Jeudi 14 Septembre 2017

Pilotage d'instruments

TP 1 : Liaison RS232 – Localisation GPS

IUT ORSAY Mesures Physiques – 2ème année

C.BUCHARD

Table des matières

I. Introduction – Objectifs 3

II. Etude theorique : la transmission de données 3

1. Système binaire et convention TTL 3

2. Transmission de chaines de caractères 3

a) Codage de l'information 3

b) Vitesse de transmission en liaison série RS-232 3

c) Méta-Data et Format des messages 4

d) Adresses 4

3. Branchements en liaison série 5

III. Communication avec LabVIEW 5

1. Verifier les branchements 5

2. Labview : Configuration du port série et Ecriture d'un message 6

3. LabVIEW : Lecture d'un message. 7

4. Lecture des messages du GPS 7

a) Principe 7

b) Code LabVIEW 8

IV. Conclusion 10

I. INTRODUCTION – OBJECTIFS

Nous allons durant ce TP rappeler les principes de base d'une liaison série RS232. Nous allons ensuite apprendre à configurer cette liaison, et à communiquer avec le périphérique.

Le périphérique est ici un GPS branché en liaison série avec l'ordinateur. Ce GPS envoi les informations de localisation sous forme de « trames » composées de caractères en format ASCII.

En connaissant le format de la trame, nous allons pouvoir extraire les informations de localisation avec LabVIEW, en écoutant le périphérique.

II. ETUDE THEORIQUE : LA TRANSMISSION DE DONNEES

1. SYSTEME BINAIRE ET CONVENTION TTL

Les systèmes de communication sont numériques. Ils utilisent le système binaire (0 ou 1). Chaque appareil respecte la convention TTL (*Transistor-Transistor Logic*). Les « 0 » et « 1 » du format binaire correspondent à des tensions précises. Par exemple, « 0 » en binaire représenter une tension égale à 0V, et « 1 » une tension égale à 5V. Les valeurs des tensions ne sont pas universelles : ainsi, il peut exister des appareils ou le chiffre binaire « 0 » est utilisé pour représenter $-12V$ et « 1 » pour $+12V$. Lorsque 2 appareils n'utilisent pas les mêmes niveaux de tension, il faut ramener les appareils à une échelle commune à l'aide d'un adaptateur.

Ainsi, dans une liaison série, il faut s'intéresser aux données échangées, mais également à l'électronique, pour injecter la bonne tension et éviter de détruire l'appareil.

2. TRANSMISSION DE CHAINES DE CARACTERES

a) Codage de l'information

La transmission de données est effectuée en binaire. Le seul moyen de transmettre des données (alphabétiques ou numériques) est de coder l'information en binaire. On peut alors introduire la notion de « chaine de bit » : c'est un nombre binaire qui permet de représenter une valeur numérique, ou alphabétique.

Dans la suite du TP, nous utiliserons le code ASCII. C'est un principe selon lequel chaque caractère utilisé (numérique, alphabétique, caractères spéciaux…) possède un unique code en binaire. La représentation logique du caractère est codée sur 8 bits.

b) Vitesse de transmission en liaison série RS-232

Lorsque nous transmettons des caractères avec une liaison série RS232, nous envoyons les caractères un par un. La vitesse de transmission est de $1\ bit/cycle$. Ainsi, pour transmettre un caractère, il faut 8 cycles (car le caractère est codé sur 8 bits).

Ce paramètre est important. Si nous envoyons trop rapidement des données, l'appareil va tout simplement être saturé. Dans le code LabVIEW, nous allons spécifier la vitesse de transmission de l'appareil.

c) Méta-Data et Format des messages

Lorsque nous transmettons une phrase, nous envoyons la représentation logique des caractères. Mais nous devons également envoyer d'autres informations, comme la signature de l'appareil, les différentes informations sur ce que nous transmettons, ou encore un signal de début et un signal de fin. Toutes ces données qui ne font pas parties de la chaine de caractère transmise sont appelées Méta-Data.

Ainsi, lorsque nous spécifierons la taille du buffer de réception (ou mémoire de réception), nous ne devrons pas simplement appliquer la taille du message en lui-même. Nous devrons également prendre en compte la taille qu'occupent ces méta-data.

