EIE3810 Microprocessor System Design Laboratory

Laboratory Report #2

Name: 马腾飞

Student ID: 121090406

Date: Oct 13,2023

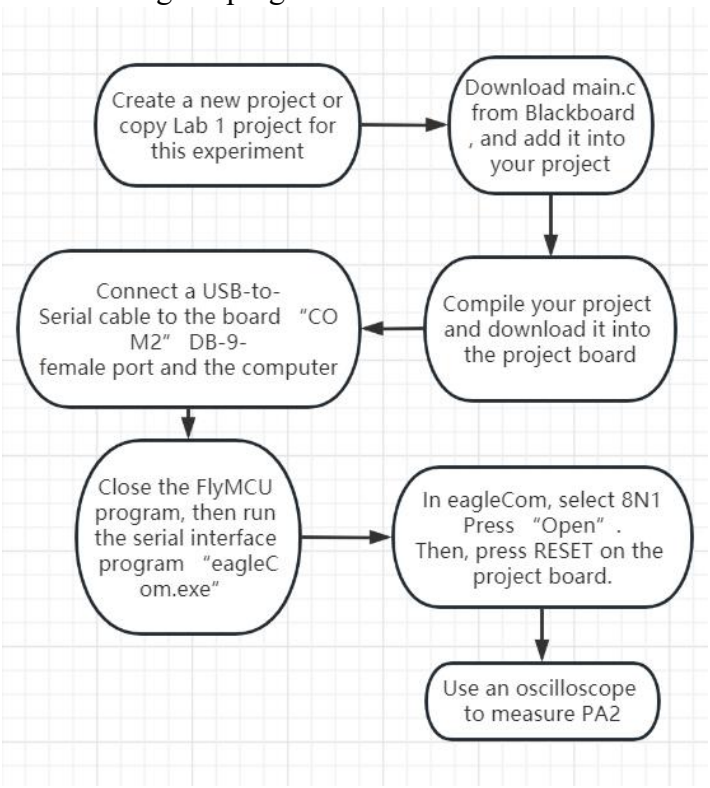
The Chinese University of Hong Kong, Shenzhen

- Experiment 1:Setup USART2 to 9600bps, 8N1 and transmit “AB”
- Experiment 2: Setup USART1 to 9600bps, 8N1 with 72 MHz and transmit student ID
- Experiment 3: Send a string to USART with checking TXE bit
- Experiment 4: Create CLOCK and USART libraries for the project board and transmit “1234567890” and student ID

1. Experiment 1

1.1 Design

1.1.1 Designed program flowchart



1.1.2 Explanation of source code

```
int main(void)
{
  EIE3810_clock_tree_init();
  EIE3810_USART2_init(36, 9600);
  //USART_print(1,"1234567890"); //This line will be used in Experiment 3
  while(1)
  {
    USART2->DR = 0x41;//put A (0100 0001) on data register of Universal Sy
    Delay(50000);
    USART2->DR = 0x42;//put B (0100 0010) on data register of USART2
    Delay(50000);
    Delay(1000000);
  }
}
```

Transmit A and B alternatively. And it is necessary to choose proper delay.

1.1.2.1 Describe the procedures of subroutine “EIE3810_clock_tree_init()”

```
void EIE3810_clock_tree_init(void)
{
  u8 PLL=7;//0111 -> PLL input clock * 9
  u8 temp=0;
  RCC->CR |= 0x00010000; //Enable High Speed External oscillator: set bit16 of control register as 1
  while(!((RCC->CR>>17)&0x1));//wait for HSE oscillator ready
  RCC->CFGR &= 0xFFFDFFFF; //HSE clock is not divided
  RCC->CFGR |= 1<<16; //HSE oscillator clock selected as PLL input clock
  RCC->CFGR |= PLL<<18; //PLL input clock * 9
  RCC->CR |=0x01000000;//Enable PLL
  while(!(RCC->CR>>25));//wait for PLL locked: be stable
  RCC->CFGR &=0xFFFFFFFE;//10->PLL selected as system clock
  RCC->CFGR |=0x00000002;//PLL selected as system clock
  while(temp !=0x02) //check if system clock switch status is "10:PLL used as system clock"
  {
    temp=RCC->CFGR>>2;
    temp &= 0x03; //Assign system clock switch status to temp
  }
  RCC->CFGR &= 0xFFFFFC0F;//SYSCLK not divided
  RCC->CFGR |= 0x00000400;//HCLK divided by 2
  FLASH->ACR = 0x32;//Set FLASH with 2 wait states
  RCC->APB1ENR |= 1<<17; //Enable USART2
}
```

