Logiciel C/QT avec pas mal de fonctions sympas pour l'analyse de trames CAN

# Boite à outils HW

| Matos | Caractéristiques | Fabricant | Prix |
|---|---|---|---|
| Nano-can | Arduino Nano + MCP251 | Mintynet (github / Twitter) | 15 € |
| USB Can Converter | STM32<br><br>Pas besoin de SLCAN | uCan devices (Tindie) | 24 € |
| Macchina M2 | 2 transceivers CAN<br>Compatible LIN / K-line<br><br>Base Arduino Leonardo | Macchina.cc | 90 € |
| ELM27 | ELM327<br>Capacité limitée | Made in China ! | 10 € |
| PiCAN 2<br>PiCAN 2 Duo | Shield raspberry<br>1 ou 2 entrées CAN | Skpang.uk | 50 - 65 € |

# Lab 1 – CAN Hello World

1- Dans une première fenêtre de terminal, taper :

```
$ candump vcan0
```

2- Dans une seconde entrer :

```
$ cansend vcan0 123#48454C4C4F2021
```

3- Relancer la commande candump en ajoutant l'option « -a » puis rejouer la commande cansend

4- Essayer d'autres valeurs d'arbitration ID ou de data via la commande cansend

5- La commande « cangen » permet d'envoyer une commande fixe ou incrémentielle à un intervalle donnée.

```
$ cangen  –I 123 –D 48454C4C4F2021 –g 1000 vcan0
```

# Lab 2 – Filtre / cansniffer

1- Pour les exercices suivants, nous aurons recours à ICSIM.
Lancer dans deux terminaux les commandes suivantes :

```
$ cd ~/CH-Workshop && ./controls vcan0

$ cd ~/ CH-Workshop && ./icsim vcan0
```

2- Lancer la commande « candump vcan0 » dans un nouveau terminal.
Il est possible de filter les ID affichés en ajoutant après le bus la référence de l'ID et un masque. Essayer :

```
$ candump vcan0,42A:7FF
```

3- La commande cansniffer affiche les données du bus triée par ID. Avec l'option –c les octets changeants sont mis en valeur.

```
$ cansniffer –c vcan0
```

4 - Avec cansniffer, trouver l'ID gérant l'affichage du voyant des clignotants.

# Lab 3 – Enregistrement / rejeu

1- candump peut logger l'ensemble des trames avec l'option « -l » (l minuscule)
Lancer la commande suivante, puis actionner les warnings (touche W). Après quelques secondes tuer la commande avec Ctrl+C

```
$ candump –l vcan0
```

2- Pour rejouer le log, on utilise la commande canplayer avec la commande « –I » (i majuscule) et la relation bus enregistré, bus de destination

```
$ canplayer –I <fichier> vcan0=vcan0
```

3- Que constate t'on une fois le rejeu effectué ? Essayer la même chose avec le déverrouillage des portes.

4- Si le rejeu est effectif, on peut rapidement chercher la trame d'intérêt en découpant le fichier de log via la commande « split »

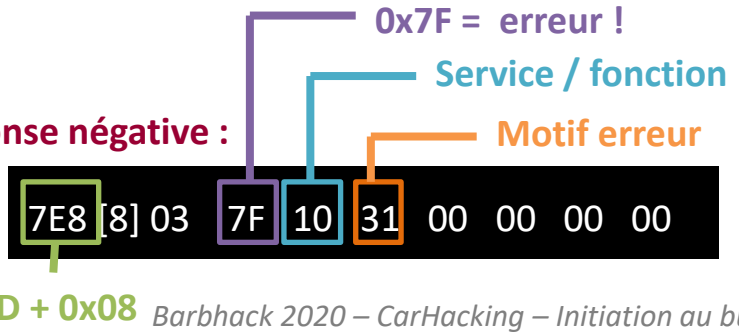
# Protocole OBD-II / UDS

| OBD-II : On Board Diagnostic – SAE J1979 | UDS : Universal Diagnostic Service – ISO 14229-1 |
|---|---|
| Permet la lecture/effacement des codes défauts et l'état du moteur.<br>**Fonctionnement** : Service / PID | Fonctions de diagnostic avancées, génériques et spécifiques du constructeur<br>**Fonctionnement** : Fonction / Sous-fontion |

**Trame type :**


**Réponse positive :**


**Réponse négative :**



# Lab 4 – Requête OBD-II

1- On souhaite connaitre la vitesse du véhicule, il suffit d'interroger le service 0x01 et le PID 0x0C

```
$ cansend vcan0 7E0#02010C
```

2- Si l'on falsifie la vitesse et que l'on lance ensuite la lecture de celle-ci, que constate t'on ?
Pour leurrer le compteur de vitesse, envoyer :

```
$ cangen –I 244 –D 0000003344 –g 5 vcan0
```

# Lab 5 – Routine Control

1- La fonction Routine Control (0x31) des outils de diagnostic est intéressante à étudier car elle contient des actions avancées/spécifiques mises en place par le constructeur automobile.