Le message transmit sera en réalité composé de :

- Un bit de début indiquant le début du message
- Les données propres
- Un bit de parité, ayant pour rôle de détecter les erreurs
- 1 ou 2 bit(s) d'arrêt indiquant la fin du message.

Lors de la communication, le bit de poids faible (se trouvant le plus à droite) sera transmis en premier. Le bit de poids fort (le plus à gauche) sera transmis en dernier.

Remarque : Les chiffres binaires faisant référence à des tensions, nous pouvons observer la transmission des données avec un oscilloscope.

d) Adresses

Sur un bus série, il n'y a pas d'adressage. Nous pouvons donc connecter au maximum que 2 périphériques. Lorsqu'un périphérique est en mode « envoi », l'autre est en mode « écoute » et inversement. Nous savons donc que l'appareil qui ne parle pas est celui qui écoute et inversement. Nous n'avons pas besoin d'adressage pour identifier les appareils sur un port série.

Sur un bus type GPIB, nous pouvons connecter plus de périphériques. Chaque périphérique possède une adresse qui lui permet de communiquer avec le périphérique souhaité. L'adresse du périphérique est codée sur 5 bits (nombre de 0 à 30).

Il existe plusieurs modes de transmission :

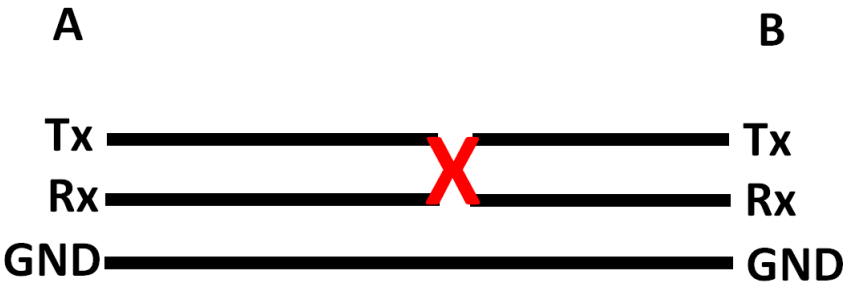
- Unidirectionnel : Nous avons un périphérique « émetteur » et un périphérique « récepteur ». Cette configuration ne change pas au cours du temps.
- Semi-Directionnel : Les périphériques peuvent envoyer ou recevoir des données. La transmission ne se fait pas de manière simultanée. Il n'y a qu'une opération à la fois.
- Bidirectionnel : Les périphériques transmettent et écoutent de manière simultanée.

3. BRANCHEMENTS EN LIAISON SERIE

Lorsque nous voulons que 2 appareils puissent envoyer et recevoir des données en liaison série, il faut effectuer certains branchements.

Comme les données font référence à des tensions, il faut une référence commune entre les 2 appareils. On relira les appareils A et B à une masse commune.

Lorsque l'appareil A transmet, l'appareil B doit recevoir et inversement. Il faut donc connecter le câble de transmission de A sur le câble de réception de B. A l'inverse, le câble de transmission de B doit être relié sur le câble de réception de A.

