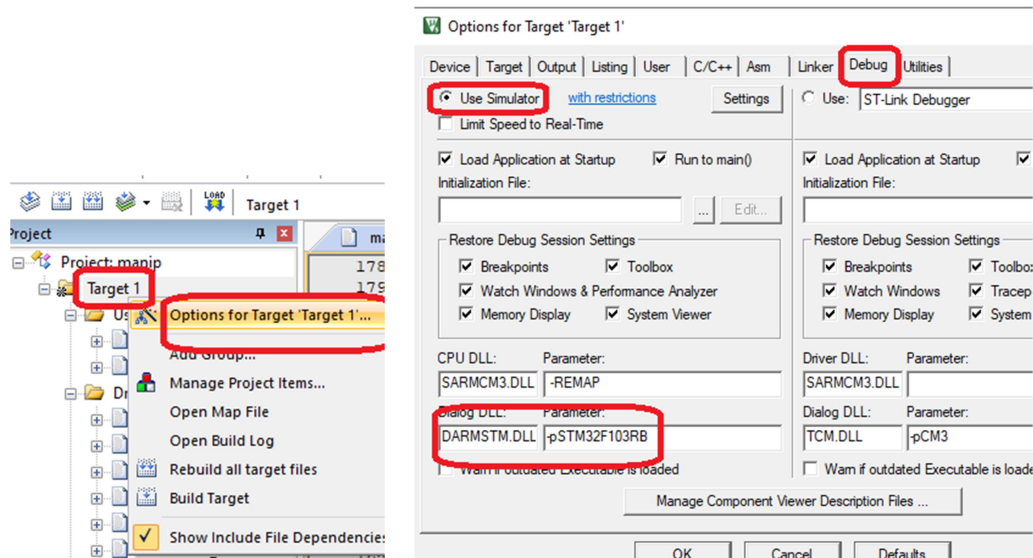


## Simulation

### A- Réception série par utilisation du DMA

(voir projet contenu dans le dossier **manip\_Rcv\_UART1\_DMA**)

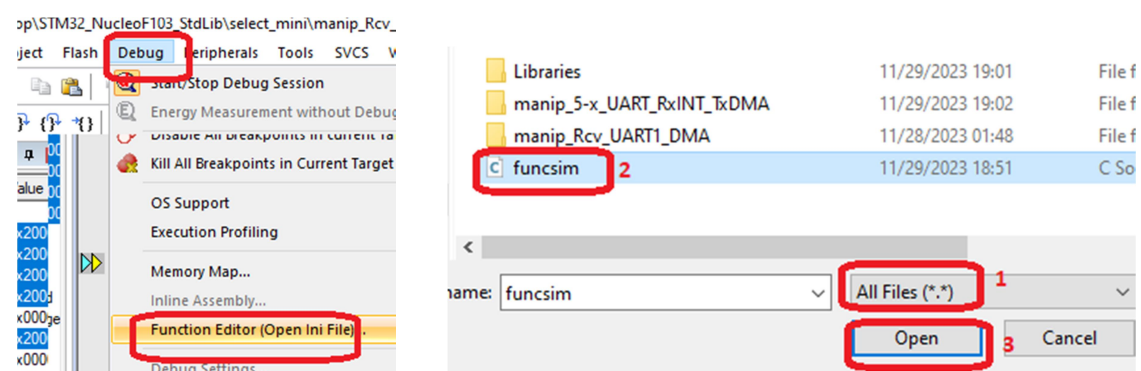
1-choisir le mode “use simulator” pour le débogage.



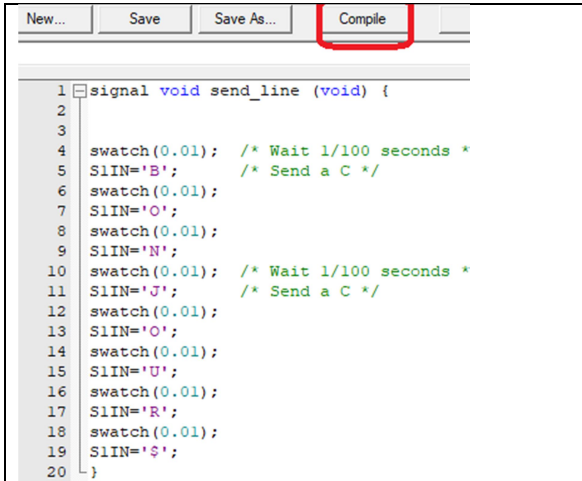
2- Lancer le débogage



3- Avant de lancer l’exécution du programme, ajouter une fonction qui permet de simuler l’envoi de données vers l’USART1 que notre application va recevoir et placer dans la chaîne **Receive\_buffer**.