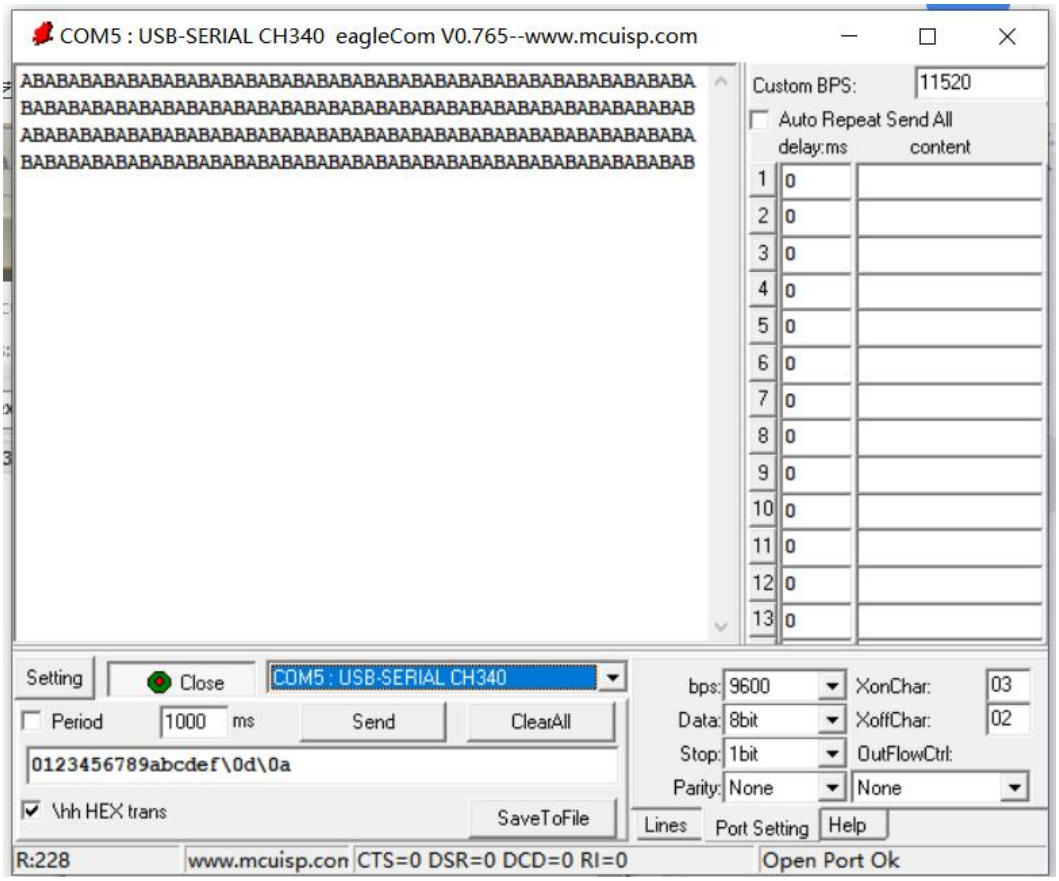
Enable HSE and use PLL to output a clock with nine times frequency i.e. 9x8=72(MHz) Select PLL as system clock, and divide HCLK by 2. Set FLASH with 2 wait states to deal with too quick frequency. In the end, enable USART2.

