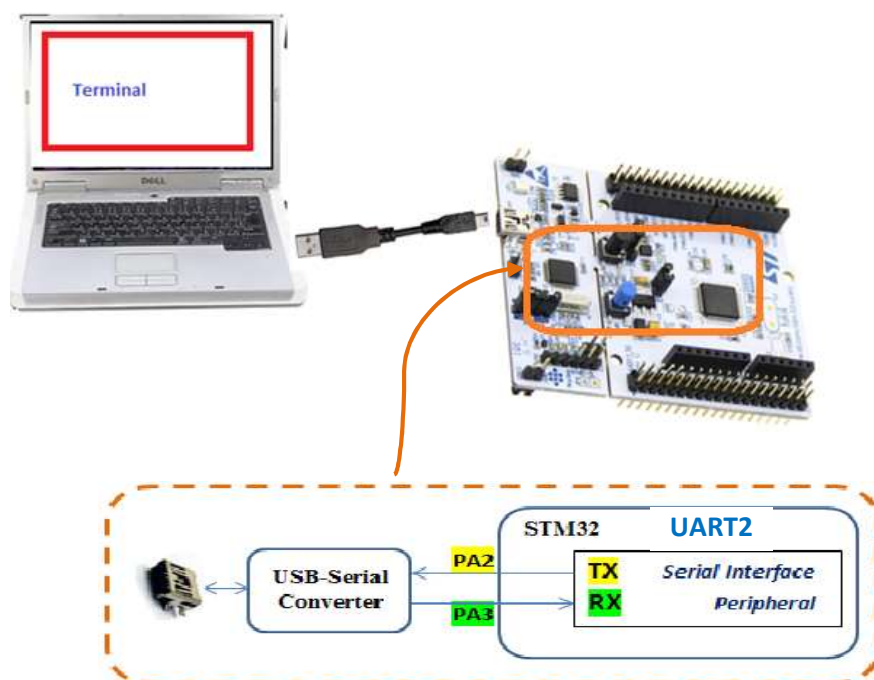


## Envoi/Réception de données sur le port série UART (STM32F1) en mode scrutation « Manip6\_SendRcvSerial »

Il s'agit d'une application permettant d'envoyer et recevoir des données (chaînes de caractères) sur le port série **UART2** en utilisant la scrutation (test des bits de Flag TXE et RXNE).

En effet, sur la carte Nucleo, **l'UART2** est le seul capable de communiquer directement avec le PC puisqu'il est relié directement un convertisseur Série/USB (intégré dans la carte Nucleo F103). « Voir Figure ci-dessous »



Après le démarrage (reset), le programme commence par envoyer une première chaîne de 20 caractères (password). L'envoi est effectué en utilisant la scrutation (test du bit de flag).

```
char password[]="ABCDEFGHIJ0123456789";
```

```
while (charcount<20)
{
    while (USART_GetFlagStatus(USART2,USART_FLAG_TXE)==RESET); //
    USART_SendData(USART2,password[charcount]); //Send Data
    charcount++;
    Delay(0xFFFF); // Wait before sending next char
}
```

Ensuite, le programme reçoit d'une façon continue les caractères envoyés par le PC et les place dans un array *rcvBuffer*. La réception est également effectuée en utilisant la scrutation (test du bit du flag).

A chaque réception d'un caractère, la led (PA5) est allumée.

```
while(1)
{
    while (USART_GetFlagStatus(USART2,USART_FLAG_RXNE)==RESET); //wait
    rcvBuffer[rcvcount++] = USART_ReceiveData (USART2); //read character
    GPIO_SetBits (GPIOA, GPIO_Pin_5);
    Delay(0xFFFFF);
    GPIO_ResetBits (GPIOA, GPIO_Pin_5);
}
```