4- Compiler la fonction. Il est maintenant possible de l'invoquer pour simuler un envoi de caractères vers UART1 (S1IN).

	<p>Cette fonction envoie la chaine « BONJOURS », caractère par caractère.</p> <p>Il faut noter qu'on ne peut envoyer un caractère à la fois et on ne peut déclarer des chaines de caractères avec le simulateur.</p> <p>Donc pour le mini-projet, il faut déclarer la séquence caractère par caractère</p>
---	--

Si le code est correct, la fenêtre « **Command** » affichera le code sans aucun code erreur.

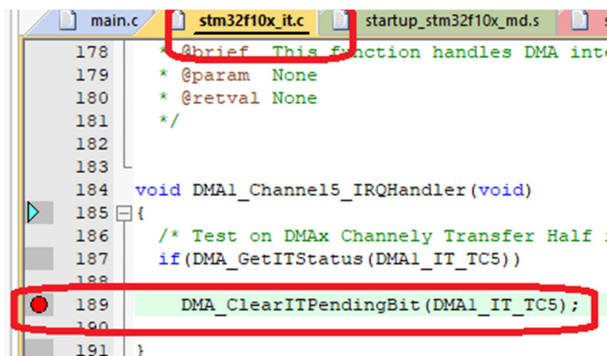
```

Command
signal void send_line (void) {

    swatch(0.01); /* Wait 1/100 seconds *.
    S1IN='B';      /* Send a C */
    swatch(0.01);
    S1IN='O';
    swatch(0.01);
    S1IN='N';
    swatch(0.01); /* Wait 1/100 seconds *.
    S1IN='J';      /* Send a C */
    swatch(0.01);
    S1IN='O';
    swatch(0.01);
    S1IN='U';
    swatch(0.01);
    S1IN='R';
    swatch(0.01);
    S1IN='$';
}

```

5- Placer un Break point au niveau du Handler (DMA\_IRQHandler) qui devrait être exécuté une fois 20 caractères reçus.

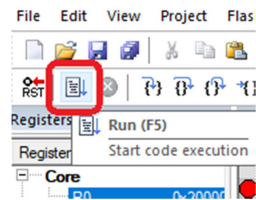


```

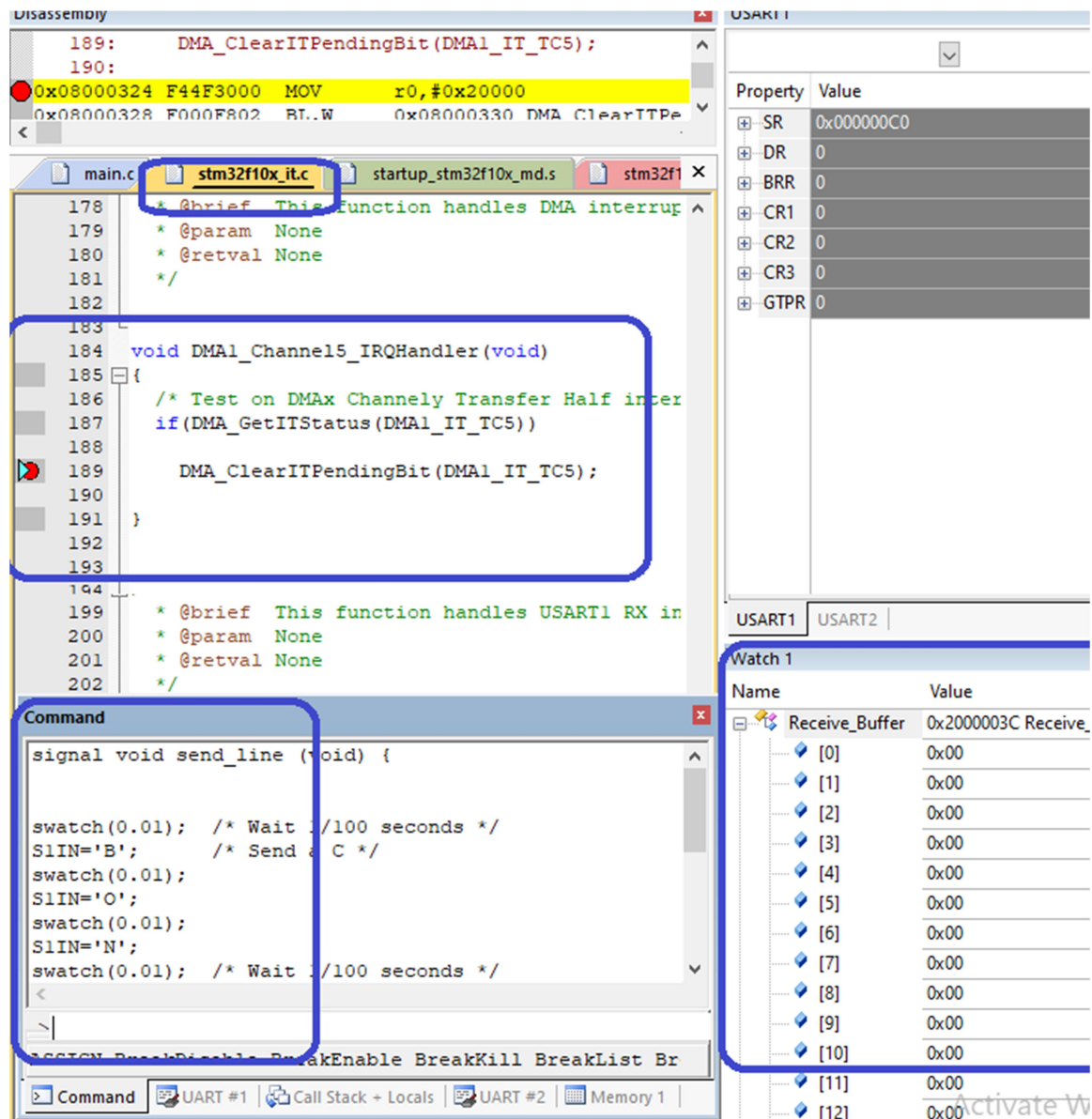
main.c  stm32f10x_it.c  startup_stm32f10x_md.s
178  /* @brief This function handles DMA interrupt request.
179  * @param None
180  * @retval None
181  */
182
183
184 void DMA1_Channel5_IRQHandler(void)
185 {
186     /* Test on DMAx Channely Transfer Half :
187     if(DMA_GetITStatus(DMA1_IT_TC5))
188     {
189         DMA_ClearITPendingBit(DMA1_IT_TC5);
190     }
191 }

