

La société PELLENC souhaite ajouter une nouvelle fonctionnalité à son attacheur afin d'optimiser la maintenance de ses appareils. Cette nouvelle fonction doit permettre d'enregistrer le nombre d'attaches posées pour que le viticulteur puisse récupérer sur son ordinateur un historique de ces enregistrements.

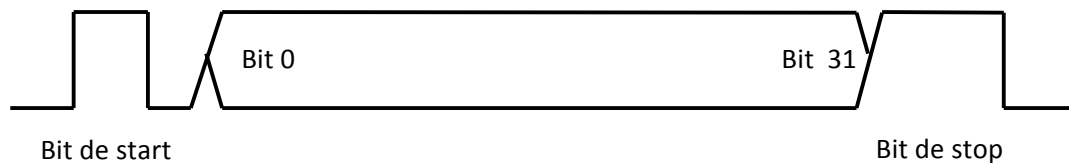
La transmission de l'information depuis le microcontrôleur jusqu'à l'ordinateur passe par un module de conversion RS232-USB. En effet le microcontrôleur communique avec le protocole RS232.

Vous êtes ingénieur dans la société PELLENC, il vous est demandé de réaliser le programme permettant de répondre à cette demande. Voici un exemple de transmission série que vous allez recréer et compléter.



1. Cahier des charges partiel de la transmission de donnée :

On désire effectuer la transmission d'informations en générant le signal représenté ci-dessous :



Chaque trame débute par un bit de START et comporte 32 bits de données. La trame se termine par un bit de STOP.

1.1. Contenu d'une trame d'émission :

Une trame (formée de 32 bits) est composée de 8 quartets (4 bits) dans l'ordre d'émission suivant :

entête							checksum	
D0 .. D3	D4 .. D7	D8 .. D11	D12 ..D15	D16 ..D19	D20 ..D23	D24 ..D27	D28 ..D31	
LSB MSB	LSB MSB	LSB MSB	LSB MSB	LSB MSB	LSB MSB	LSB MSB	LSB MSB	LSB MSB

D0=D1=D3=0 et D2=1 Entête

- D4 .. D7 : chiffre des unités de la donnée
- D8 .. D11 : chiffre des dizaines de la donnée
- D12 .. D15 : chiffre des centaines de la donnée
- D16 .. D19 : chiffre des milliers de la donnée
- D20.. D23 : chiffre des dizaines de milliers de la donnée
- D24 .. D27 : chiffre des centaines de milliers de la donnée

D28 ...D31: checksum = somme modulo 16 des 7 premiers quartets (permet de vérifier la trame)

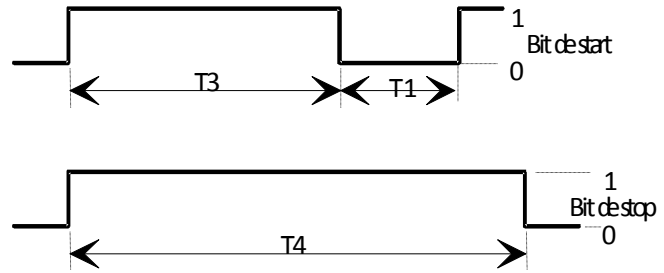
Exemple :

0x04	0x05	0x06	0x05	0x06	0x05	0x06	0x09
------	------	------	------	------	------	------	------

- Le « zéro » est transmis sous la forme d'un état bas de 220 µs,
- Le « un » est transmis sous la forme d'un état haut de 220 µs.

1.2. Bit de start et bit de stop

ETAT DU BIT	Durée minimale en µs	Durée Typique en µs	Durée maximale en µs
Bit de start / niveau haut	380	500 T3	590
Bit de start / niveau bas	75	220 T1	500
Bit de stop / niveau haut	675	840 T4	900



1.3. Comportement attendu :

- Le signal sera généré sur la broche RA4 du µC (microcontrôleur) à chaque action sur le bouton poussoir BP.
- Temps minimum entre 2 trames 10 ms.

Le cahier des charges peut prêter à diverses interprétations et toutes améliorations ou modifications sont possibles.

Q1. Indiquer le nombre d'attaches réalisées indiqué par l'exemple ci-dessus.

2. Travail demandé :

2.1. Programmation :

Démarrer FLOWCODE :

Ouvrir le programme **trame_ELEVES_V2.fcf**.

