Mise en oeuvre d'un périphérique USB sous Windows

Thierry Vaira <tvaira@free.fr>

Table des matières

Site : tvaira.free.fr

Mise en oeuvre d'un périphérique USB sous Windows

Ici, nous allons mettre en oeuvre le périphérique USB suivant : un **récepteur GPS de MTK**. Il émet des trames suivant la norme **NMEA 0183**. Il sera pris en charge par le système d'exploitation comme un **port série virtuel**.

La norme NMEA 0183

La norme **NMEA 0183** est une spécification pour la communication entre équipements marins, dont les équipements GPS. Elle est définie et contrôlée par la *National Marine Electronics Association* (NMEA), association américaine de fabricants d'appareils électroniques maritimes.

La norme 0183 utilise une simple communication série pour transmettre une "phrase" (*sentence*) à un ou plusieurs écoutants. Une trame NMEA utilise tous les caractères ASCII.

Exemple : *Waypoint Arrival Alarm*

```
$GPAAM,A,A,0.10,N,WPTNME*32
```

Il existe plus d'une trentaine de trames GPS différentes. Le type d'équipement est défini par les deux caractères qui suivent le `$`. Le type de trame est défini par les caractères suivants jusqu'à la virgule.

Par exemple :

```
$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*0E
```

C'est une trame **GP** (pour GPS) de type **GGA**. La trame GGA est très courante car elle fait partie de celles qui sont utilisées pour connaître la position courante (longitude, latitude et altitude) du récepteur GPS (www.coordonnees-gps.fr).

- `$` : délimiteur de début de trame
- `GP` : Id du "parleur" (GPS)
- `GGA` : Type de trame (*Global Positioning System Fixed Data*)
- `064036.289` : Trame envoyée à 06h40m36,289s (heure UTC)
- `4836.5375,N` : Latitude ddmm.mmmm -> 48,608958° Nord = 48°36'32.25" Nord
- `00740.9373,E` : Longitude dddmm.mmmm -> 7,682288° Est = 7°40'56.238" Est
- `1` : Type de positionnement (le 1 est un positionnement GPS)
- `04` : Nombre de satellites utilisés pour calculer les coordonnées
- `3.2` : Précision horizontale ou HDOP (Horizontal dilution of precision)
- `200.2,M` : Altitude 200,2, en mètres
- `,,,,,0000` : D'autres informations peuvent être inscrites dans ces champs
- `*0E` : Somme de contrôle de parité, un simple XOR sur les caractères précédents
- `<CR><LF>` : délimiteur de fin de trame

*Remarque : Les trames NMEA font toutes référence à l'ellipsoïde **WGS84** comme base de son système de coordonnées.*

Chaque trame a sa syntaxe propre, mais selon le cas elles peuvent ou doivent se terminer, après le `*`, par un octet formant une somme de contrôle (*checksum*) qui permet de détecter une erreur dans la transmission.

La somme de contrôle à la fin de chaque phrase est le **OU EXCLUSIF** (`XOR`) de tous les octets de la phrase à l'exclusion du premier caractère (`$`) et jusqu'au caractère avant l'étoile (`*`). Cf. C implementation of checksum generation.

Site officiel : www.nmea.org

Le port série virtuel

Un port série virtuel est une **solution logicielle qui émule un port série standard**.

Cela permet généralement :