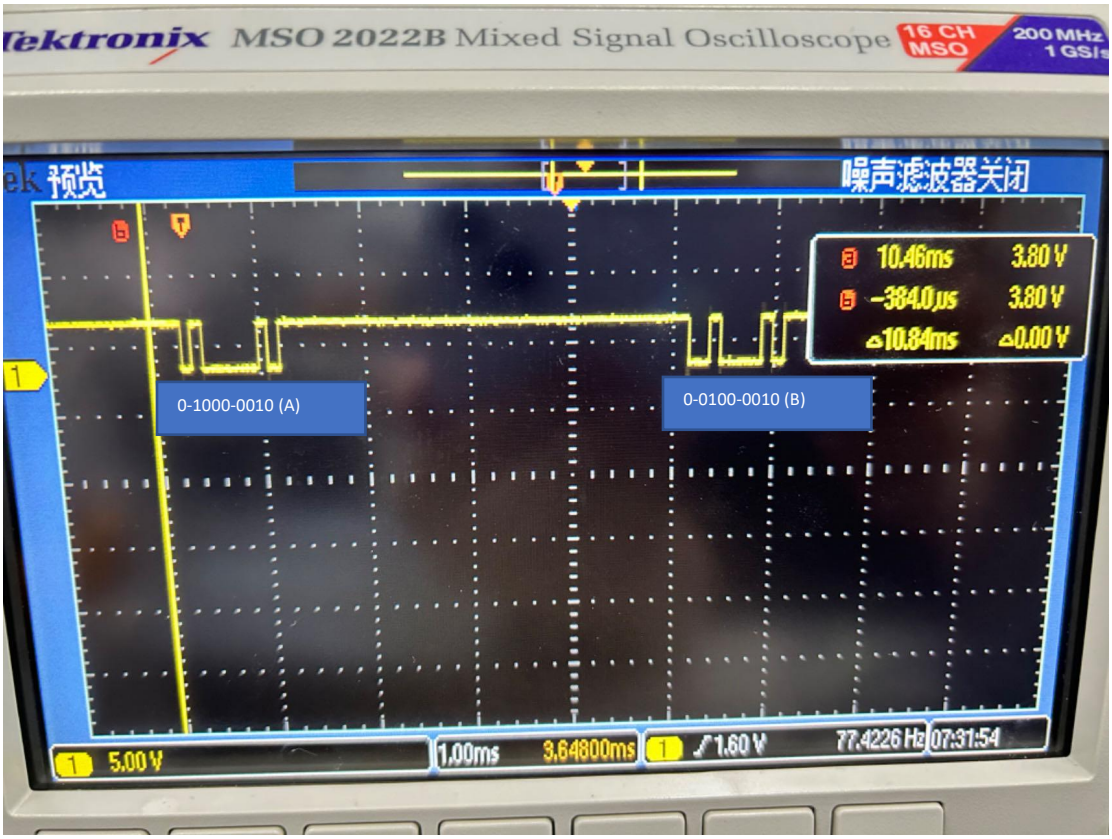
1.1.2.2 Describe the procedures of subroutine “EIE3810_USART2_init()”

```
void EIE3810_USART2_init(u32 pclk1, u32 baudrate)
{
  //USART2
  float temp;
  u16 mantissa;
  u16 fraction;
  temp=(float) (pclk1*1000000)/(baudrate*16);
  mantissa=temp;
  fraction=(temp-mantissa)*16;
  mantissa<<=4;
  mantissa+=fraction; //calculate divider
  RCC->APB2ENR |= 1<<2; //reset IO port A
  GPIOA->CRL &= 0xFFFF00FF; //PA2 max speed 50MHz,Alternate Function Output Push-pull;
                            //PA3 Input with pull-up/pull-down
  GPIOA->CRL |= 0x00008B00; //PA2 max speed 50MHz,Alternate Function Output Push-pull;
                            //PA3 Input with pull-up/pull-down
  RCC->APB1RSTR |= 1<<17; //reset USART2
  RCC->APB1RSTR &= ~(1<<17); //No effect
  USART2->BRR=mantissa;//the USART Divider
  USART2->CR1=0x2008; //bit0 = 0:no break character is transmitted
                      //bit1 = 0:Receiver in active mode
                      //bit2 = 0:Receiver is disabled
                      //bit3 = 1:Transmitter is enabled
                      //bit4 = 0:IDLE Interrupt is inhibited
                      //bit5 = 0:RXNE Interrupt is inhibited
                      //bit6 = 0:Transmission complete interrupt is inhibited
                      //bit7 = 0:TXE interrupt is inhibited
                      //bit8 = 0:PE interrupt is inhibited
                      //bit9 = 0:Even parity
                      //bit10 = 0:Parity control disabled
                      //bit11 = 0:Idle Line
                      //bit12 = 0:Start bit, 8Data bits,n Stop bit
                      //bit13 = 1:USART enabled
```

