TP02_USART

1. Introduction

1.1 Objectifs :
On a besoin de relier une liaison série à un ordinateur hôte.

1.2 Présentation :
Afin de relier une liaison série à un ordinateur hôte, il est mieux que l'on utilise un port USB à l'aide un composant permettant de transformer les trames séries en paquets USB (convertisseur USB/série). L'adaptateur de débogage présent sur la carte Nucleo, est capable de jouer, en plus du débogage, le rôle d'un convertisseur USB/série. Cet adaptateur est relié sur la carte aux broches PA2 et PA3 du STM32. Ces deux broches peuvent être contrôlées par le périphérique USART2. Une fois configuré, il sera possible de communiquer avec l'hôte via cette liaison série.
La liaison série via PA2 (TX) et PA3 (RX) nécessite d'une part de configurer le GPIO pour ces 2 pins en mode 'alternate' en choisissant la fonction alternative N°7 (AF07) et d'autre part de configurer le mode de transmission de l'USART. La transmission choisie est une transmission asynchrone, sans parité, un bit de stop et sans contrôle de flux. On souhaite travailler à 115200 bauds.

2. Problématique

On doit faire l'initialisation et configurer le mode de transmission de l'USART et le GPIO et également écrire les fonctions pour réaliser transmettre les données correctement.

3. Solutions

3.1 Code :

```
19 #include "stm32f4xx.h"
20
21
22 extern void USART2_Init(uint32_t baud);//en sorte que l'on utilise les initialisation de l'USART2
23
24 GPIO_TypeDef *PA = GPIOA;
25 USART_TypeDef *UA = USART2;//type of struct
26
27
28 int _io_putchar(int ch)
29 {
30     while (!(UA->SR & USART_SR_TXE));//attendre le drapeau du registre d'état à 1
31     UA->DR = ch;                     //copier la donnée dans le registre de donnée
32     while (!(UA->SR & USART_SR_TC)); //vérifier que tous les bytes ont été transformées
33     return ch;                       //USART_SR_TXE, USART_SR_TC sont dans les registres d'état
34 }
35
36 int  __io_getchar(void)
37 {
38     int out = 0;
39
40     while (!(UA->SR & USART_SR_RXNE));//Lorsque ce drapeau est 1, une donnée est présente dans DR
41     out = UA->DR;                     //recevoir des données
42
43 //  while(UA->SR & USART_SR_RXNE);    //vérifier qu'il n'y pas de donnée restée mais pas utile
44
45     return out;
46 }
```

```c
void USART2_Transmit(uint8_t * data, uint32_t len)
{
    for(int i=0; i<len; i++)             //copier les données jusqu'à la fin
    {
        while (!(UA->SR & USART_SR_TXE));//attendre le drapeau du registre d'état à 1
        UA->DR = data[i];                //copier la donnée dans le registre de donnée
    }
}

int32_t USART2_Receive(uint8_t * data, uint32_t len)
{
    int32_t i = 0;
    int timeout = 10000000;              //mettre un time-out pour détecter les fins de trame.

    while((i<len) && (timeout>0))        //deux conditions obligatoires
    {
        while (!(UA->SR & USART_SR_RXNE))//Lorsque ce drapeau est 1, une donnée est présente dans DR
            timeout--;
        if(timeout > 0)
            data[i++] = UA->DR;          //recevoir des données
    }
    return (i-1);                        //renvoie le nombre de données effectivement reçues.
}

int main(void)
{
    uint8_t *data = "hello world";//une trame à envoyer
    uint32_t len = 11;            //le longue de la trame

    USART2_Init(115200);          //initialisation de l'USART2 : baud
    //génération de la clock
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN| RCC_APB1ENR_USART2EN;

    //pin PA2 en sortie
    PA->MODER &= ~GPIO_MODER_MODER2_Msk;
    PA->MODER |= GPIO_MODER_MODER2_1;//utiliser ..._1 parce que (0x2UL << GPIO_MODER_MODER2_Pos)
                                     //Alternate function mode:  10  (reference manual)
    //pin PA3 en sortie
    PA->MODER &= ~GPIO_MODER_MODER3_Msk;
    PA->MODER |= GPIO_MODER_MODER3_1;

    /* Connect  pins to Alternate function */
    PA->AFR[0] |= (7 << 8); //AFL0-AFL7 correspond à GPIOx0-GPIOx7
    PA->AFR[0] |= (7 << 12);


    _io_putchar('4');//on doit utiliser '' pour entrer un integer ici

    __io_getchar();

    USART2_Transmit(data, len);

    USART2_Receive(data, len); //les fonctions demandées

    return 0;

}
```

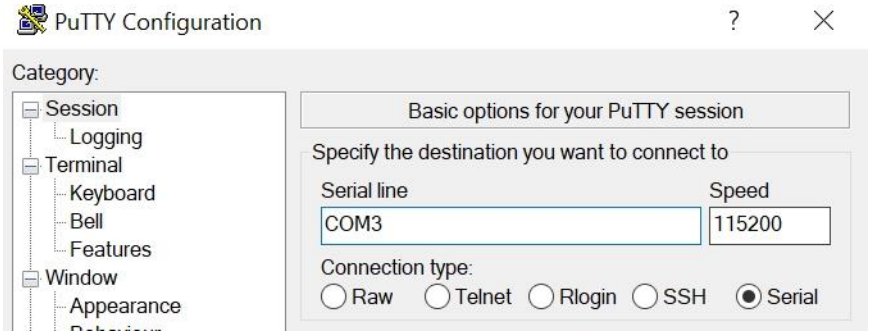
3.2 Exécution :

3.3 Démarches :
On généralise la clock et après configurer les deux pins utilisés : PA2(TX), PA3(RX).
On doit également les adapter en mode 'alternate' afin de faire la transmission.
La transmission choisie est une transmission asynchrone, sans parité, un bit de stop et sans contrôle de flux. L'une des étapes importante est de saisir la valeur de Baud : 115200. L'initialisation de l'USART2 a été fait par le professeur dans le fichier «usart.c» qui nous permet d'utiliser
extern void USART2_Init(uint32_t baud).

Les quatre fonctions utilisées sont :
int __io_putchar(int ch)
// cette fonction prend en paramètre un caractère et renvoie le même caractère (sous format int).
void USART2_Transmit(uint8_t * data, uint32_t len)
// data est l'adresse de la chaîne et len le nombre de données à transmettre.
int __io_getchar(void)
// La fonction renverra le caractère reçu sur l'octet de poids faible de l'entier.
int32_t USART2_Receive(uint8_t * data, uint32_t len)
// la fonction renvoie le nombre de données effectivement reçues.

Les trois drapeaux utilisés sont :
USART_SR_TXE
//c'est le drapeau du registre d'état. Lorsqu'il passe à 1, on met la donnée dans le registre de donnée.
USART_SR_TC
//c'est le drapeau du registre d'état. Lorsqu'il passe à 1, tous les bits ont été transmis.
USART_SR_RXNE(registre de donnée réception Not Empty)
// c'est le drapeau du registre d'état. En réception, le logiciel attend qu'une donnée soit reçue en scrutant le drapeau RXNE du registre d'état.
Lorsque ce drapeau est à 1, une donnée non lue est présente dans DR et le logiciel peut la récupérer.
Il est à noter que la lecture de registre de donnée remet le bit RXNE à 0.

En fin, on utilise un terminal (putty) sur la liaison série avec les paramètres 115200, 8, N, 1 afin de vérifier que les données sont correctement transmises au terminal hôte.



4. Conclusion

Il est indispensable de configurer les pins PA2 et PA3 en mode 'alternate' et on peut trouver l'adresse correspondant dans le fichier «stm32f401xe.h». Il est également important d'apprendre d'utiliser «reference manual» pour identifier GPIO port mode registre par exemple.
Les résultats dans le terminal s'affichent correctement. Grâce à ce TP, je sais comment écrire les fonctions pour relier une liaison série à un ordinateur hôte et apprends les drapeaux nécessaires pour la configuration.