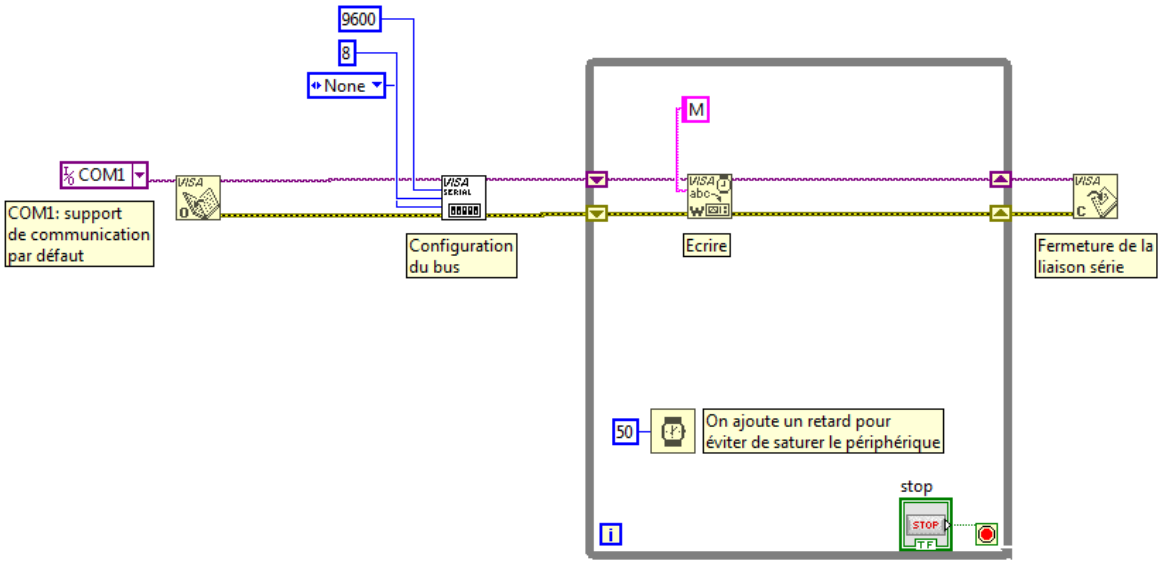


III. COMMUNICATION AVEC LABVIEW

1. VERIFIER LES BRANCHEMENTS

Par défaut, l'appareil connecté sur l'ordinateur à pour port série « COM 1 ». Pour vérifier les branchements, et le bon fonctionnement de l'appareil, nous allons recevoir les données brutes transmises par le GPS. Pour ce faire, nous allons utiliser le logiciel NI-MAX. Nous sélectionnons COM 1, puis « Ouvrir le panneau de tests ». Avec ce panneau, nous réglons le « Baut rate » à 4800 et nous faisons une lecture du buffer pour vérifier que l'ordinateur reçoit les données transmises par le GPS. Ensuite, il ne faut pas oublier de fermer cet utilitaire pour libérer le port série.

2. LABVIEW : CONFIGURATION DU PORT SERIE ET ECRITURE D'UN MESSAGE



Le premier objectif est de savoir écrire un message sur le port série avec LabView. Ce code permet d'écrire sur le port série la lettre « M ». Ce code est composé de plusieurs VIs :

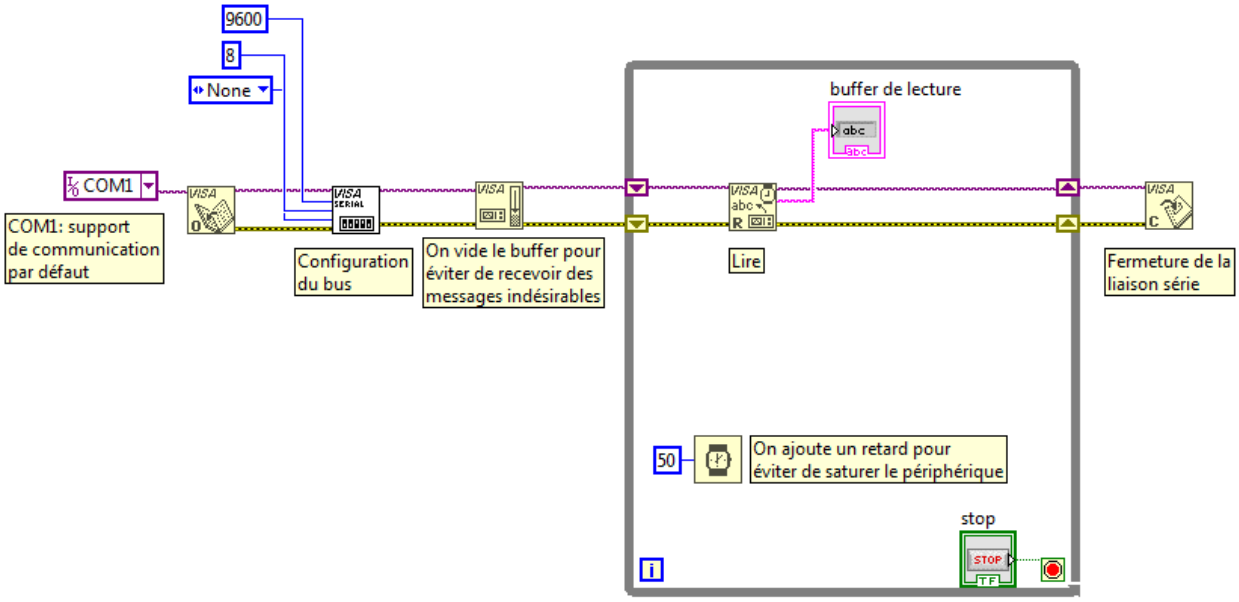
- Le VI « Visa Open » qui permet d'ouvrir le port sélectionné. Il s'agit ici de « COM 1 ».
- Le VI « Visa Configure Serial Port » qui nous permet de configurer le port. Nous avons ici paramétré un débit de 9600 bauds, la donnée est sur 8 bits, et il n'y a pas de bit de parité.
- Le VI « Visa Write » qui permet d'écrire le message. Ici, nous envoyons sur le port série la lettre « M ». Ce VI est placé dans une boucle While avec un retard de 50ms. Toutes les 50ms, nous envoyons ce caractère sur la liaison série. Le délai est important. Sans lui, on risquerait de saturer le port en envoyant plus de caractère que l'appareil n'est capable d'en réceptionner en une durée Δt.
- Enfin, le VI « Visa Close » qui permet de fermer la liaison lorsque l'utilisateur quitte la boucle while en appuyant sur le pouton stop.