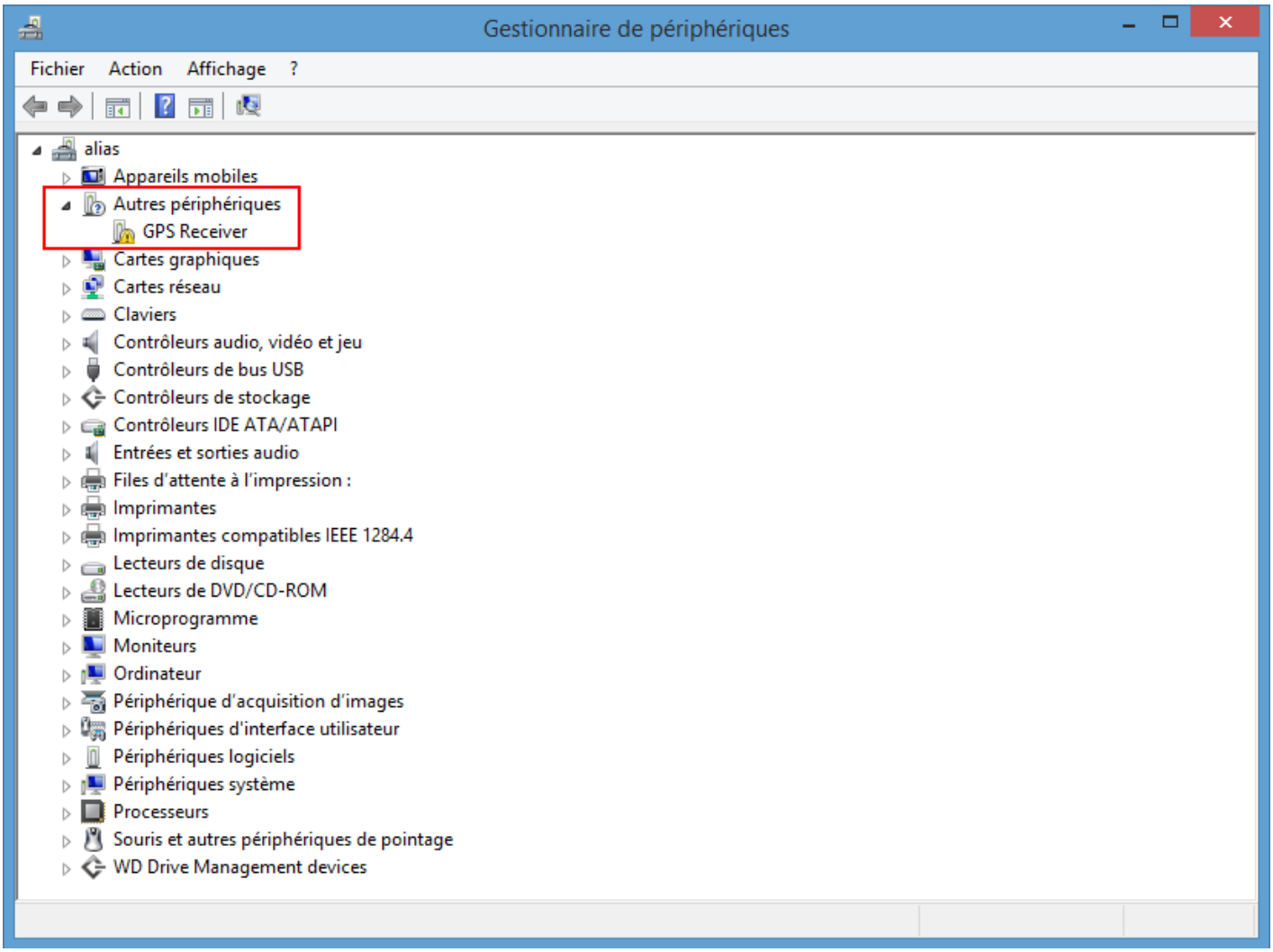
- d'augmenter le nombre de ports série (sans installation de matériel supplémentaire mais dans les limites des ressources disponibles)
- de partager les données en provenance d'un périphérique entre plusieurs applications
- de raccorder un périphérique série standard (RS232, ...) sur un port USB avec un adaptateur (manque ou absence de ports série physiques disponibles)

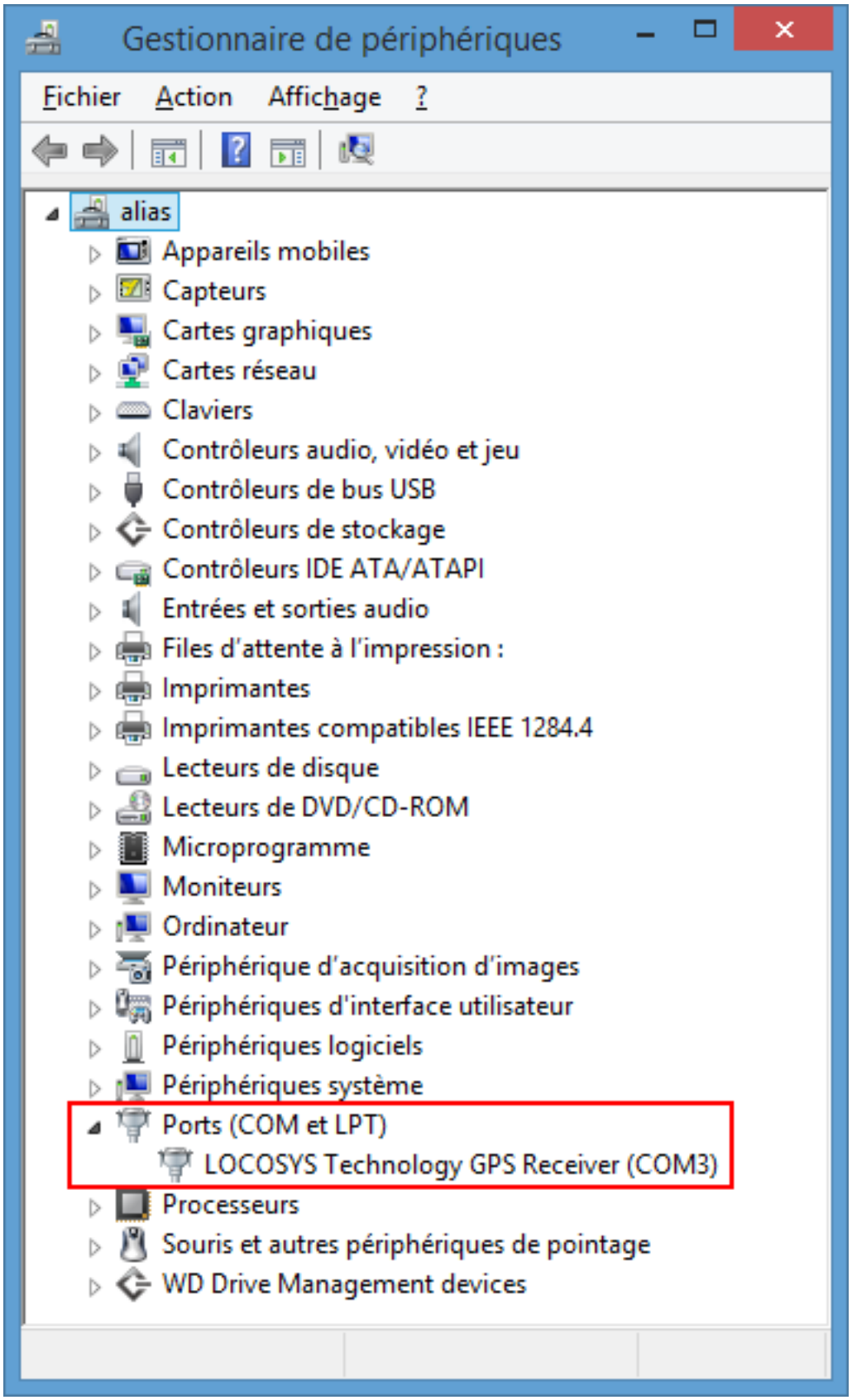
Pour établir une communication effective via un port série physique (RS-232) ou virtuel, il est nécessaire de définir le protocole utilisé : notamment, le **débit de la transmission (en bits/s)**, le codage utilisé, le découpage en trame, etc. La norme RS-232 laisse ces points libres, mais en pratique on utilise souvent des trames d'un caractère ainsi constituées :

- 1 bit de départ (START)
- **7 à 8 bit de données**
- **1 bit de parité optionnel (aucune, paire ou impaire)**
- **1 ou plusieurs bits d'arrêt (STOP)**