2.1.1. Etude du programme principal :

Le programme principal, respecte ce pseudo code :

Début

```

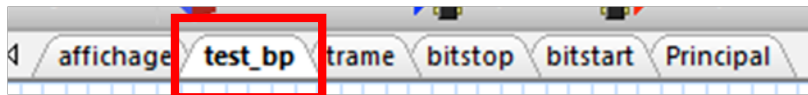
Faire toujours
  test_BP           // appel d'un sous-programme de test de l'état du Bouton Poussoir
  trame             // appel d'un sous-programme d'envoi d'une trame
  temporisation_10ms
  affichage        // appel d'un sous-programme d'affichage
Fin Faire
Fin
    
```

Q2. En observant l'algorithme ci-contre (voir aussi Flowcode), indiquer comment on s'assure de répéter le programme indéfiniment.

Q3. Que se passe-t-il lorsque le programme principal arrive sur le bloc « Appel d'un test BP » ou « appel d'une trame » ?

2.1.2. Etude du sous programme « Test BP » :

Aller sur l'onglet « test BP » :



- Le bouton poussoir est connecté sur la broche A2 du microcontrôleur
- On désire tester l'appui PUIS le relâchement du bouton :

Comportement attendu :

Début

A2 > BP // On lit la broche A2 et on stocke dans la variable BP

Faire tant que BP = 0 // Cette boucle test l'appui du bouton

A2 > BP

Fin Faire

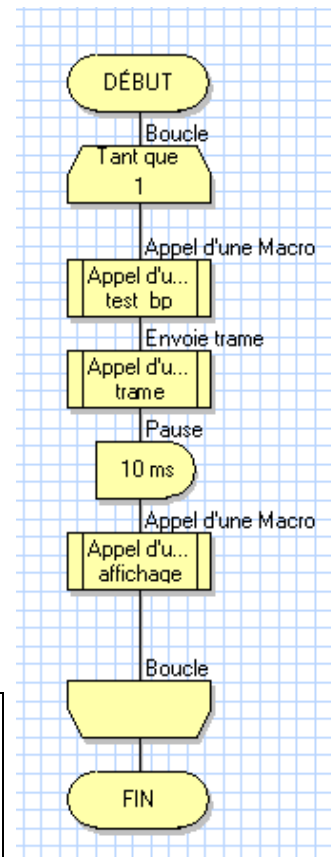
A2 > BP // On lit la broche A2 et on stocke dans la variable BP

Faire tant que BP = 1 // Cette boucle test le relâchement du bouton

A2 > BP

Fin Faire

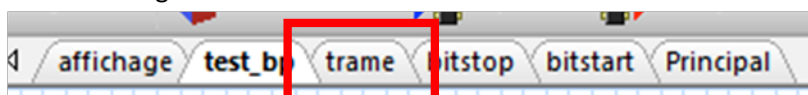
Fin



Q4. Effectuer sur feuille de copie l'algorithme respectant le comportement décrit ci-dessus (faire valider par le professeur) puis construire le programme avec Flowcode.

2.1.3. Etude du sous programme « Trame » :

Aller sur l'onglet « Trame » :



- Elle va faire appel à diverses fonctions qui ne seront pas toutes traitées ici
- On initialisera la trame avec des valeurs définies dans un tableau T de 8 cases de 8 bits. Les valeurs seront celles données dans l'exemple de la page 1 ;
- La trame doit être émise sur la broche A4 du microcontrôleur ;


Comportement attendu :

Début

```

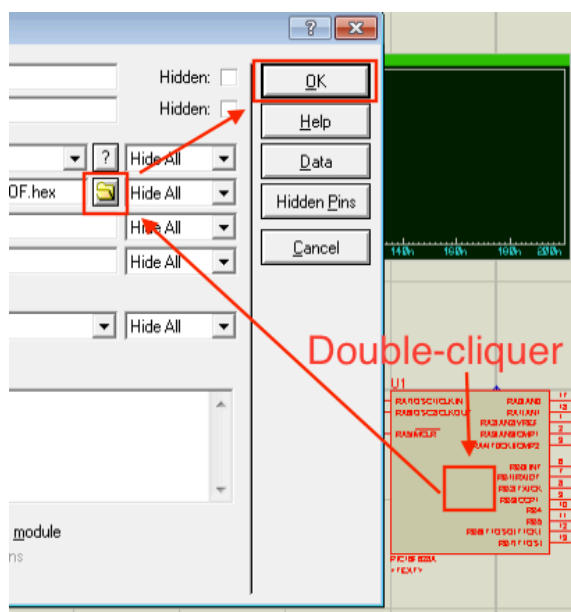
Bit_start // appel d'un sous-programme d'envoi d'un bit de start
Initialisation_tableau_T // Données à transmettre (8 octets)
i=0 // initialisation de la variable « i »
Tant que i<8 faire
  masque=1 // initialisation de la variable « masque »
  cpt=0 // initialisation de la variable « cpt »
  Tant que cpt<4
    Si (T[i] & masque=0) Alors
      RA4 = 0 // broche 4 du port A du microcontrôleur
    Sinon
      RA4 = 1 // Broche 4 du port A du microcontrôleur
    Temporisation_200us
    Masque = masque * 2
    cpt = cpt + 1
  Fin Tant que
  i = i + 1
Fin Tant que
bitstop // appel d'un sous-programme d'envoi d'un bit de stop
Fin
  