Avec ce programme, le caractère « M » est envoyé toutes les 50ms sur le port série. Le périphérique va donc recevoir le caractère « M » toutes les 50ms.

Maintenant, nous voulons effectuer l'opération inverse : nous voulons recevoir des données avec LabVIEW.

3. LABVIEW : LECTURE D'UN MESSAGE.

Pour lire le buffer de sortie avec LabVIEW, on s'inspire du code précédent, en effectuant quelques modifications.



Nous retrouvons dans ce programme les mêmes instructions d'ouverture et de configuration du port (Vis « Visa Open » et « Visa Configure Serial Port ») que dans le programme d'écriture de message.

Nous pouvons remarquer que avant la boucle while, nous avons rajouté le VI « Visa Flush I/O Buffer » qui permet d'éffacer le buffer. Cette opération n'est pas obligatoire, mais est vivement conseillée pour ne pas recevoir de messages pouvant être déjà sur le port.

Le VI « Visa Write » a été remplacé par le VI « Visa Read » et nous avons relié un indicateur pour observer les messages transmis. Comme le code d'écriture, ce VI est dans une boucle while. Ainsi, toutes les 50ms, nous allons recevoir les données que le périphérique envoie.

Enfin, lorsque nous sortons de cette boucle, nous fermons la liaison avec le VI « Visa Close ».

4. LECTURE DES MESSAGES DU GPS

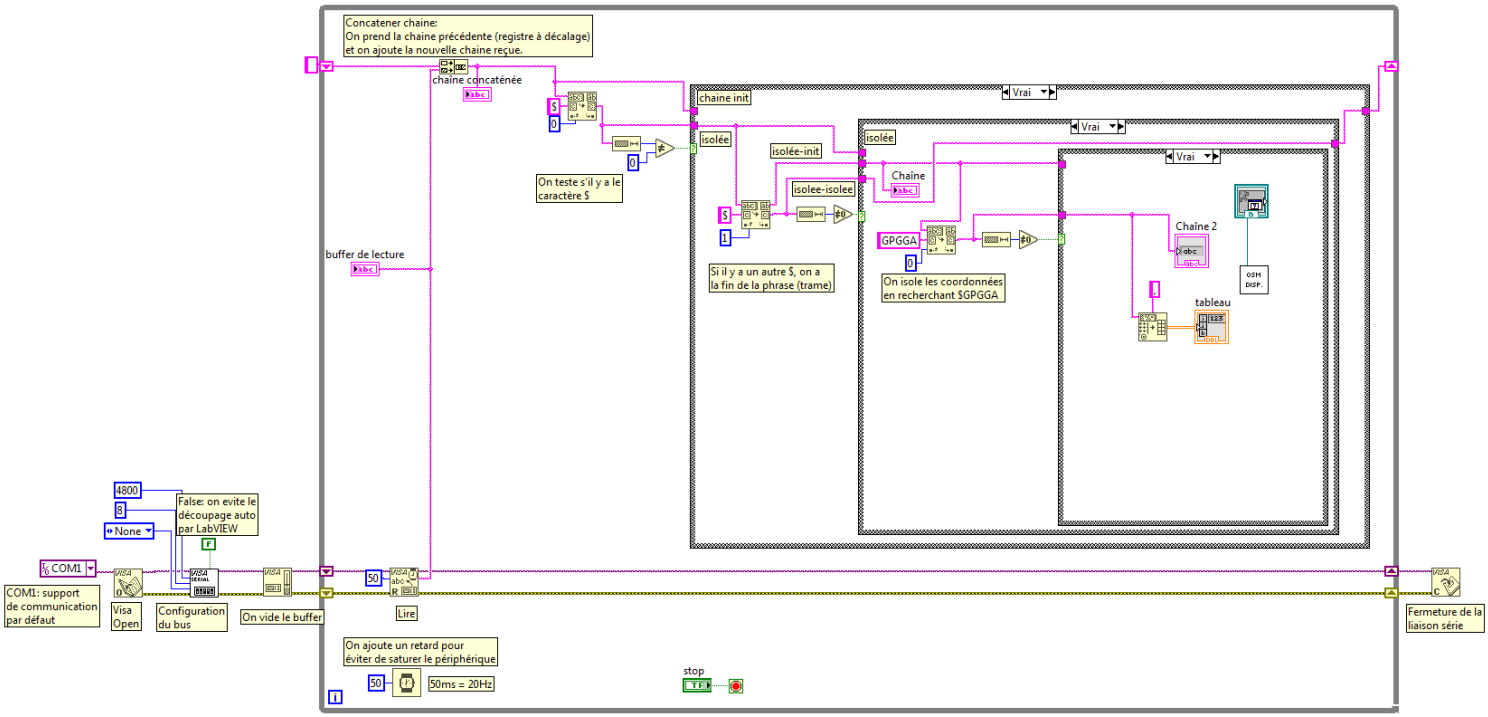
Le GPS utilisé transmet une trame dont le format est connu. Cette trame contient en particulier la position GPS que nous souhaitons récupérer sur LabView. Chaque trame commence par le caractère « $ ».

