**Conception général :**

**Matériels utilisées :**

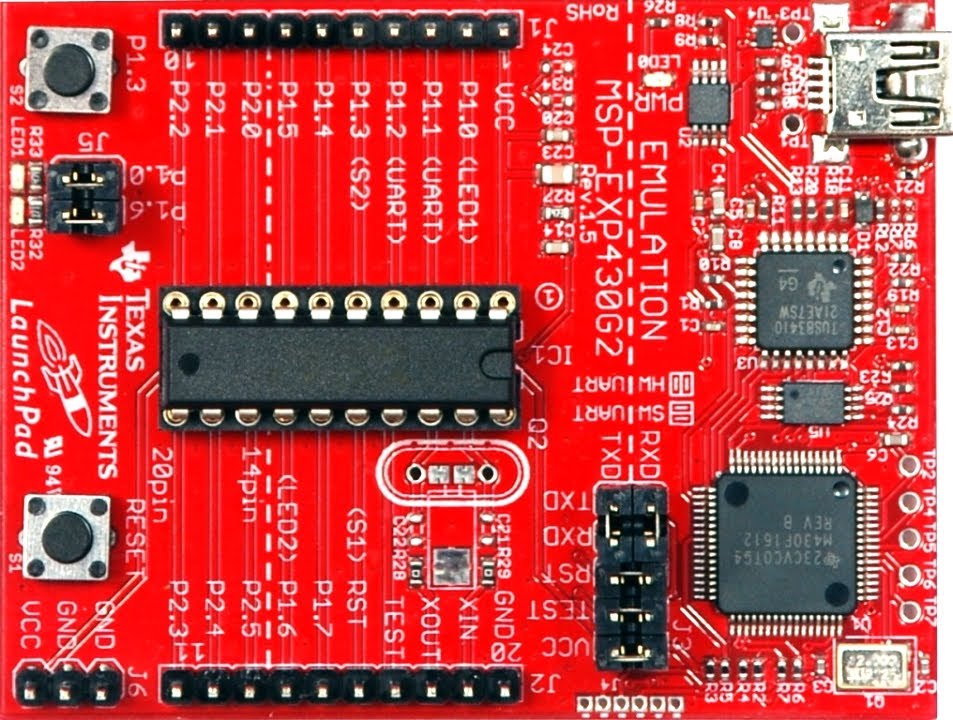


Figure 1 : Lauchpad



Figure 2 : MSP430G2231

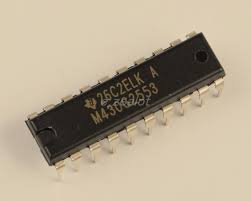


Figure 3 : MSP430G2553



Figure 4 : Servomoteur



Figure : module Bluetooth (installé sur le master)



Figure : L'infrarouge

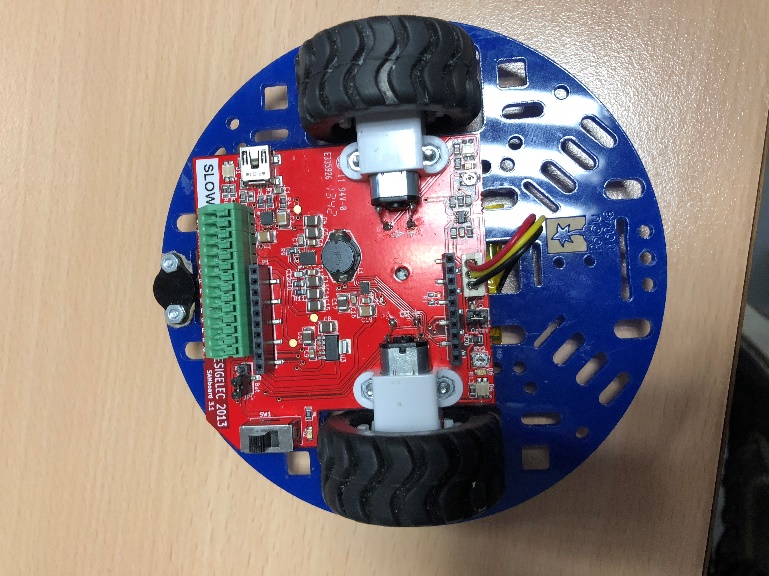


Figure : SAMbot

**Prendre une photo pour la construction finale**

Explication générale du projet :