Parallèlement, à chaque appui sur le bouton poussoir (PC13), une interruption est générée. Au niveau du Handler une deuxième chaine de 20 caractères (password2) est envoyée. L'envoi est également effectué en utilisant la scrutation (test du bit de flag).

```
char password2[]="abcdefghij9876543210";
```

```
extern char password2[];
uint8_t pwd2count=0;

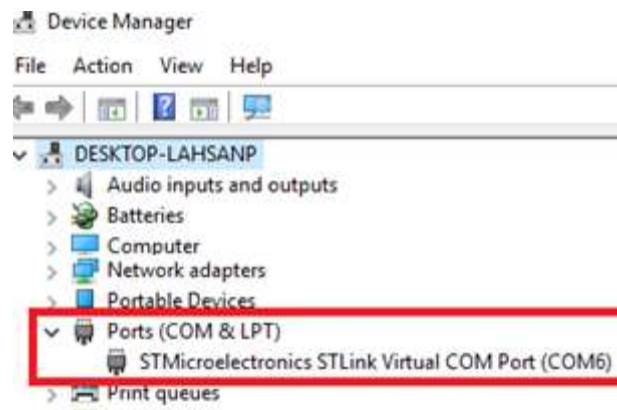
void EXTI15_10_IRQHandler(void)
{
    while (pwd2count<20){
        while (USART_GetFlagStatus(USART2,USART_FLAG_TXE)==RESET); //wait
        USART_SendData(USART2,password2[pwd2count]); //Send Data
        pwd2count++;
        Delay(0xFFFFF); // Mask The Rebound
    }
    pwd2count=0;
    EXTI_ClearITPendingBit(EXTI_Line13);
}
```

Évidemment, une fois l'interruption générée, il faut attendre l'exécution complète du Handler (envoi de la chaine de 20 caractères) avant de retourner au main (réception de données).

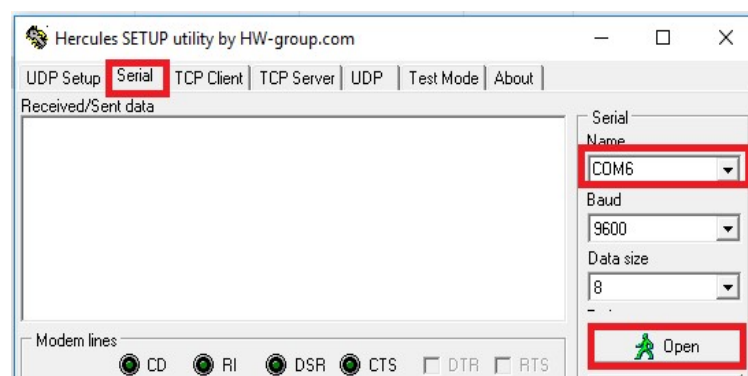
## **1- Test de la communication entre Carte et PC**

Pour le test, il faut utiliser au niveau du PC une application « terminal » : l'application « *Hecrules* » dans ce cas.

Commencer par récupérer le numéro du port série (COMx) attribué à la carte Nucleo au niveau du PC (au niveau du Device Manager de Windows)



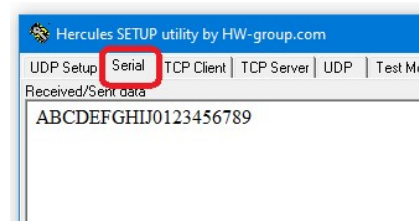
Lancer Hercules et ouvrir le port série attribué à la carte



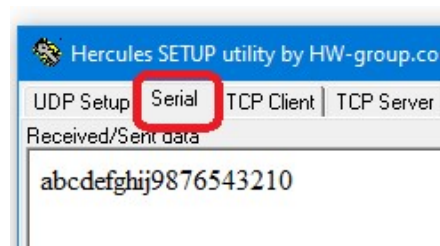
Charger l'application et l'exécuter sur la carte Nucleo.

### Envoi de données

- Dès le démarrage la 1<sup>ère</sup> chaîne de caractères devrait être affichée au niveau de l'interface « Hercules ».



- A Chaque appui sur le bouton (PC13), la 2<sup>ème</sup> chaîne de caractères est affichée.



## Réception de données

Il faut dans ce cas envoyer des données à partir du PC (Application Hercules) qui seront reçues par la carte Nucleo.