a) Principe

Sur LabView, une première étape consiste à lire le port série et à repérer la présence du caractère « $ » pour repérer le début de la trame. Si ce test est vérifié, alors nous allons effectuer la même opération pour repérer la fin de la trame, ***avec un décalage d'un caractère pour ne pas comptabiliser le caractère repéré avant***. Ainsi, on isolera une chaine de caractère contenant les informations GPS, et autres informations comme l'identification de l'appareil.

b) Code LabVIEW

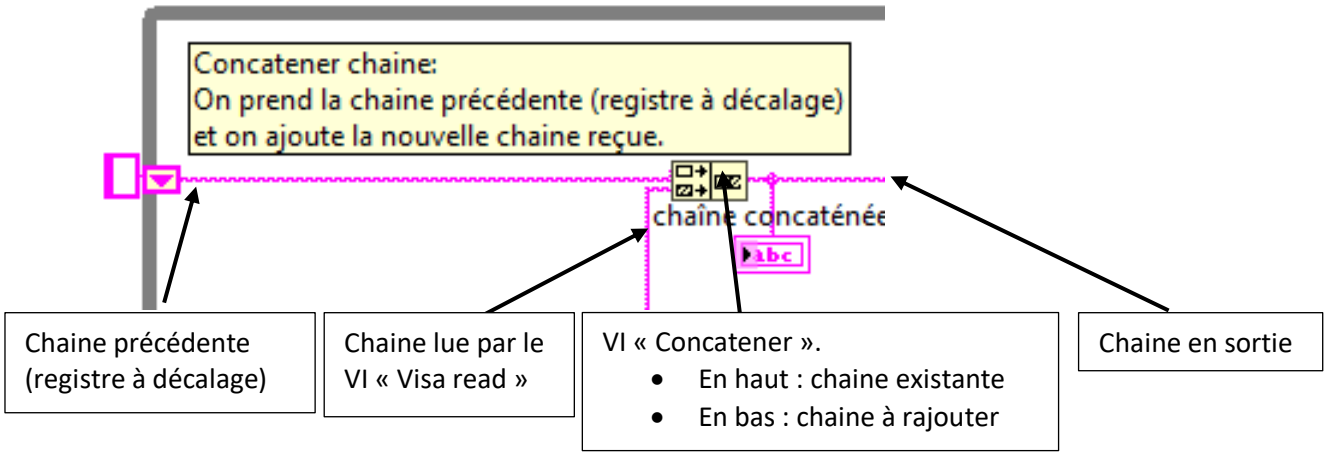
Voici le code LabVIEW permettant de lire le buffer et d'effectuer les tests pour identifier le début et la fin d'une trame.