```

## 6- Lancer l'exécution



## 7- faire apparaitre les fenêtres « Command » et « Watch »



The screenshot displays the IDE interface with several windows open:

- Disassembly:** Shows assembly code for the DMA1\_Channel5\_IRQHandler function. The instruction `MOV r0, #0x200000` is highlighted.
- Source:** Shows the C code for the DMA1\_Channel5\_IRQHandler function. The function is highlighted with a blue box.
- Command:** Shows the C code for the send\_line function. The function is highlighted with a blue box.
- Watch:** Shows the Receive\_Buffer array. The array is highlighted with a blue box.

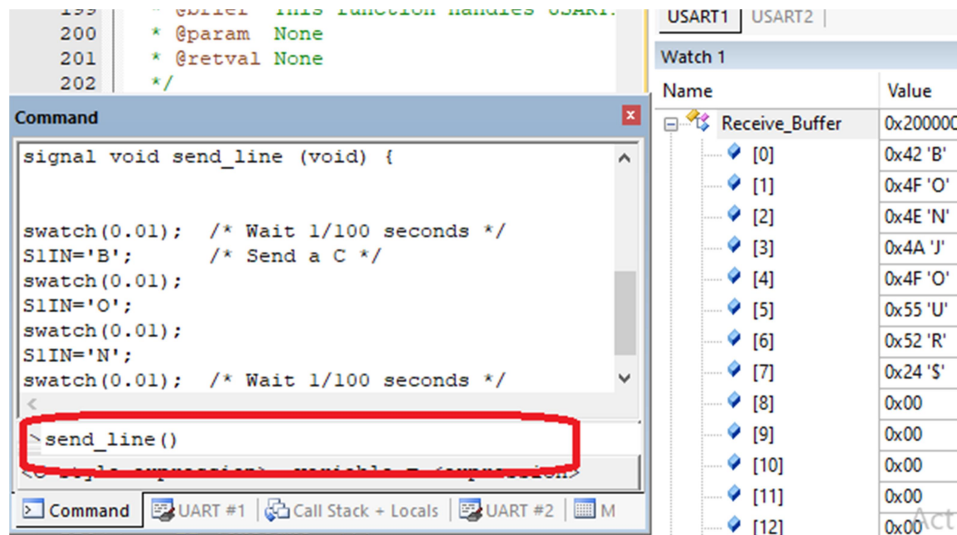
The Command window contains the following code:

```
signal void send_line (void) {  
  
    swtch(0.01); /* Wait 1/100 seconds */  
    S1IN='B'; /* Send a C */  
    swtch(0.01);  
    S1IN='O';  
    swtch(0.01);  
    S1IN='N';  
    swtch(0.01); /* Wait 1/100 seconds */  
}
```