Pour vérifier que les données ont été reçues, il est possible de visualiser en temps réel le contenu de la chaîne **rcvBuffer** :

- Sélectionner le menu **View >> Watch Window >> Watch1** (Figure-1).
- Saisir le nom de la variable à visualiser. (Figure-2)



Figure-1

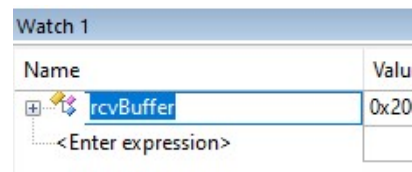


Figure-2

Pour l'envoi des données à partir du PC (Hercules), il est possible de saisir un message complet et d'appuyer sur **Send** (Figure 1) ou bien de saisir les caractères au niveau de l'interface même (Figure-2)

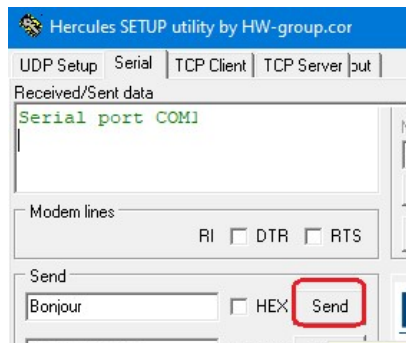


Figure-3

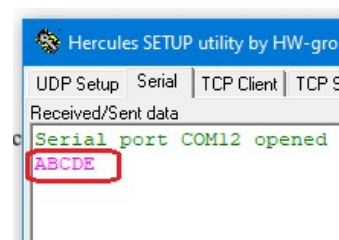


Figure-4

Le message envoyé à partir de « Hercules » devrait être placé dans la chaîne **rcvBuffer**

Name	Value
rcvBuffer	0x20000058 rcvBuffer[
[0]	0x41 'A'
[1]	0x42 'B'
[2]	0x43 'C'
[3]	0x45 'D'
[4]	0x45 'E'
[5]	0x00
rcv	...

## Test de la communication en mode « Simulation »

Pour choisir le mode « simulation », cliquer (droit) sur « **Traget** » au niveau de la fenêtre « **project** » et ensuite sur « **Options for Traget ..** » (Figure 5)

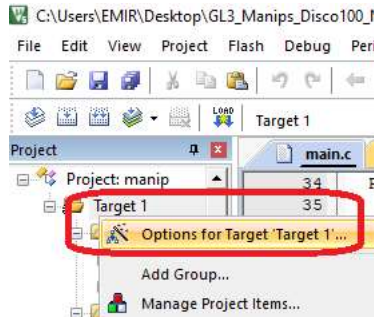


Figure 5

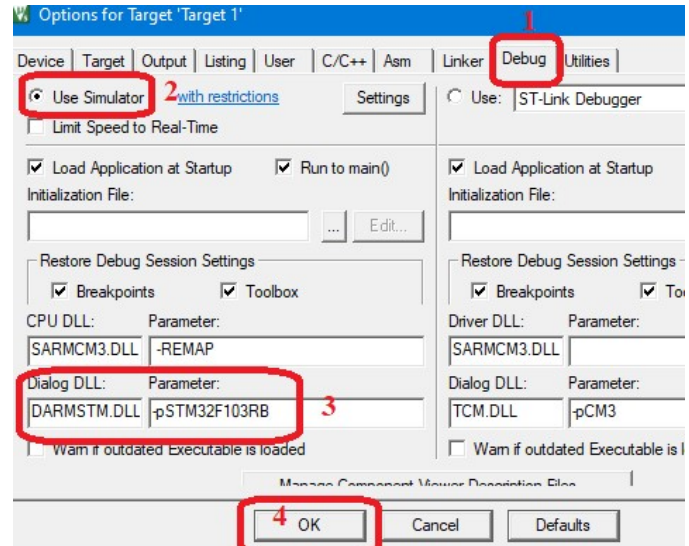


Figure 6

Au niveau de la fenêtre « Option for Target » : (Figure 6)

- 1- Choisir l'onglet « **Debug** ».
- 2- Sélectionner « **Use Simulator** ».
- 3- S'assurer des paramètres (Dialog DLL = **DARMSTM.DLL** , Parameter = **-pSTM32F103RB** )
- 4- Confirmer.

**Remarque** : Evidement, pour revenir au mode exécution sur la carte, il faut sélectionner « **use st link debugger** ».