En étudiant les données transmises lors de l'activation de la valise de diagnostique, essayer de débloquer la routine cachée.

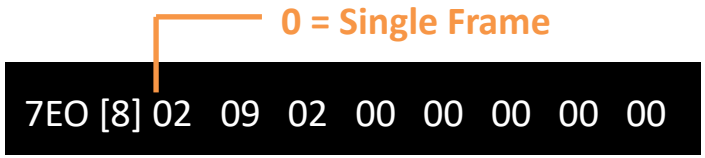
Conseils :

- La session doit être maintenue via la commande « Tester Present » (0x3E)

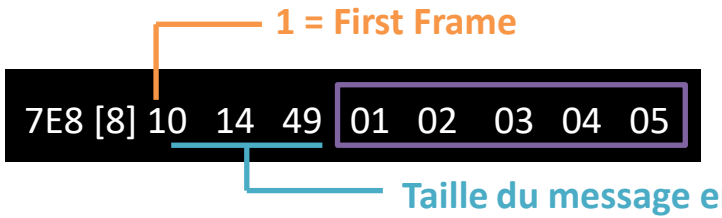
# Protocole ISO-TP

Le protocole **ISO-TP** (ISO 15765-2) permet d'envoyer des messages de plus de 8 octets, limité à **4096** octets

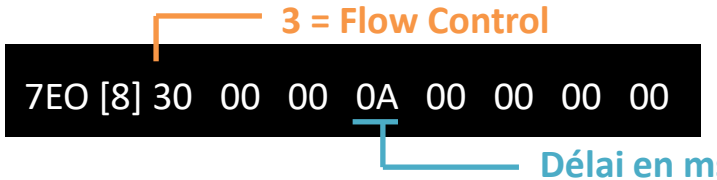
**Requête initiale :**



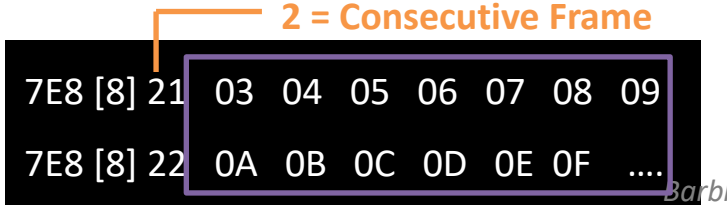

**Réponse de l'ECU :**




**Envoi du flow control :**




**Messages complémentaires :**



# Lab 6 – Requête VIN

1- Pour connaître le VIN du véhicule, la requête OBD-2 se fait sur le service 09 et le PID 02

```
$ cansend vcan0 7E0#020902
```

2- La réponse se faisant via le protocole ISO-TP, on peut normalement récupérer l'ensemble des données via la commande isotprecv. Attention, la commande isotprecv n'est pas fonctionnelle sur la VM, voir le point 3.

```
$ isotprecv –s7E8 –d7E0 vcan0
```

3- Pour palier au soucis de la commande isotprecv sur la VM entrer la commande suivante :

```
$ cansend vcan0 7E0#020902 && sleep 0.01 && cansend vcan0 7E0#30000A
```

# Lab 7 – Session Control

1- Le protocole UDS prévoit différent niveau de session dont :
0x01 – usage normal
0x02 – diagnostic
0x03 – programming mode

Observer l'influence du module de diagnostic sur le type de session.

2- La session 0x03 est protegée par une clé de session.

Une fois en mode 0x03, il est possible via la fonction 0x27 et la sous-fonction 0x01 de demander un seed.

Avec la fonction 0x27 et la sous-fonction 0x02 vous pouvez envoyer la réponse au challenge. Essayer de de trouver la bonne clé. Il est nécessaire de garder la session active avec la fonction « Tester present » (0x3E)

```
7E0 [3] 02   27   01
7E8 [5] 04   67   01   DE  AD
7E0  [3] 04  27   02   ??   ??
```

# Documentation utile

**Car Hacker Handbook** : http://opengarages.org/handbook/

**Illmatics – adventure in car hacking** : http://illmatics.com/carhacking.html

# Bonus : Hack a C-MhAckX

En SSH, vous pouvez vous connecter aux différents bus CAN du C-MhAckX via les adresses suivantes :

**CAN-HS** (motorisation) : 192.168.12.200

**CAN-MS** (habitacle) : 192.168.12.201

**CAN-ICAN** (multimédia) : 192.168.12.202

Login : barbhack
Password : barbhack