Preparation:

Q.1:

|  |
| --- |
| extern ARM\_DRIVER\_CAN Driver\_CAN1;  Driver\_CAN1.Initialize(NULL,NULL);  Driver\_CAN1.PowerControl(ARM\_POWER\_FULL);  Driver\_CAN1.SetMode(ARM\_CAN\_MODE\_INITIALIZATION);  Driver\_CAN1.SetBitrate(  ARM\_CAN\_BITRATE\_NOMINAL, 125000,  ARM\_CAN\_BIT\_PROP\_SEG(5U) |  ARM\_CAN\_BIT\_PHASE\_SEG1(1U) |  ARM\_CAN\_BIT\_PHASE\_SEG2(1U) |  ARM\_CAN\_BIT\_SJW(1U));  Driver\_CAN1.ObjectConfigure(0,ARM\_CAN\_OBJ\_RX);  Driver\_CAN1.SetMode(ARM\_CAN\_MODE\_NORMAL); |

Q.2:

|  |
| --- |
| extern ARM\_DRIVER\_CAN Driver\_CAN2;  Driver\_CAN2.Initialize(NULL,NULL);  Driver\_CAN2.PowerControl(ARM\_POWER\_FULL);  Driver\_CAN2.SetMode(ARM\_CAN\_MODE\_INITIALIZATION);  Driver\_CAN2.SetBitrate(  ARM\_CAN\_BITRATE\_NOMINAL, 125000,  ARM\_CAN\_BIT\_PROP\_SEG(5U) |  ARM\_CAN\_BIT\_PHASE\_SEG1(1U) |  ARM\_CAN\_BIT\_PHASE\_SEG2(1U) |  ARM\_CAN\_BIT\_SJW(1U));  Driver\_CAN2.ObjectConfigure(1,ARM\_CAN\_OBJ\_TX);  Driver\_CAN2.SetMode(ARM\_CAN\_MODE\_NORMAL); |

Q.3: Dans cet cas on peut utilizer une table avec des filtres pour des identifiants specifiques.

|  |
| --- |
| Driver\_CAN1.ObjectSetFilter( 0, ARM\_CAN\_FILTER\_ID\_EXACT\_ADD , ARM\_CAN\_STANDARD\_ID(0x161), 0) ;  Driver\_CAN1.ObjectSetFilter( 0, ARM\_CAN\_FILTER\_ID\_EXACT\_ADD , ARM\_CAN\_STANDARD\_ID(0x0b6), 0) ; |

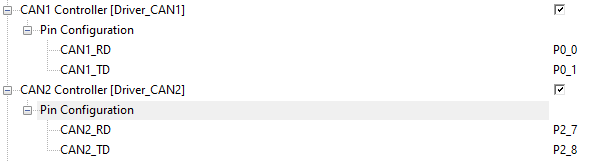
Q.4:

|  |
| --- |
| ARM\_CAN\_MSG\_INFO tx\_msg\_info;  tx\_msg\_info.id = ARM\_CAN\_STANDARD\_ID (0x0b6);  tx\_msg\_info.rtr = 0; // 0 = trame DATA  data\_buf [0] = 0xFA; // data à envoyer à placer dans un tableau de char  Driver\_CAN1.MessageSend(1, &tx\_msg\_info, data\_buf, 1); |

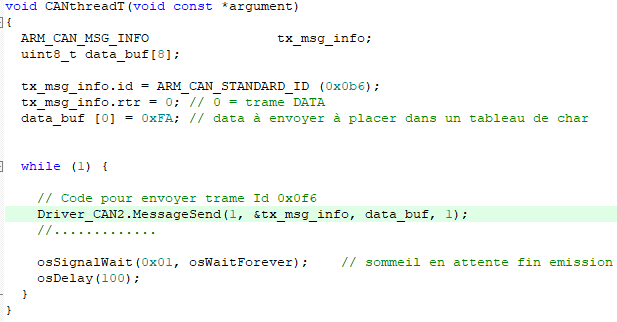
<https://drive.google.com/drive/folders/1iqufj9sVkTgJt7lpXY9O_f5XWARdJt3G>