Commencer par lancer le programme en mode « débogage ».



Pour simuler l'interface série UART (Figure-7) :

1. Sélectionner le menu « **View** »
2. Cliquer sur “**Serial Windows**”
3. Sélectionner l'interface UARTx à simuler (**UART #2**).
4. La fenêtre **UART #2** sera affichée en bas. Tout caractère envoyé sera affiché dans cette fenêtre.  
Tout caractère saisi au niveau de la fenêtre **UART #2** sera considéré comme reçu.

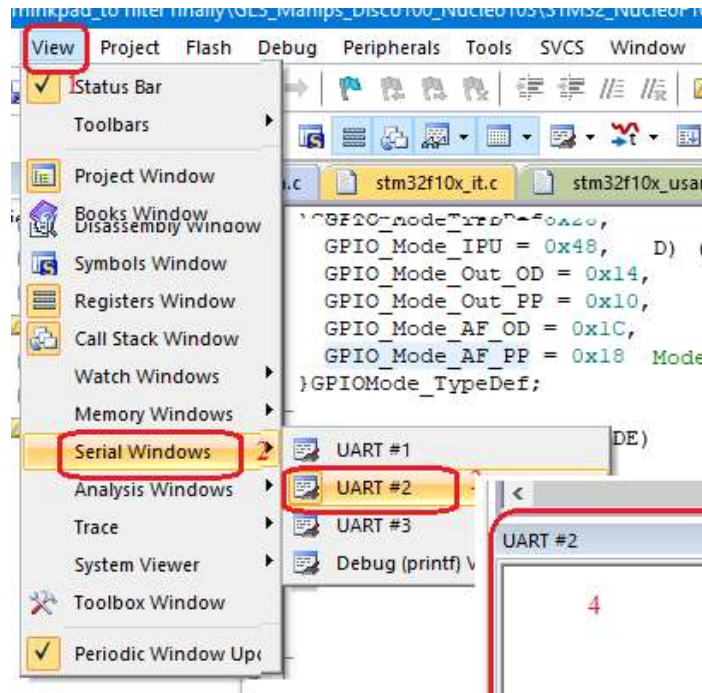


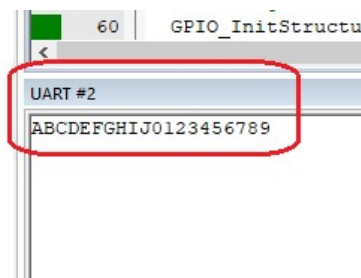
Figure-7

## Simulation de l'envoi

Lancer l'exécution de l'application (Run)



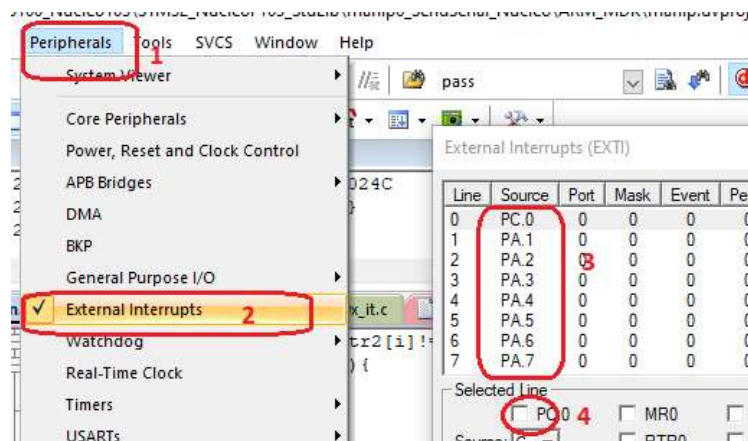
Dès le démarrage la 1<sup>ère</sup> chaîne de caractères devrait être affichée au niveau de la fenêtre **UART #2**



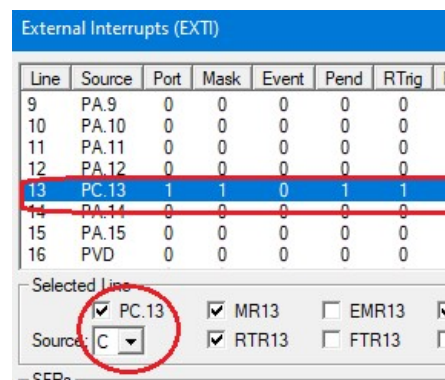


Pour envoyer la 2<sup>ème</sup> chaîne, il faut simuler une interruption.