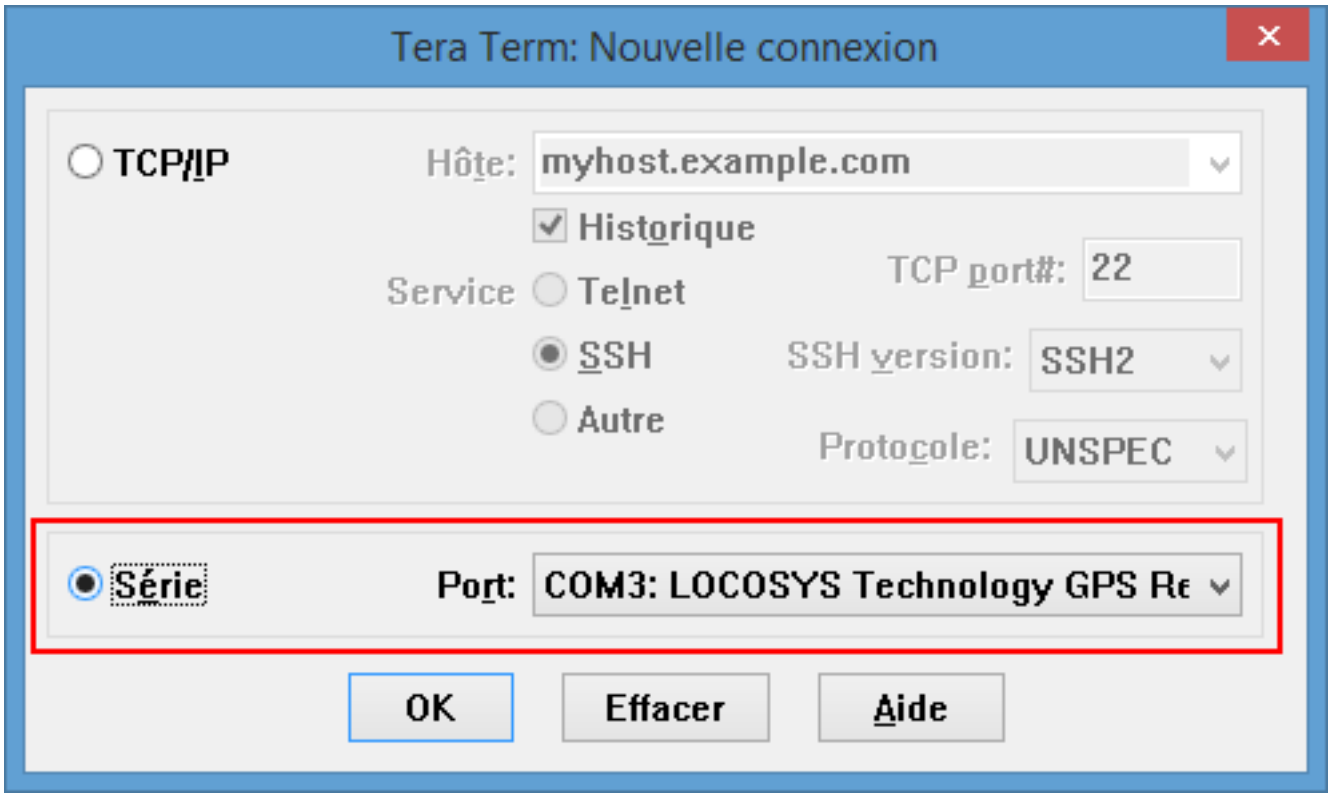
Tests

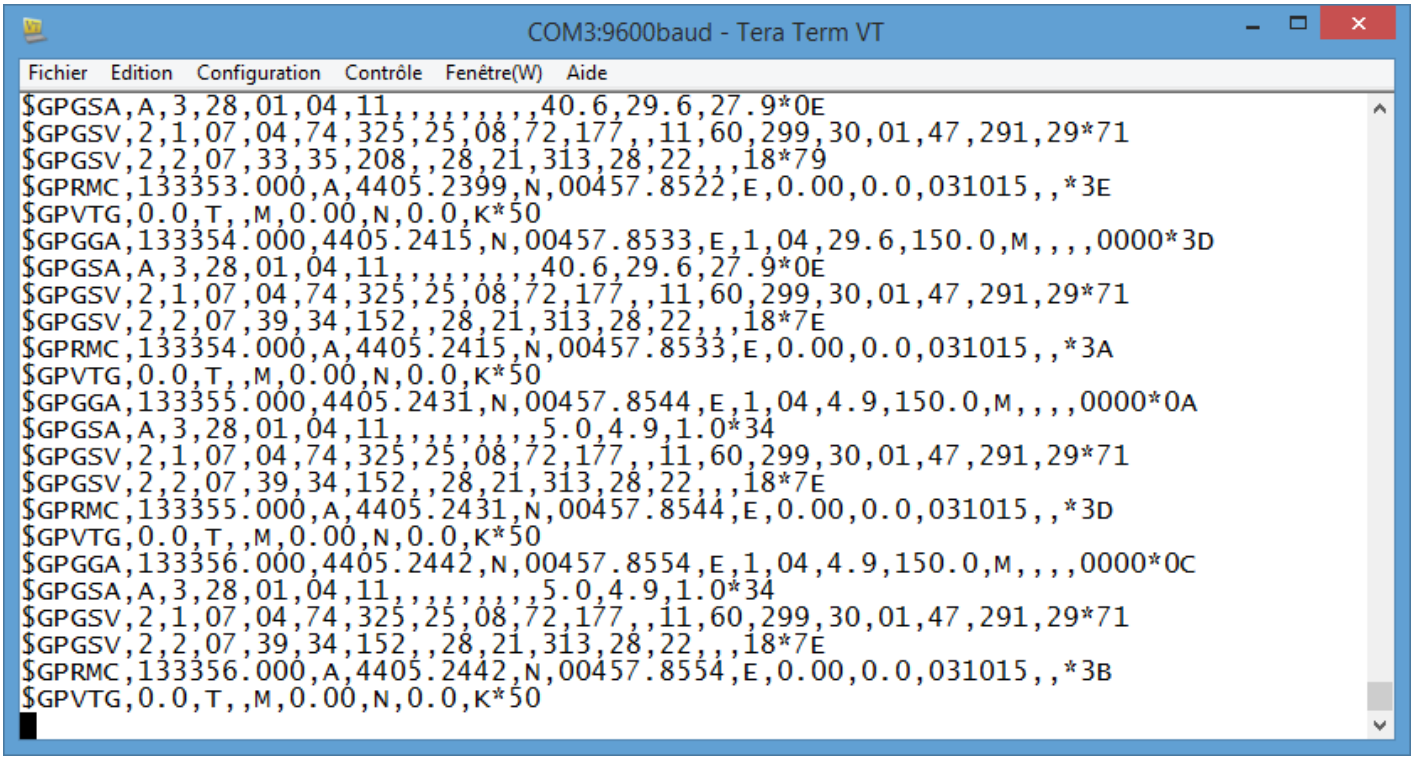
- **Brancher maintenant votre périphérique USB.**
- Ensuite, démarrer le "gestionnaire de périphériques" pour vérifier la prise en charge du périphérique :
  - Raccourci clavier : "Windows + R" -> Commande "Exécuter"
  - `DEVMGMT.MSC` -> Gestionnaire de Périphériques





– Ici, le périphérique USB sera accessible via un port série virtuel `COM3`
– On peut maintenant tester la réception de phrases NMEA 0183 avec le logiciel TeraTerm :