<https://drive.google.com/drive/folders/1iqufj9sVkTgJt7lpXY9O_f5XWARdJt3G>

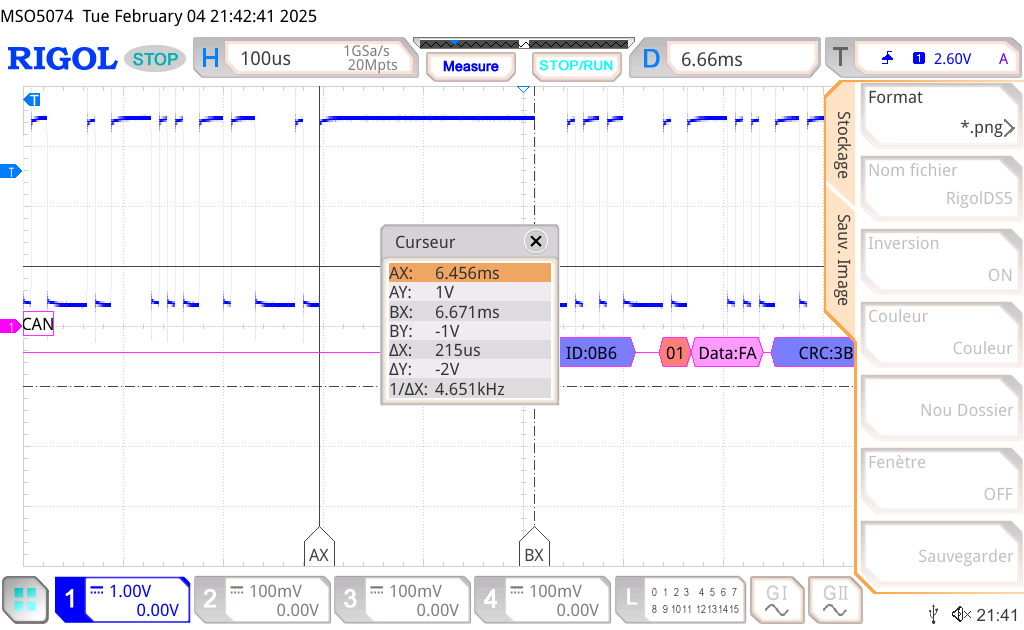
Q.5:



Q.6:

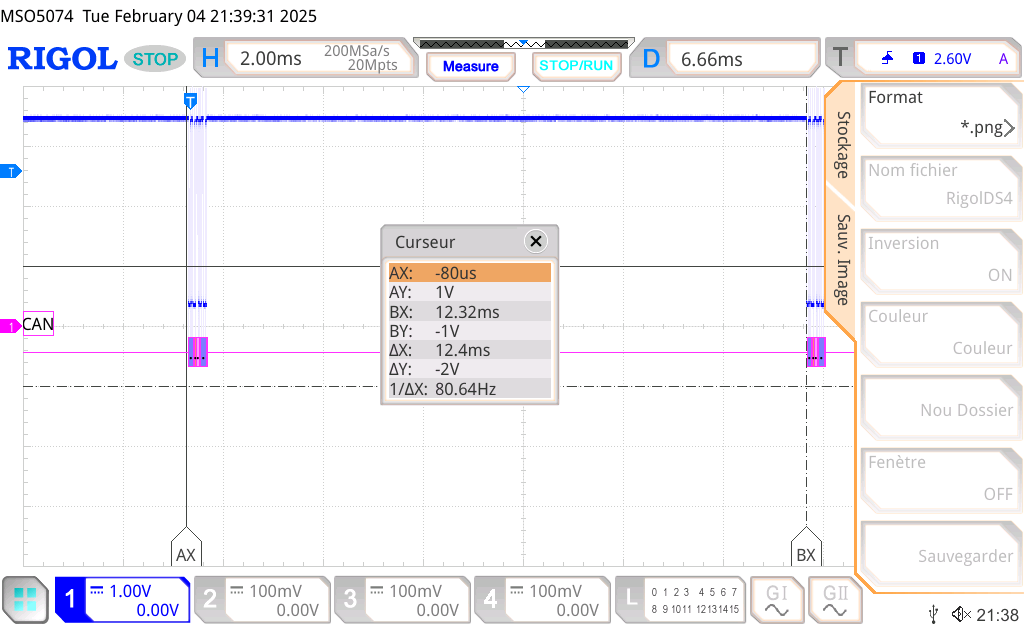


Q.7:

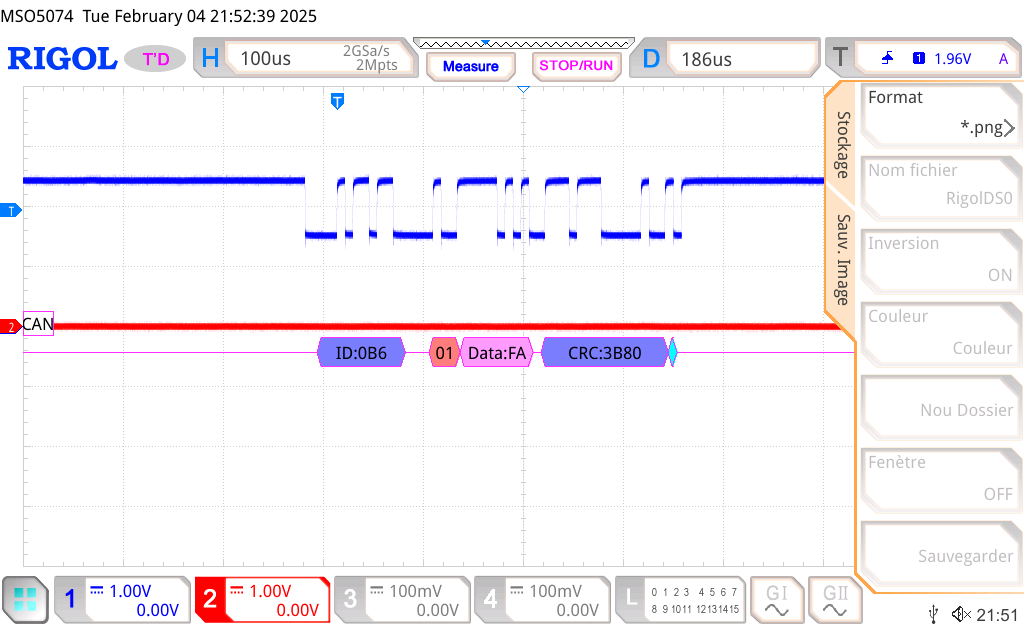


100ms = 10 Hz,

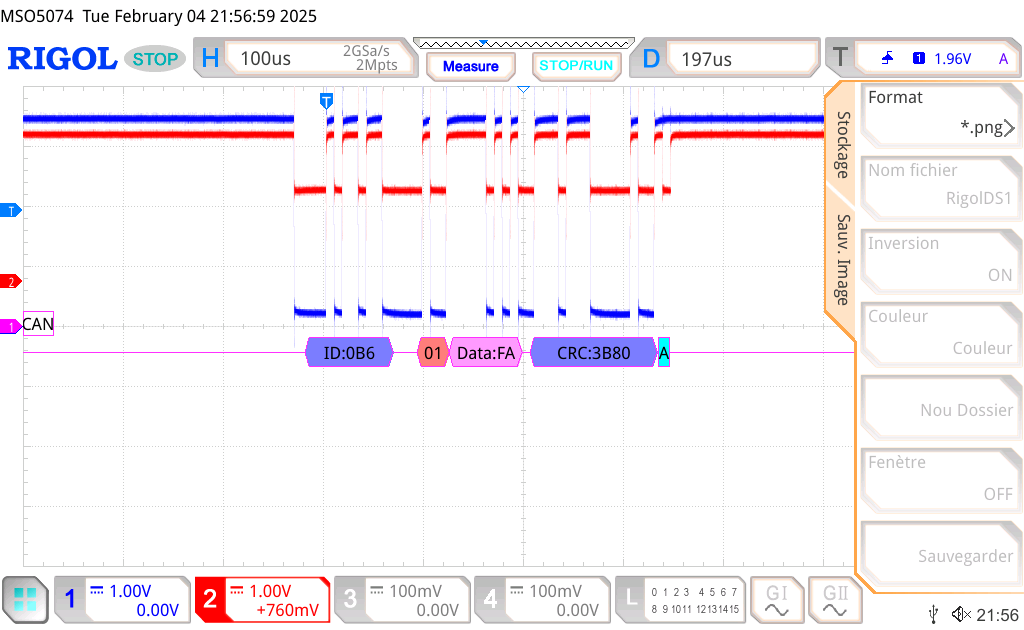
Ici, la trame est envoyé en permanence car il n’y à pas un ‘ACK’ pour indiquer la reception.

Q.8:

Mainenant, la trame est envoyé periodiquement.



Q.9:



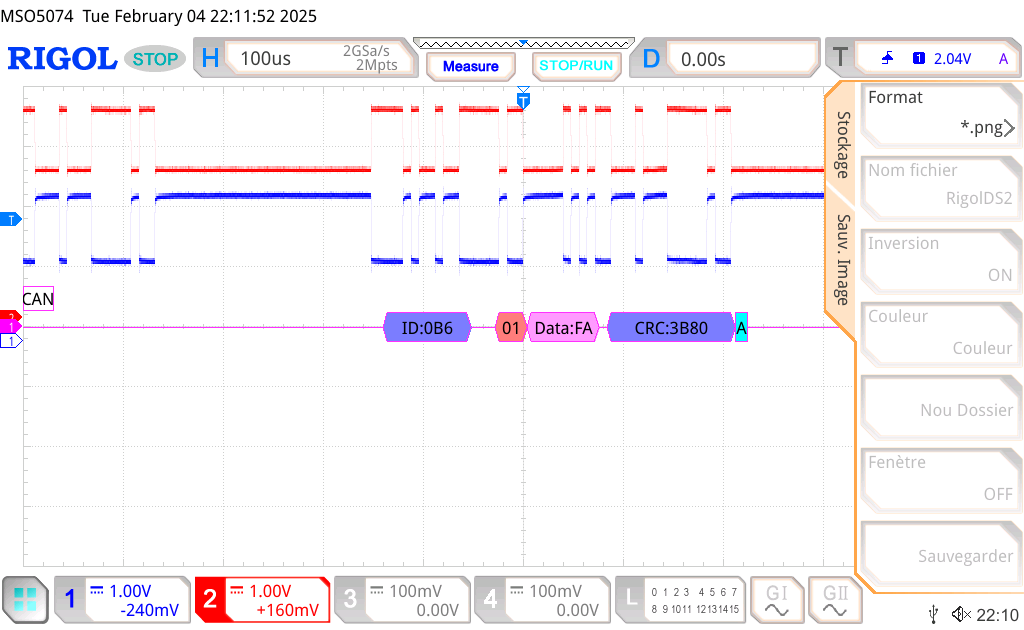
Ici CH1 = P2.8 et CH2 = Tx2

On peut voir le ‘ACK’ depuis le CAN1 qui vient d’ecraser la valeur récessife de CAN2.

(dernier bit a 1 sur CH1, mais 0 sur CH2)

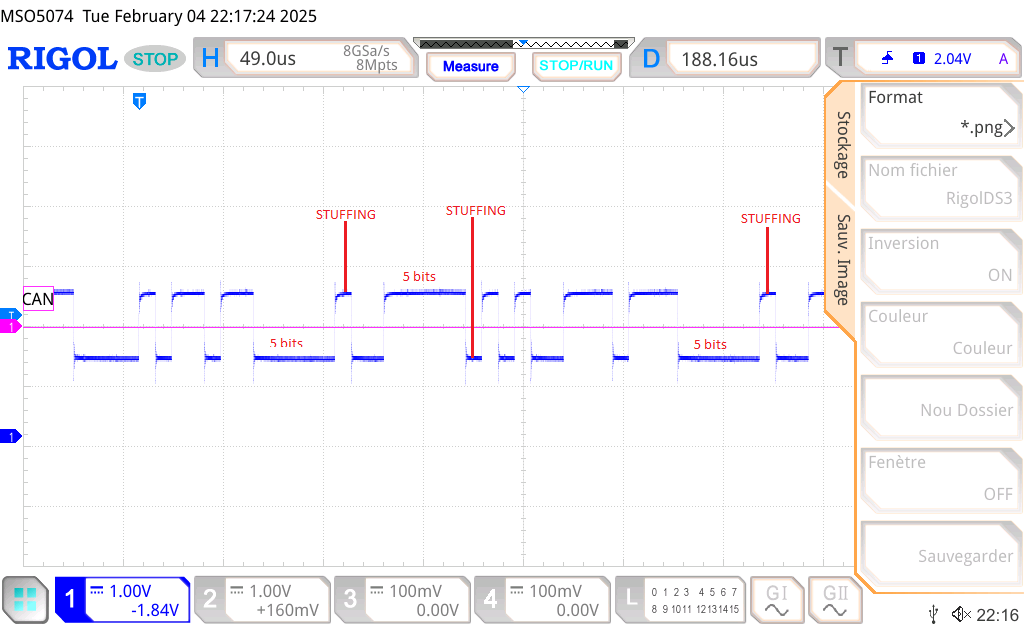
Q.10:

CH1 = CAN\_L



CH2 = CAN\_H

Q.11:



Q.12:

Q.1:

|  |
| --- |
| // tache envoi toutes les secondes  void CANthreadT(void const \*argument)  {  ARM\_CAN\_MSG\_INFO tx\_msg\_info;  uint8\_t data\_buf[8];  tx\_msg\_info.id = ARM\_CAN\_STANDARD\_ID (0x0b6);  tx\_msg\_info.rtr = 0; // 0 = trame DATA  data\_buf [0] = 0xFA; // data à envoyer à placer dans un tableau de char    while (1) {  // Code pour envoyer trame Id 0x0f6  Driver\_CAN2.MessageSend(1, &tx\_msg\_info, data\_buf, 1);  //.............  // osSignalWait(0x01, osWaitForever); // sommeil en attente fin emission  osDelay(100);  }  } |

Q.1:

Q.1:

Q.1: