

## Compte rendu : Feature #13630

### Modification d'un script C permettant d'envoyer des données sur le port série

#### Rappel cahier des charges :

Coder un script en .c qui émet sur le port série des données extraites d'un .csv et qui les récupère ensuite sur le même port série.

#### Fonctionnement du code :

Tout d'abord, pour modifier le nom du fichier csv qu'on souhaite ouvrir, il suffit de modifier le nom de cette constante.

```
16 | const char scilabCSV[] = "csvData.csv";// Scilab csv file name
```

La partie ci-dessous contient tout le protocole d'ouverture du fichier .csv :

```
34 | printf("\n\n +=====+");
35 | printf("\n |                OPENING .CSV FILE                |");
36 | printf("\n +=====+\n");
37 | /* opening file for reading */
38 | printf("\n\n    opening file...");
39 |
40 | FILE *fp = fopen(scilabCSV, "r");
41 |
42 | if(fp == NULL)
43 | {
44 |     perror("\n\nError opening file");
45 |     return 1;
46 | }
47 | rewind(fp);
48 | if (fp!=NULL)
49 | {
50 |     printf("\n\n    File opening successfull");
51 | }
52 |
```

Pour changer le port de communication, il faut modifier le pointeur suivant :

```
22 | char *pcCommPort = "COM8";
```

Ci-dessous sont les paramètres de communication du port série :

```
93 | dcbSerialParams.BaudRate = CBR_115200;           // Setting BaudRate = 115200
94 | dcbSerialParams.ByteSize = 8;                     // Setting ByteSize = 8
95 |                                                    //dcbSerialParams.
96 | dcbSerialParams.StopBits = ONESTOPBIT;            // Setting StopBits = 1
97 | dcbSerialParams.Parity = NOPARITY;                // Setting Parity = None
98 |
```

Pour récupérer les valeurs dans le .csv, on utilise :

```
128     char buff1 [1024];
129
130     while (fgets(buff1, sizeof(buff1), fp))
```

On copie ensuite le buffer dans la variable lp :

```
133     char lp [strlen(buff1)];
134     strcpy(lp, buff1);
```

Ici, le programme envoie le contenu de la variable lp (une ligne du fichier .csv), et teste en même temps un potentielle erreur d'envoi :

```
144     if (!WriteFile(hComm, lp, dNoOFBytestoWrite,
145                   &dNoOfBytesWritten, NULL))
146     {
147         printf("Error writing text to %s\n", pcCommPort);
148     }
```

Après l'envoi, on attend la bonne réception des caractères renvoyés par l'outillage :

```
174     Read_Status = WaitCommEvent(hComm, &dwEventMask, NULL);
```

Ici, on vient lire le buffer série avec la variable SerialBuffer, et on teste une erreur possible

```
192     if (!ReadFile(hComm, SerialBuffer, BUFFERLENGTH, &NoBytesRead, NULL))
193     {
194         printf("\n |Reading error");
195     }
196     printf("%s", SerialBuffer);
```

Enfin, on utilise des "memset" pour réinitialiser les variables (sans ca on se retrouve avec des morceaux des variables précédentes dans les nouveaux buffers à chaque lignes.

```
200     /*-----resetting values-----*/
201
202     memset(lp, 0, sizeof(lp));
203     memset(buff1, 0, sizeof(buff1));
204     memset(SerialBuffer, 0, sizeof(SerialBuffer));
205
```

Après toutes les boucles de lecture du csv, on ferme la liaison série et le fichier :

```
211     //Closing the Serial Port
212     fprintf(stderr, "Closing serial port...\n");
213
214     if(CloseHandle(hComm))
215     {
216         printf("File closed.\n");
217     }
218     else
219     {
220         printf("Error file not closed.\n");
221         exit(EXIT_FAILURE);
222     }
223     fclose(fp);
```

PS : toutes les fonctions "sleep(1000)" permettent de temporiser le programme qui overflow sinon.

### L'outillage :

```
#include <Wire.h>

const unsigned int MAX_MSG_LN = 1024;

void setup()
{
  Serial.begin(115200);
  pinMode(13,OUTPUT); //LED_BUILTIN activation
}
//Create a place to hold the incoming message
static char msg[MAX_MSG_LN];
static unsigned int msg_pos = 0;

void loop()
{
  //Check to see if anything is available in the serial receive buffer

  while (Serial.available())
  {
    //Read the next available byte in the serial receive buffer
    char inByte = Serial.read();

    //Message coming in (check not terminating character)
    if ((inByte != '\n') && (msg_pos < MAX_MSG_LN - 1 ))
    {
      digitalWrite(13,HIGH);          //visual identification

      msg[msg_pos] = inByte;
      msg_pos++;
    }

    //Full message received

  else
  {
    digitalWrite(13,LOW);          //visual identification
    msg[msg_pos] = '\0';

    //Print the message (or do other things)
    Serial.println(msg);

    //Reset for the next message
    msg_pos = 0;
  }
}
```

Ici le programme (arduino) s'active dès qu'une donnée est envoyée dans la liaison série.

La fonction Serial.read () permet de récupérer la valeur stockée dans le buffer série.

Cette valeur est ensuite stockée dans un tableau de caractère (msg), pour ensuite tout renvoyer (la ligne du fichier csv communiquée par le programme .c) par la fonction Serial.println dans la liaison série.

Une LED est activée lors de la réception des données et éteinte lors de l'émission et le reste du temps (fonction digitalWrite), pour une identification visuelle.