Programmation en C sous Windows

```
// Lit une trame NMEA 183 sur le port série virtuel
// Version Windows

// Attention : vérifiez le nom du fichier de périphérique et la configuration du port série
```



```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>

// On fixe la taille maximum à 128 octets
#define MAX 128

//#define DEBUG

int main(void)
{
    HANDLE hPort = INVALID_HANDLE_VALUE; // Handle sur le port série
    DCB old_dcb; // anciens parametres du port série
    DCB dcb; // parametres du port série
    COMMTIMEOUTS comout = {0}; // timeout du port série
    COMMTIMEOUTS oldTimeouts; // ancien timeout du port série
    char nomPeripherique[16] = "COM3"; // nom du port série
    char* buffer = (char *)NULL; // un buffer
    char c = 0; // un caractère
    int fini = 0; // fin de reception d'une trame
    int i = 0;
    DWORD n = 0;
    BOOL succes;
    int debutTrame = 0; // debut de reception d'une trame

    /* Ouverture de la liaison serie */
    hPort = CreateFile( "COM3",
                        GENERIC_READ | GENERIC_WRITE,
                        0,
                        0,
                        OPEN_EXISTING,
                        0,
                        0);

    if( hPort == INVALID_HANDLE_VALUE )
    {
        printf("Erreur d'ouverture du peripherique COM1 !\n"); // message d'erreur client
        exit(1);
    }
    //printf("Peripherique ouvert avec succes !\n");

    /* Lecture des parametres courants  */
    GetCommState(hPort, &dcb);
    old_dcb = dcb; // sauvegarde l'ancienne configuration

    /* Liaison a 9600 bps, 8 bits de donnees, pas de parite, lecture possible */
    dcb.BaudRate = CBR_9600;
    dcb.ByteSize = 8;
    dcb.StopBits = ONESTOPBIT;
    dcb.Parity = NOPARITY;
    dcb.fBinary = TRUE;
    /* pas de control de flux */
    dcb.fOutxCtsFlow = FALSE;
    dcb.fOutxDsrFlow = FALSE;

    /* Sauvegarde des nouveaux parametres */
    if( !SetCommState(hPort, &dcb) )
    {
      printf("Impossible de configurer le port COM1 !");
      CloseHandle(hPort);
      exit(1);
    }
    SetupComm(hPort, 2048, 1024);
    GetCommTimeouts(hPort, &oldTimeouts);
```



```
SetCommTimeouts(hPort, &comout);
//printf("Sauvegarde de la nouvelle configuration avec succes !\n");

/* Lecture de MAX octets */
buffer = (char *)malloc((MAX+1) * sizeof(char)); // on alloue un buffer
i = 0;
debutTrame = 0;
do
{
    succes = ReadFile(hPort, &c, 1, &n, NULL);
    if(!succes)
    {
        printf("erreur : %d\n", GetLastError()); // cf. FormatMessage
        //break;
    }
    #ifdef DEBUG
    printf("read : %d -> 0x%02X ", n, (unsigned char)c);
    #endif
    // est-ce qu'on a lu 1 caractère ?
    if(n == 1)
    {
        // est-ce le début d'une trame ?
        if(c == '$')
        {
          debutTrame = 1;
        }
        // est-ce qu'on est entrain de lire une trame ?
        if(debutTrame == 1)
        {
          // on copie le caractère lu dans le buffer
          *(buffer + i) = c;
          i++;
        }
        // est-ce que c'est la fin de la trame qu'on est entrain de lire ?
        if (debutTrame == 1 && c == '\n')
        {
          fini = 1;
          debutTrame = 0;
        }
    }
    /*else
    {
        if(n == -1)
        {
            printf("Erreur de lecture\n"); // message client
            //fini = 1;
        }
        else
        {
            printf("Aucune lecture !\n"); // message warning
        }
    }*/
}
while(i <= MAX && fini == 0);

*(buffer + i) = 0x00; // fin de chaîne (attention à i > MAX !)

printf("Trame lue :\n%s\n", buffer);

/* Restaure les anciens parametres du port série */
SetCommState(hPort, &old_dcb);
SetCommTimeouts(hPort, &oldTimeouts);
/* Fermeture du port série */
CloseHandle(hPort);
//printf("Fermeture du peripherique avec succes !\n");
```



```
    free(buffer); // on libère la mémoire allouée

    return 0;
}
```

Vous pouvez compiler ce programme avec **GCC** (MinGW) ou à partir d'un EDI (DevCpp, Code : :Blocks, . . .).

```
C:\> a.exe
Trame lue :
$GPZDA,001448.799,06,01,1980,,.0000,E,0,00,0,00,,,M,,,,0000*01
```

Code source : test-port-com-usb-windows.c

Conversions des coordonnées GPS

"$GPGGA,084222.000,4405.2015,N,00457.8908,E,1,05,1.7,26.5,M,,,,0000*3F"

Latitude et longitude dans le format : ddmm.mmmm et dddmm.mmmm

Conversion format GPS en degré décimal dd.dddd

dd + mm.mmmm/60

4405.2015 —> 44 + 05.2015/60 = 44,0867 degres

Conversion degré décimal dd.dddd en Degres Minutes decimal-Seconds (D° M' S")

d = 44.0823

D = int(d) —> 44

M = int((d - D) x 60) —> ((44.0823 - 44) x 60) = 4,938 = 4

s = (d - D - M/60) x 3600 —> (44.0823 - 44 - 4/60) x 3600

ou s = (d - D) x 3600 - M x 60

44.0867 —> 44° 5' 12.12"

Liens

- www.coordonnees-gps.fr
- www.sunearthtools.com

Retour au sommaire