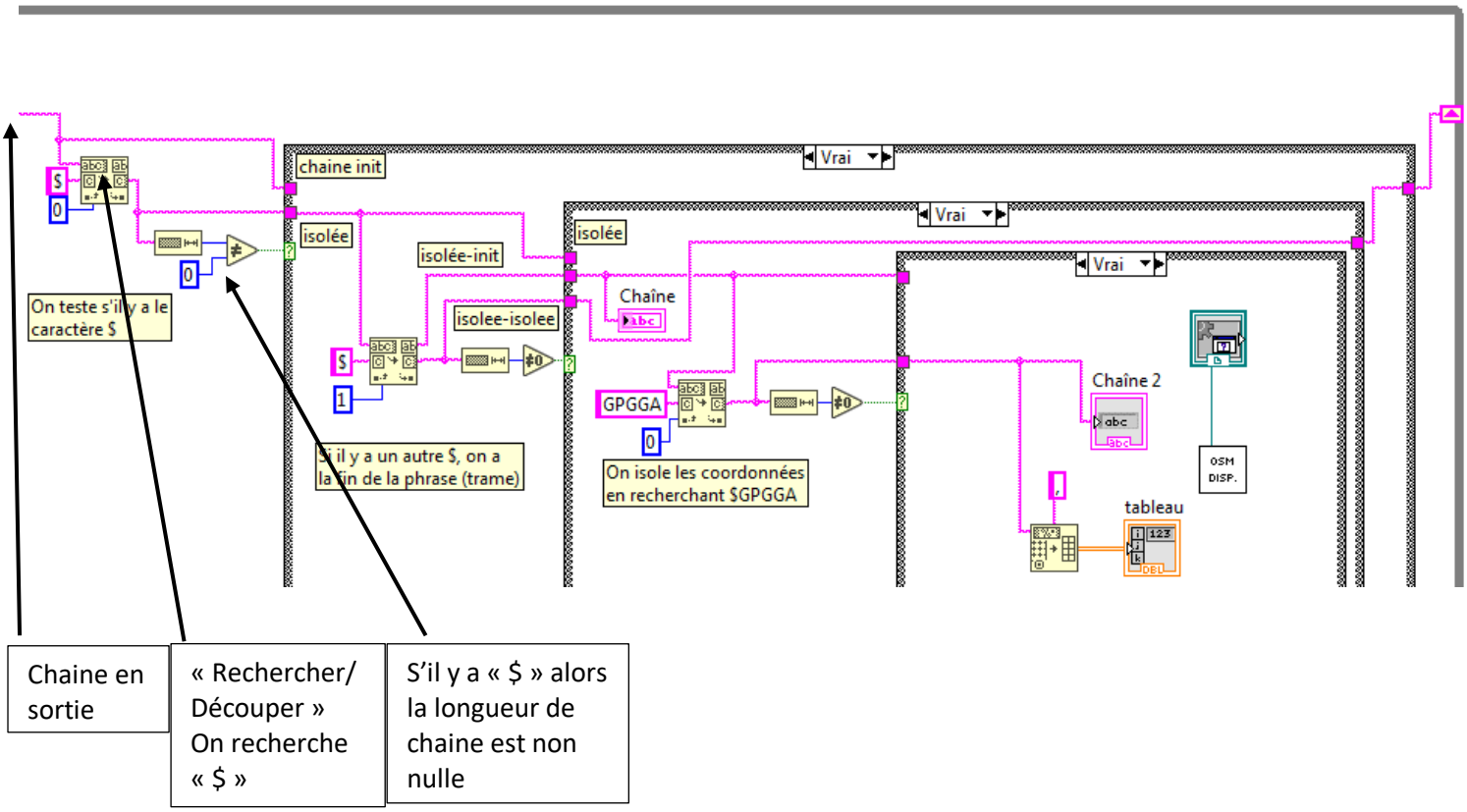
Dans ce programme, nous retrouvons les VI utilisés précédemment : « Visa Open », « Visa Configure Serial Port », « Visa Flush I/O Buffer », « Visa Read », « Visa Close ».