Globalement, ce projet nous demande de réaliser un SAMbot (robot rouge) en deux modes :

Mode manuel :

Pour le mode manuel, il faut utiliser un module Bluetooth qui permet de faire la communication avec une téléphone portable ou ordinateur par UART. Et dirige le robot avancer, reculer, tourner à gauche ou droite. Il faut principalement programmer sur le microprocesseur MSP430G2553.

Mode autonome :

Pour le mode autonome, il faut faire la communication entre MSP430G2253(Master) et MSP430G2231(Slave) par SPI. Le slave commander le servomoteur de tourner un certain angle. Et le master envoie les différents caractères vers slave pour tourner à différent angle. Une fois cela est réalisé. Il faut aussi installer l’infrarouge sur le servomoteur pour qu’il peut détecter les objets en différents sens (devant, gauche, droite). Et l’infrarouge est connecté avec le master. Une fois, l’infrarouge détecte un objet devant, il va détecteur à une autre direction pour savoir s’il y a des objets dans les autres directions. Une fois c’est fait, le master va diriger le robot d’avancer sur le chemin où qu’il n’y a pas d’objet devant. Installation des deux modules est montré par l’image suivante.

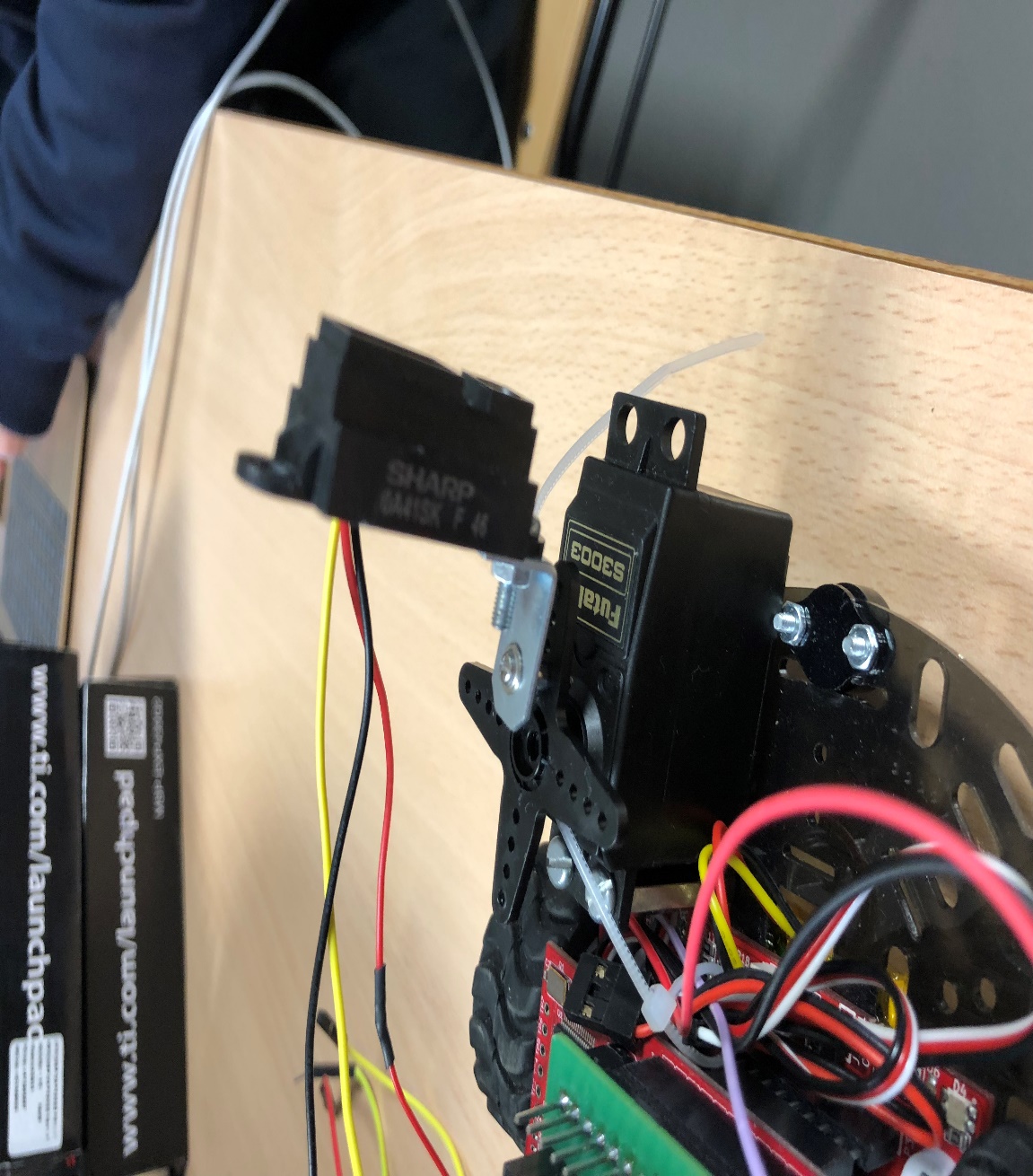


Figure : installation d'infrarouge et servomoteur

Les parties les plus important sont transporter les informations entre deux cartes de microprocesseurs de types différents par SPI et aussi faire communication entre une carte de microprocesseur et le Bluetooth par UART.

**Conception spécifiée :**

Pour réaliser ce projet, nous allons séparer en 2 grandes parties (master et slave) :

Et en-dessous nous avons séparé en 5 sous parties :

1) Communication entre Bluetooth et MSP430G2553

2) L’infrarouge donne l’information à MSP430G2553(Master)

3) Communication entre MSP430G2553 et MSP430G223(SPI)

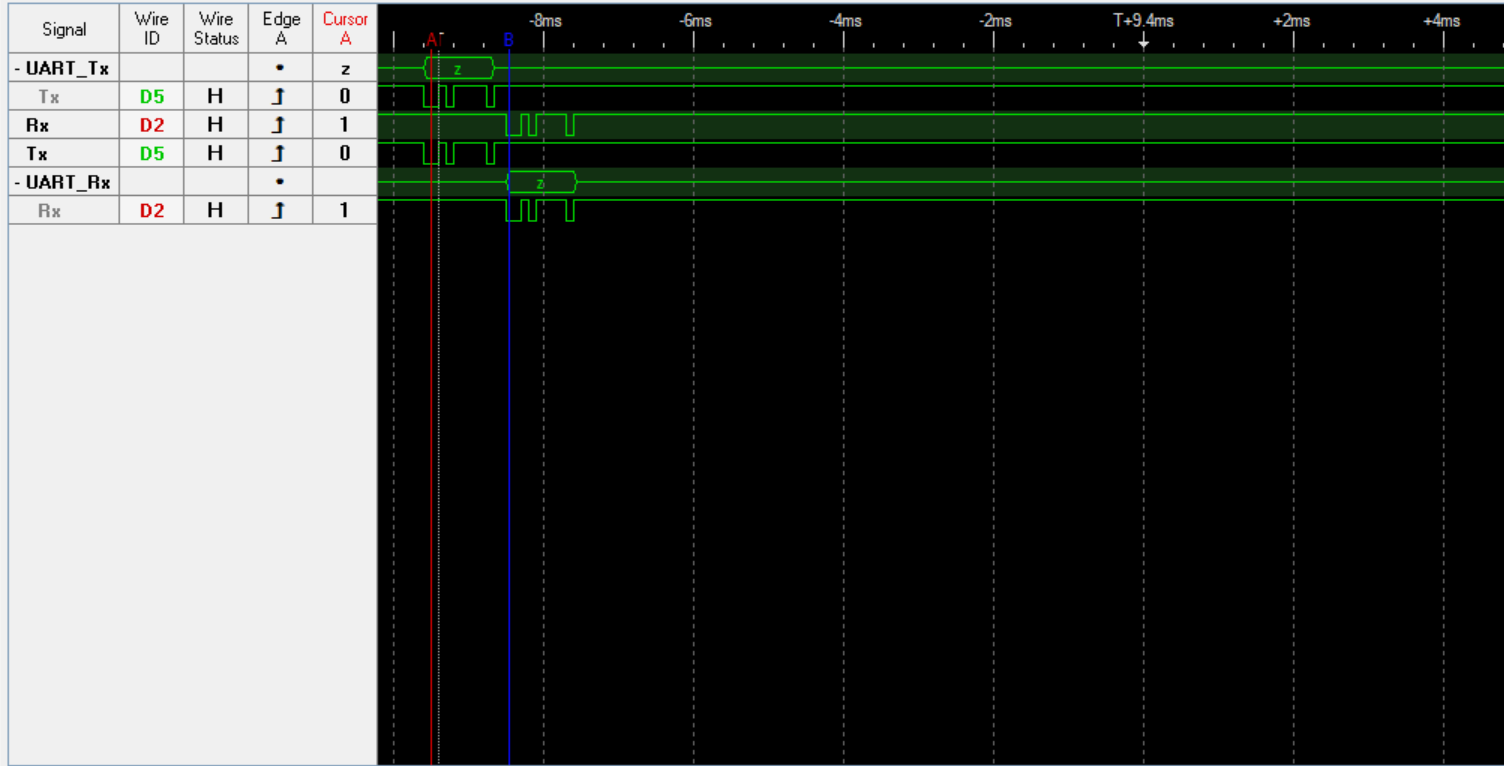
4) MSP430G2231 dirige le mouvement de servomoteur.