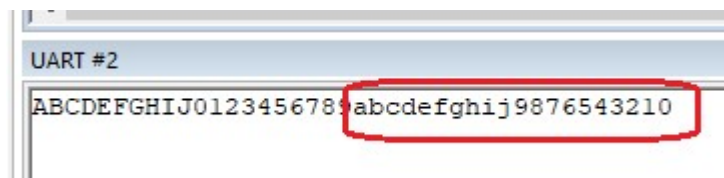
1. Sélectionner le menu « **Peripherals** »
2. Cliquer sur « **External Interrupts** »
3. Sélectionner le pin (source EXTI) à tester.
4. Il est possible de simuler une interruption en cochant le pin sélectionné (**selected line**).



Dans ce cas, il faut cocher la ligne PC13 pour simuler l'appui sur le bouton poussoir.

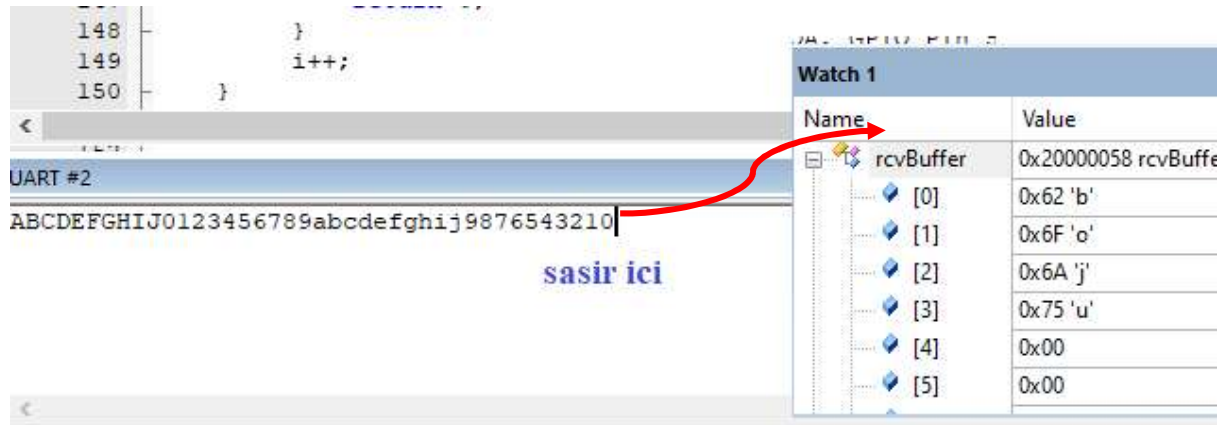


Le 2<sup>ème</sup> message sera affiché au niveau de la fenêtre **UART #2**



## Simulation de la réception

Pour simuler la réception d'un caractère, il faut juste saisir le caractère au niveau de la fenêtre UART#2, et observer la chaine rcvBuffer dans Watch Window.



The screenshot shows a simulation environment with a code editor on the left and a Watch Window on the right. The code editor displays lines 148, 149, and 150 of a program, with line 149 containing the statement `i++;`. The UART#2 window shows a sequence of characters: `ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz9876543210`. A red arrow points from the end of this sequence to the Watch Window. The Watch Window, titled "Watch 1", displays the variable `rcvBuffer` and its value `0x20000058 rcvBuffer`. Below this, the array elements are listed with their corresponding values:

Name	Value
<code>rcvBuffer</code>	<code>0x20000058 rcvBuffer</code>
<code>[0]</code>	<code>0x62 'b'</code>
<code>[1]</code>	<code>0x6F 'o'</code>
<code>[2]</code>	<code>0x6A 'j'</code>
<code>[3]</code>	<code>0x75 'u'</code>
<code>[4]</code>	<code>0x00</code>
<code>[5]</code>	<code>0x00</code>