



Compte Rendu TP :

Bas Niveau



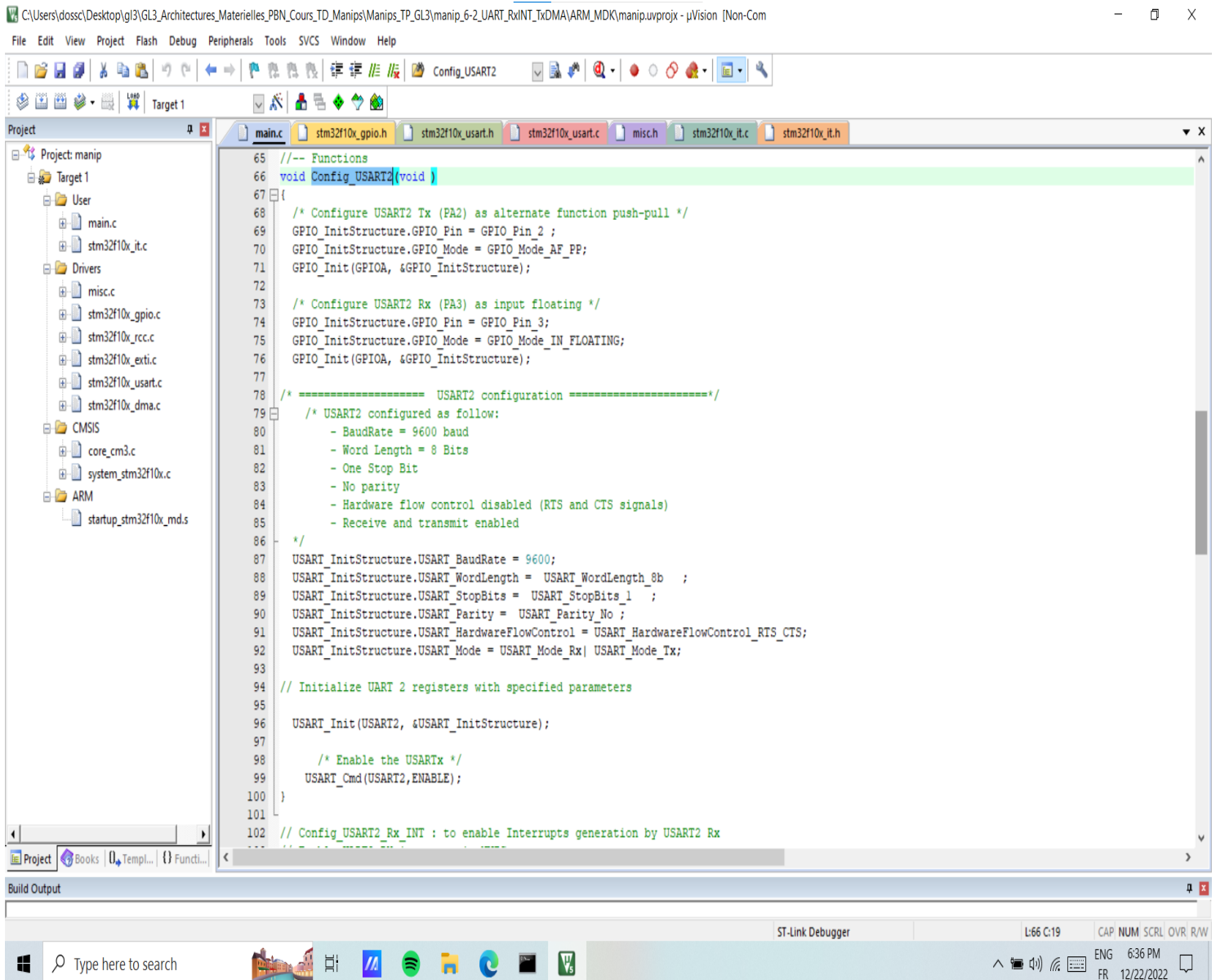
Doss Chiraz
Seddik Farah

GL 3/2

I. La manip 6_2 :

1- Manip 5 (1 ère partie):

1) Compléter les 2 fonctions (Config USART2 et Config_USART2_Rx_INT) au niveau de main.c:



```

}
// Config_USART2_Rx_INT : to enable Interrupts generation by USART2 Rx
// Enable UART2_RX interrupt in NVIC
void Config_USART2_Rx_INT (void)
{
    /*Enable Rx Interrupt (s) on USART2*/
    USART_ITConfig(USART2, USART_IT_RXNE,ENABLE);

    /* Enable and set USART2 Interrupt */
    NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 3;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}