5) MSP430G2553 dirige le mouvement du robot.

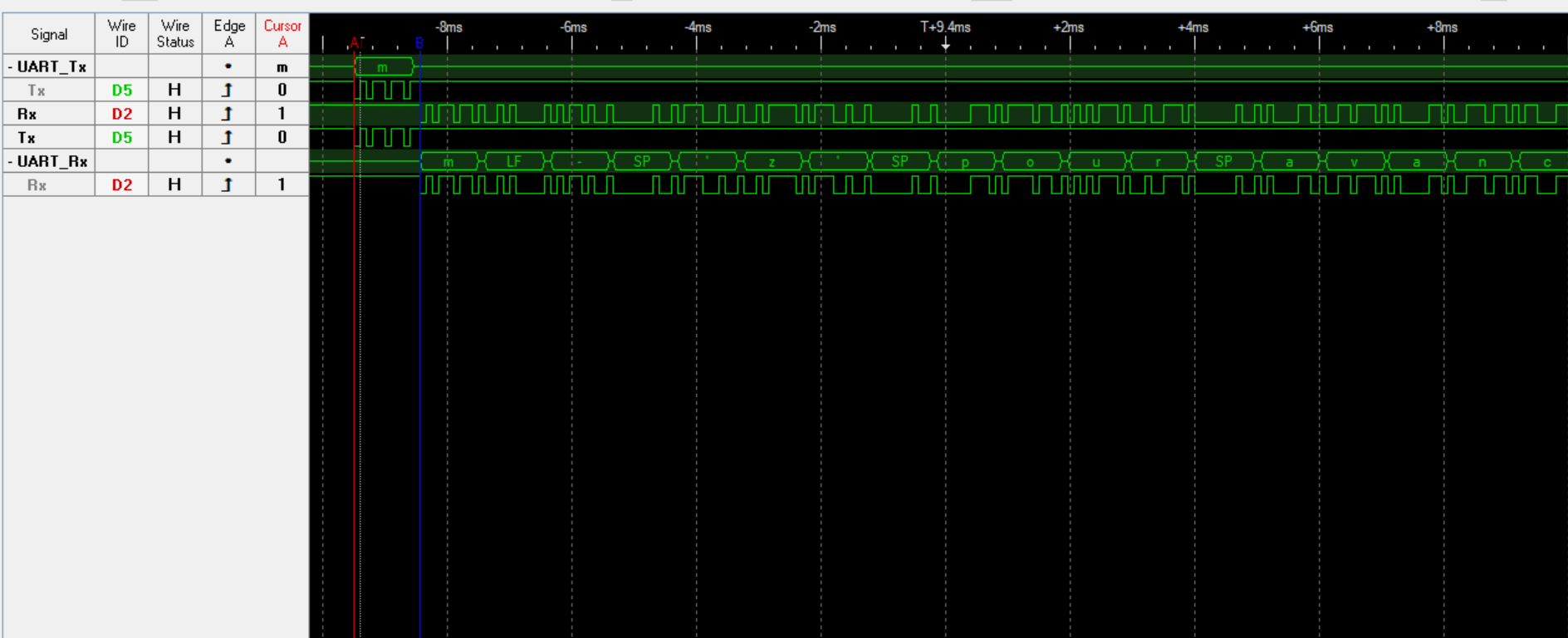
# Test unitaire :