1) Assemblage des chaines de caractère

LabVIEW ne reçoit pas la trame en une fois, mais reçoit la trame en morceaux. La première opération consiste donc assembler la chaine reçue avec la chaine précédente. Ainsi, nous obtiendrons une seule chaine de caractère que nous allons pouvoir traiter. Cette opération s'effectue par ce code :



2) ***Recherche des chaines***



Chaine en sortie

« Rechercher/ Découper » On recherche « $ »

S'il y a « $ » alors la longueur de chaine est non nulle

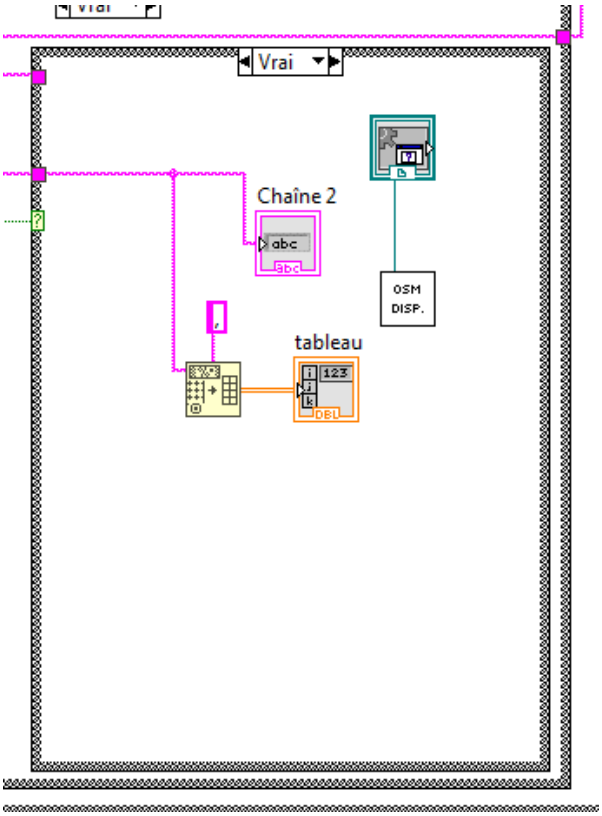
Le premier test sert à dédecter le premier caractère « $ » pour repérer le début de la trame. Une fois de début repéré, on injecte le « $ » et les caractères suivants dans un nouveau test. On recherche à nouveau le caractère « $ » pour repérer la fin de la trame.

Lorsque la fin de la trame est trouvée (nous avons alors une chaine ressemblant à ceci : « $GPGGAcoordonnees$ »), il faut isoler la chaine contenant les coordonnées, c'est-à-dire ce qui se trouve après « GPGGA ». Nous effectuons un 3ème test pour vérifier la présence de « GPGGA ». Si cette trame est trouvée, alors on aura les coordonnées.

Les opérations ce recherche de caractère et d'extraction de chaine (avant ou après le caractère recherché) se font avec le VI « Rechercher/Découper chaine ».

3) ***Affichage des coordonnées (théorie)***

On obtient à la fin les informations de localisation séparées par des virgules. L'idée pour extraire uniquement les informations de localisation est de créer un tableau de nombres. Comme nous connaissons le séparateur (« , »), nous utilisons le VI « Chaine au format tableur ».



Nous prenons la chaine de caractère épurée que nous injectons dans ce VI. Nous spécifions le séparateur « , ». A chaque « , », le VI va injecter le nombre dans une colonne. Il suffit ensuite d'injecter les colonnes contenant les informations de localisation au format numérique dans le SousVI distribué « OSM DISP » pour afficher les coordonnées sur une carte.

Malheureusement, même si ce principe semble correct, cette partie de code génère des erreurs lors de l'exécution du programme global.

IV. CONCLUSION

Durant ce TP, nous avons révisé les bases du système de communication de données par liaison série entre 2 périphériques. Nous avons ensuite réussi à lire les données brutes envoyées par le GPS. Une fois ces données lues, nous les avons rassemblées en une unique chaine de caractère. Puis nous avons isolé les trames en recherchant le caractère « $ » utilisé lors de chaque nouvelles trames. Malheureusement, l'affichage des coordonnées ne fonctionne pas. En affichage, en supprimant la dernière partie de code, nous obtenons les coordonnées sous forme de chaine de caractères. Mais nous ne pouvons pas afficher ces coordonnées sur une carte.