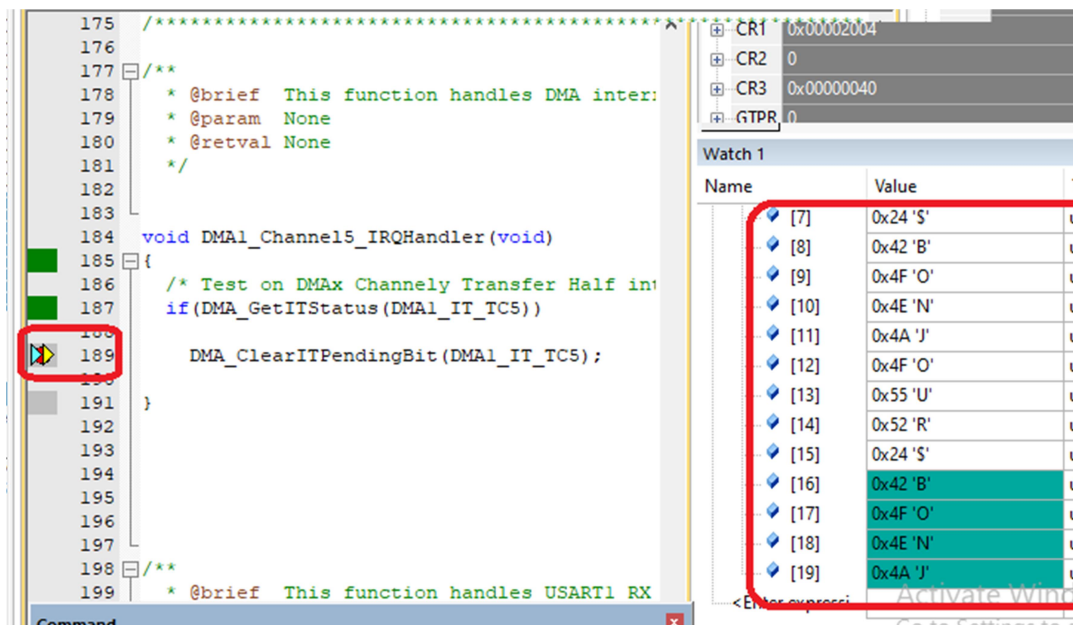
The Watch window shows the Receive\_Buffer array with the following values:

Name	Value
Receive_Buffer	0x2000003C Receive_
[0]	0x00
[1]	0x00
[2]	0x00
[3]	0x00
[4]	0x00
[5]	0x00
[6]	0x00
[7]	0x00
[8]	0x00
[9]	0x00
[10]	0x00
[11]	0x00
[12]	0x00

8- Au niveau de la ligne de commande dans la fenêtre « Command », saisir le nom de la fonction **send\_line()** + enter . Observer le contenu de la variable **receive\_Buffer**.



Après 3 exécutions de la fonction **send\_line ()** soit l'envoi de plus que 20 caractères, on trouvera que la variable **Receive\_Buffer** est pleine et que le **Breakpoint** (TC= Transfer complete) est atteint.

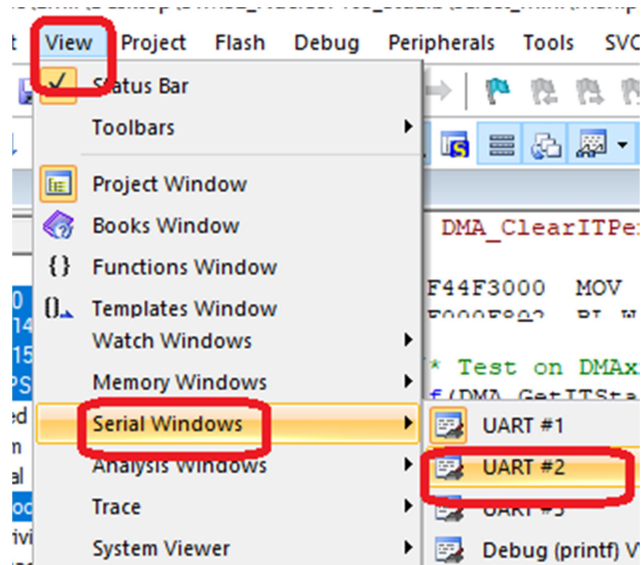


## B- Transmission série

Quand on simule la transmission, il n'y a aucune fonction à ajouter. Il suffit d'observer les données émises sur l'interface UART.

Par exemple, en transmettant « Bonjour » sur l'interface UART2.

- 1- Il faut afficher la fenêtre « **UART2** »



- 2- Exécuter le programme et observer la fenêtre **UART2**