## Communication entre Bluetooth et MSP430G2553 :

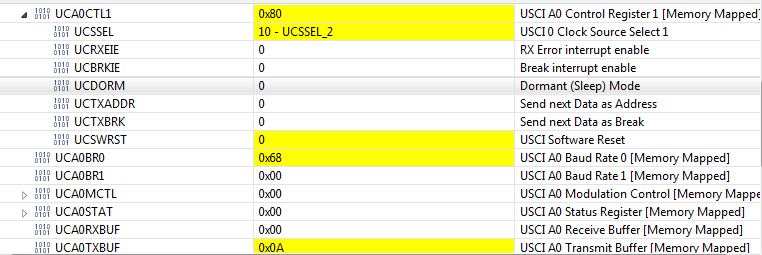
|  |  |
| --- | --- |
| Nom de la fonction : | Txdata |
| Méthode de test | Envoyer lettre ‘z’ au UART |
| Résultat attendu | Lettre ‘z’ est affiché sur PuTTY |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |



|  |  |
| --- | --- |
| Nom de la fonction : | Print\_str |
| Méthode de test | Envoyer une chaine caractère au UART |
| Résultat attendu | Une chaine est affichée sur PuTTY |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |



|  |  |
| --- | --- |
| Nom de la fonction : | InitUART |
| Méthode de test | Voir les registres est bien identique que ceux qu’on veut (UCA0CTL1 et UCA0BR0 et UCA0BR1) |
| Résultat attendu | Les registres sont bien configurés d’après la spécification (UCA0CTL1 voir sur la spécification), (UCA0BR0=104 et UCA0BR1=0) |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |



Tips : 0x68 = 104

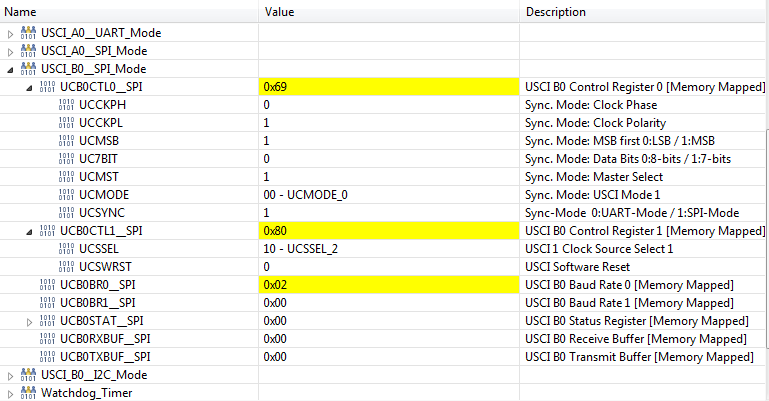
2) L’infrarouge donne l’information à MSP430G2553(Master)

## 3) Communication entre MSP430G2553 et MSP430G223(SPI)

De même comme la spécification détaillée, nous avons séparé le test unitaire en deux parties :