```

2) Compléter le traitement à assurer au niveau du Handler relatif à l'interruption série (USART2_IRQHandler) dans le cas où les 2 chaînes sont différentes (stm32f10x_it.c).

```

216  /*
217  void USART2_IRQHandler(void)
218  {
219      /* Test on USART2 RX Interrupt */
220      if(USART_GetITStatus(USART2, USART_IT_RXNE))
221      {
222          /* Code a ajouter pour assurer le fonctionnement désiré */
223
224          if(i<=9)
225          {
226              Receive_Buffer[i]=USART2->DR;
227              i++;
228          }
229          else if (i==9)
230          {
231              i=0;
232              if (stringCompare(Receive_Buffer,password,10) ==0)
233              {
234                  GPIO_SetBits( GPIOA, GPIO_Pin_2);
235                  Delay(0xFFFFF);
236              }
237              else
238              {
239                  GPIO_ResetBits( GPIOA, GPIO_Pin_2);
240              }
241          }
242      }
243  }
244  }
245

```

2-Manip 5 (2ème partie):

- 1) Préciser le canal DMA à utiliser:

→ le canal 7

- 2) Compléter la fonction Config_USART2_Tx_DMA() qui permet de configurer le transfert DMA:

```

116
117 // Config_USART2_Tx_DMA : Enable DMA Transfer from Mem to USART2
118 // Enable and configure TC (Transfer Complete) Interrupt
119 void Config_USART2_Tx_DMA (void)
120 {
121     /*Enable Tx DMA Request (s) on USART2*/
122     USART_DMACmd(USART2,USART_DMAReq_Tx, ENABLE);
123
124     /* ===== DMA configuration ===== */
125     /*DMAx channelx (UART2 TX) configuration */
126     DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t) &(USART2->DR);
127     DMA_InitStructure.DMA_MemoryBaseAddr = (uint32_t) Transmit_Buffer;
128     DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralDST;
129     DMA_InitStructure.DMA_BufferSize = 10;
130     DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
131     DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable
132     DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
133     DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
134     DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
135     DMA_InitStructure.DMA_Priority = DMA_Priority_VeryHigh;
136     DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
137
138     //Initilize DMA Channel Registers
139     DMA_Init(DMA1_Channel7, &DMA_InitStructure );
140
141     /* Enable DMAx Channely Transfer Complete/ Half Transfer interrupts */
142     DMA_ITConfig(DMA1_Channel7, DMA_IT_TC, ENABLE);
143
144
145     /* Enable and set DMAx Channel y Interrupt */
146     NVIC_InitStructure.NVIC_IRQChannel = DMA1_Channel7_IRQn;
147     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 2;
148     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;
149     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
150     NVIC_Init(&NVIC_InitStructure);
151 }
152

```

- 3) Dé-commenter et compléter le code du Handler relatif aux interruptions DMA (stm32f10x_it.c) de telle sorte à allumer la LED si la transmission est effectuée.

```

181  /*
182  void DMA1_Channel7_IRQHandler(void)
183  {
184      /* Test on DMAx Channely Transfer Complete interrupt */
185      if(DMA_GetITStatus(DMA1_IT_TC7))
186      {
187          /* Code a ajouter pour assurer le fonctionnement désiré */
188          GPIO_SetBits(GPIOA,GPIO_Pin_5);
189          /* Clear DMAx Channely TC interrupt pending bit*/
190          DMA_ClearITPendingBit(DMA1_IT_TC7);
191      }
192
193      /* Test on DMAx Channely Transfer Half interrupt */
194      if(DMA_GetITStatus(DMA1_IT_HT7))
195      {
196          /* Code a ajouter pour assurer le fonctionnement désiré */
197
198          /* Clear DMAx Channely HT interrupt pending bit */
199          DMA_ClearITPendingBit(DMA1_IT_HT7);
200      }
201  }
202  }

```

- 4) Compléter le traitement à assurer au niveau du Handler relatif à l'interruption série (USART2_IRQHandler) dans le cas où les 2 chaînes sont identiques (lancer le transfert DMA)

```

214  /*
215  void USART2_IRQHandler(void)
216  {
217      /* Test on USART2 RX Interrupt */
218      if(USART_GetITStatus(USART2, USART_IT_RXNE))
219      {
220          if(i<10){
221              Receive_Buffer[i]=USART2 ->DR;
222              i++;
223          }
224          if(i==10)
225          {
226              i=0;
227              if (stringCompare(Receive_Buffer,password,10) ==0){
228                  GPIO_SetBits (GPIOA, GPIO_Pin_5);
229                  Delay(0xffffffff);
230                  GPIO_ResetBits(GPIOA,GPIO_Pin_5);}
231              else{
232                  DMA_Cmd(DMA1_Channel7,ENABLE);
233              }
234          }
235          USART_ClearITPendingBit(USART2,USART_IT_RXNE);
236      }
237  }
238  }
239  }