```

Q5. Compléter le programme « Trame » sous Flowcode en transcrivant les lignes rouges ci-dessus en algorithme.

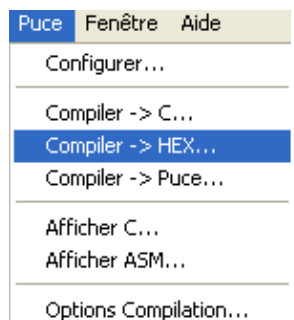
Q6. Vérifier que votre programme fonctionne en appuyant sur le bouton exécuter  et faites valider le programme par votre enseignant. Le modifier si nécessaire.

2.2. Test sur machine virtuelle :

On va maintenant procéder aux essais en machine virtuelle.




Sur Flowcode faire Puce puis Compiler→HEX. On génère ainsi une multitude de fichier.

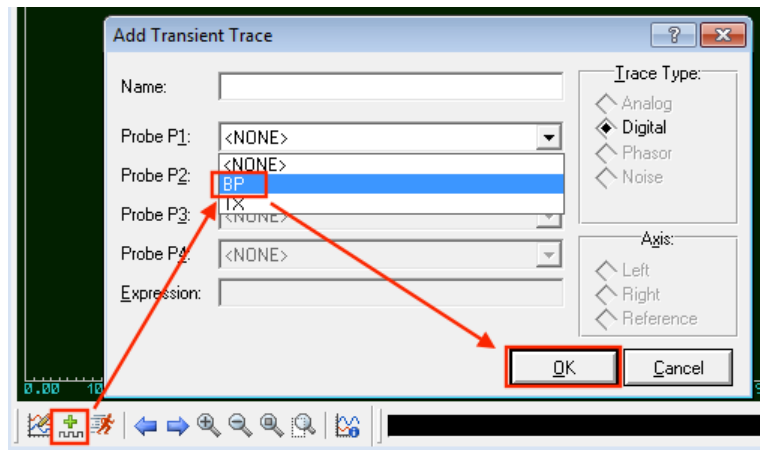


Vérifier qu'un fichier portant l'extension « .hex » a bien été créée.

On désire afficher et valider la trame émise. Pour ceci vous devez :

- Ouvrir le fichier **Trame_ELEVES_V3.DSN**.
- Double-cliquer sur le microcontrôleur et indiquer le programme en .hex

- Cliquer droit sur le graphe « digital analysis »
- Maximiser ce graphe
- Indiquer les signaux à afficher (voir ci-contre pour le signal « BP », recommencer pour « TX »)
- Simuler le graphe .
- Zoomer sur la dernière trame émise
- Imprimer le graphe (après validation du professeur).



- Q7. A la lecture du graphe complet (à l'écran), indiquer le nombre de fois ou le bouton poussoir a été pressé.
- Q8. Sur le graphe imprimé, colorier en bleu le bit de Start, en rouge l'entête et en vert le checksum (voir page 1 & 2)
- Q9. A la lecture de la trame imprimée, donner les 24 bits de données (sans prendre en compte l'entête et le checksum)
- Q10. Effectuer la conversion en décimal, en respectant le protocole défini page 1 (attention au LSB et MSB) et expliquer le nombre obtenu.

3. Pour aller plus loin...

Notre système original ne dispose pas de l'acquisition du nombre d'attaches posées. Il est donc nécessaire de comptabiliser les impulsions de gâchette.

- Q11. Compléter le travail déjà effectué afin de :
- Comptabiliser les impulsions de gâchette (il faut créer un nouveau bouton)
 - Afficher le nombre d'impulsions sur l'afficheur LCD
 - Mettre le résultat de comptage dans la trame à émettre
 - Valider le fonctionnement en machine virtuelle.

Indication : $1234\%10 = 4$ et $1234/10\%10 = 3$