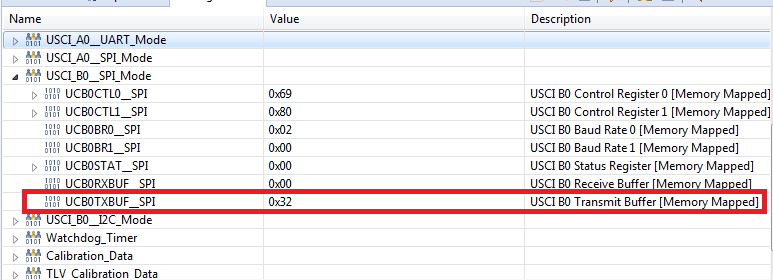
### 3-1) Master :

|  |  |
| --- | --- |
| Nom de la fonction : | SPI\_Init |
| Méthode de test | Voir les registres est bien identique que ceux qu’on veut (UCB0CTL1, UCB0CTL0 et UCA0BR0 et UCA0BR1) |
| Résultat attendu | Les registres sont bien configurés d’après la spécification (UCA0CTL1, UCB0CTL0 voir sur la spécification), (UCA0BR0=0x02 et UCA0BR1=0) |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |



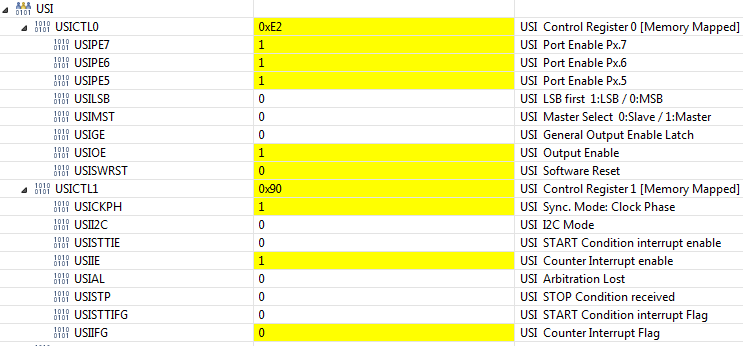
|  |  |
| --- | --- |
| Nom de la fonction : | SPI\_Tx |
| Méthode de test | Envoyer 0x32 à slave |
| Résultat attendu | 0x32 est reçu par le slave |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |





### 3-2) Slave :

|  |  |
| --- | --- |
| Nom de la fonction : | Init\_SPI |
| Méthode de test | Voir les registres est bien identique que ceux qu’on veut (USCICTL0, USCICTL1 et UCA0BR0 et UCA0BR1) |
| Résultat attendu | Les registres sont bien configurés d’après la spécification (USCICTL0, USCICTL1 voir sur la spécification) |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |

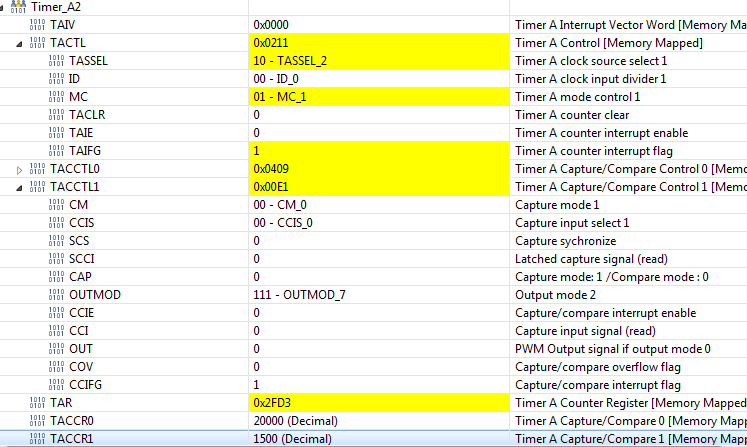


|  |  |
| --- | --- |
| Nom de la fonction : | RX\_Data |
| Méthode de test | Recevoir 0x32 par le slave |
| Résultat attendu | 0x32 est reçu par le slave |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |

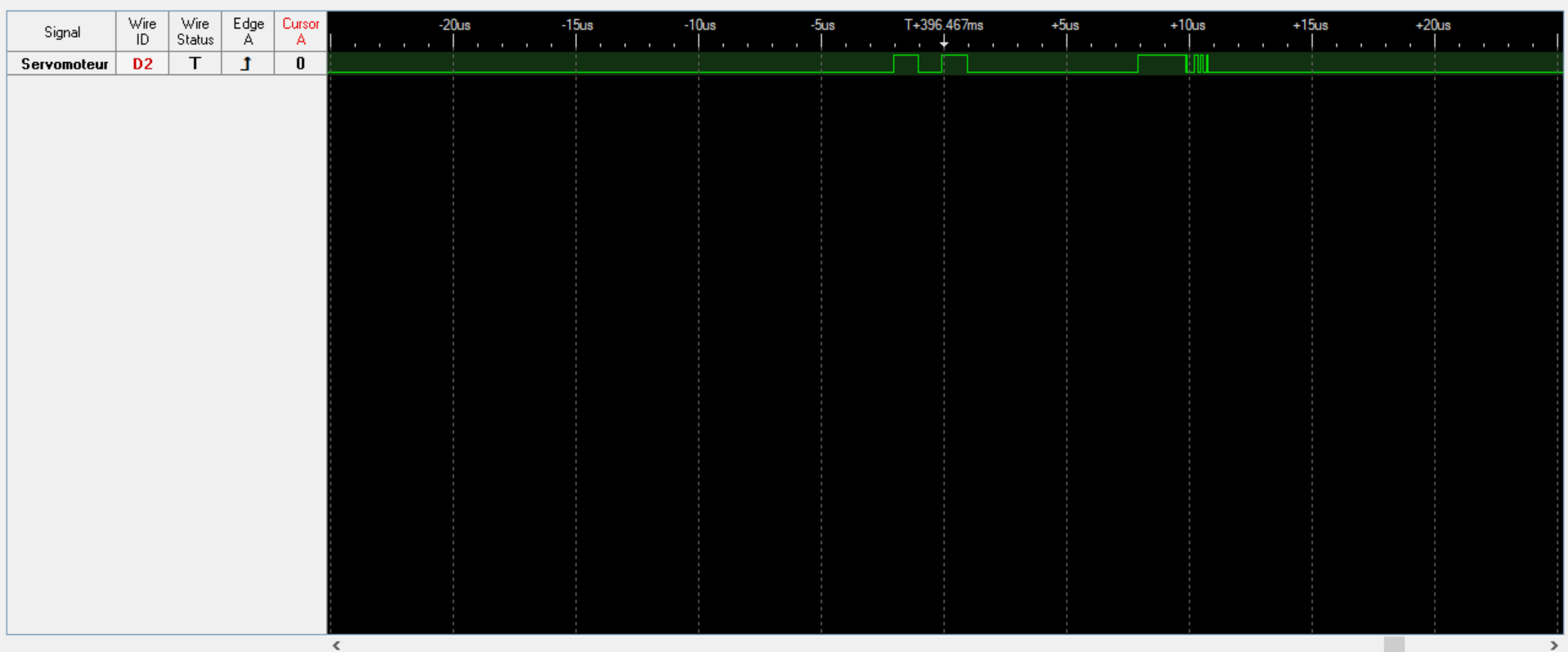


### 4) MSP430G2231 dirige le mouvement de servomoteur.

|  |  |
| --- | --- |
| Nom de la fonction : | Motor\_PWM\_Init |
| Méthode de test | Voir les registres est bien identique que ceux qu’on veut (TACTL, TACTL1 et TACCR0 et TACCR1) |
| Résultat attendu | Les registres sont bien configurés d’après la spécification (TACTL, TACTL1 voir sur la spécification. ( TACCR0=20000(période)),( TACCR1=1500(rapport cyclique)) |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |



|  |  |
| --- | --- |
| Nom de la fonction : | Motor\_Set\_Deg |
| Méthode de test | Vérifier si le nombre retourner est bien le rapport cyclique qu’on veut |
| Résultat attendu | 1. 4000 pour 0° 2. 10000 pour90° 3. 20000 pour 180° |
| Résultat obtenu | Voir sur le photo |
| Validé ? | Oui |



### 5) MSP430G2553 dirige le mouvement du robot.