```

3- Simulation :

C:\Users\doss\c\Desktop\gl3\GL3_Architectures_Materielles_PBN_Cours_TD_Manip\Manips_TP_GL3\manip_6-2_UART_Rx\TxDMA\ARM_MDK\manip.uvprojx - µVision [Non-Com]

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

receive_buffer

Project: manip Target 1

User

main.c

stm32

Drivers

misc

stm32

stm32

stm32

stm32

stm32

stm32

CMSIS

core

Registers Project

Disassembly

```

0x08000AB4 BF00 NOP
0x08000AB6 E7FE B 0x08000AB6
35: RCC_APB2PeriphClockCmd(RCC_APB2Periph
36:
37: /* Enable USART2 clocks */
0x08000AB8 2101 MOVS r1,#0x01
0x08000ABA 2015 MOVS r0,#0x15
0x08000ABC F7FFDB0 BL.W 0x08000620 RCC_AP
38: RCC_APB1PeriphClockCmd(RCC_APB1Periph

```

main.c

```

34 /* Enable GPIOx and AFIO clocks */
35 RCC_APB2PeriphClockCmd(RCC_APB2Periph
36
37 /* Enable USART2 clocks */
38 RCC_APB1PeriphClockCmd(RCC_APB1Periph
39
40 /* Enable DMA1 clock */

```

main.c

```

34 /* Enable GPIOx and AFIO clocks */
35 RCC_APB2PeriphClockCmd(RCC_APB2Periph
36
37 /* Enable USART2 clocks */
38 RCC_APB1PeriphClockCmd(RCC_APB1Periph
39
40 /* Enable DMA1 clock */

```

DMA1

Property	Value
ISR	0
IFCR	0
CCR1	0
CNDR1	0
CPAR1	0
CMAR1	0
CCR2	0
CNDR2	0
CPAR2	0
CMAR2	0

Watch 1

Name	Value	Type
Receive_Buffer	0x20000040 Receive_B...	uchar[10]
[0]	0x68 'h'	uchar
[1]	0x65 'e'	uchar
[2]	0x6C 'l'	uchar
[3]	0x6C 'l'	uchar
[4]	0x6F 'o'	uchar
[5]	0x77 'w'	uchar
[6]	0x6F 'o'	uchar
[7]	0x6C 'l'	uchar
[8]	0x64 'd'	uchar
[9]	0x00	uchar
Receive_Buffer[10]	0x00	uchar
<Enter expression>		

UART #2

GPIOA

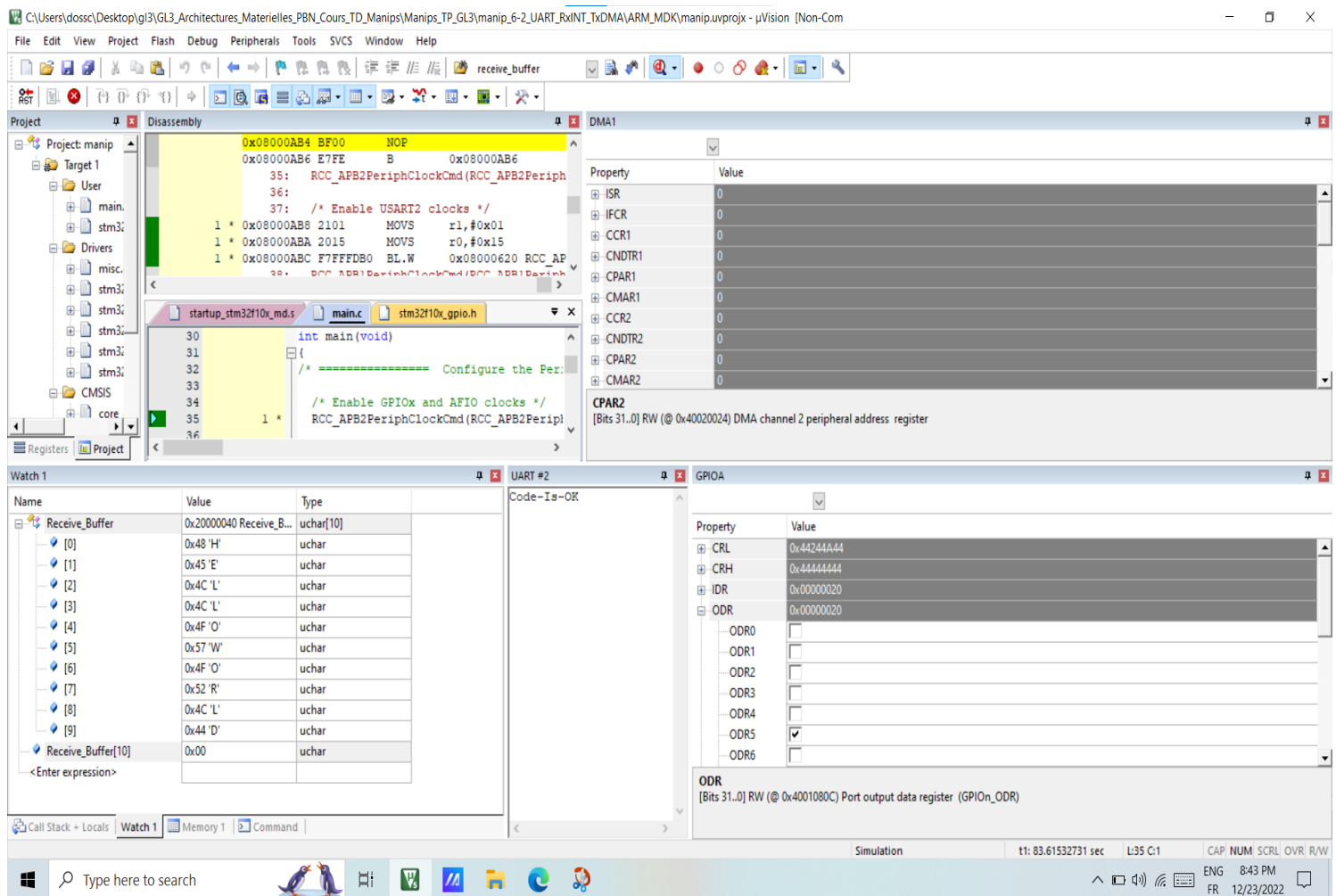
Property	Value
CRL	0x44244A44
CRH	0x44444444
IDR	0
ODR	0
BSRR	0
BRR	0
LCKR	0

Simulation

t1: 12.84022283 sec L:35 C:1 CAP: NUM SCRL OVR: R/W

ENG 8:44 PM

FR 12/23/2022



II. Serial 00

1)

on aura besoin de deux fichiers:

-le fichier header "Serial.h" qui contiendra les déclarations de la classe Serial et les prototypes de fonction; la méthode "send", le constructeur de Serial [Serial(uint32_t txPin, uint32_t rxPin);] et les attributs "TxPin" et "RxPin". (Ce fichier sera inclus dans tout fichier source qui utilise la classe Serial.)

-le fichier source "Serial.c" qui contiendra le constructeur des objets de type Serial [Serial::Serial(uint32_t txPin, uint32_t rxPin)] ,et la définition de fonctions de classe: "send" . (Ce fichier doit inclure le fichier "Serial.h").

2) Code:

```

//constructeur
Serial(PinName TxPin,PinName RxPin) {
    GPIO_InitTypeDef GPIO_InitStruct;

    /* Enable USART2 clocks */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);

    /* Configure USART2 Tx (PA2) as alternate function push-pull and
    Configure USART2 Rx (PA3) as input floating */

    GPIO_InitStruct.GPIO_Pin= 1<<( TxPin & 0x0F) ;
    GPIO_InitStruct.GPIO_Speed= GPIO_Speed_50MHz;
    GPIO_InitStruct.GPIO_Mode= GPIO_Mode_AF_PP;
    GPIO_Init( GPIOA, &GPIO_InitStruct);

    GPIO_InitStruct.GPIO_Pin= 1<<( RxPin & 0x0F) ;
    GPIO_InitStruct.GPIO_Mode= GPIO_Mode_IN_FLOATING;
    GPIO_Init( GPIOA, &GPIO_InitStruct);


    USART_InitTypeDef  USART_InitStructure;

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);

    /* USART2 configuration */
    USART_InitStructure.USART_BaudRate= 9600;
    USART_InitStructure.USART_WordLength=USART_WordLength_8b;
    USART_InitStructure.USART_StopBits=USART_StopBits_1;
    USART_InitStructure.USART_Parity=USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl=
    USART_HardwareFlowC ontrol_None;
    USART_InitStructure.USART_Mode= USART_Mode_Tx | USART_Mode_Rx;
    USART_Init( USART2, & USART_Init Structure);

    USART_Cmd( USART2, ENABLE);
}

```

```
// fonction send
send(char *data)
{
    int length = strlen(data);
    for (int i = 0; i < length; i++) {
        USART_SendData(USART2, data[i]);
    }
}
```