First, calculate USART Divider, and then set PA2 as Alternate Function output push-pull, PA3 as pull-up input. Reset USART2 and set USART Divider as the value calculated. In the end, set Transmit mode.

1.2 Result



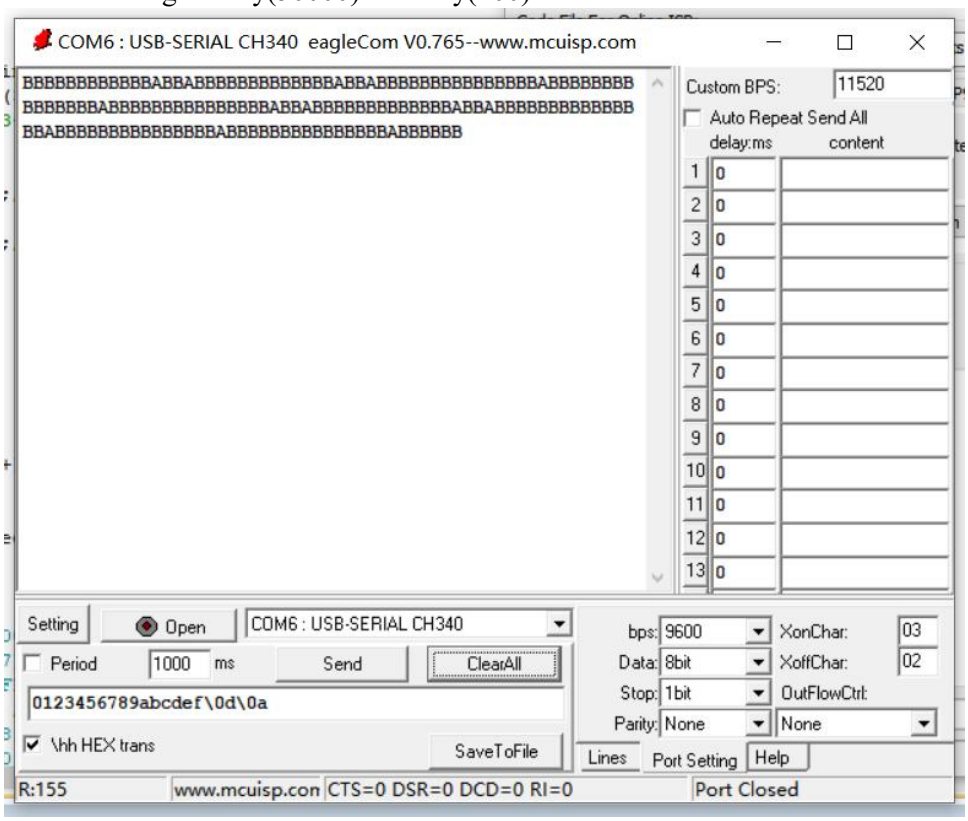


1.2.1

ASCII of A is 0100 0001, ASCII of B is 0100 0010. When transmitting, the data is reversed due to Little Endian. There will be a stop bit 0 at the end of every 8 bits.

Look at the marks on the figure, there are longer low level between two high level in the left wave. So the left wave is A, and the right wave is B.

1.2.2 Change Delay(50000) to Delay(100)




```
USART2->DR = 0x41;//put A (0100 0001)
Delay(100);
USART2->DR = 0x42;//put B (0100 0010)
Delay(100);
Delay(1000000);
```

You will see there are many B but few A. Because the delay between “transmit A” and “transmit B” is too short, which leads to covering A with B. But there are long delay after transmitting B.

1.3 Questions

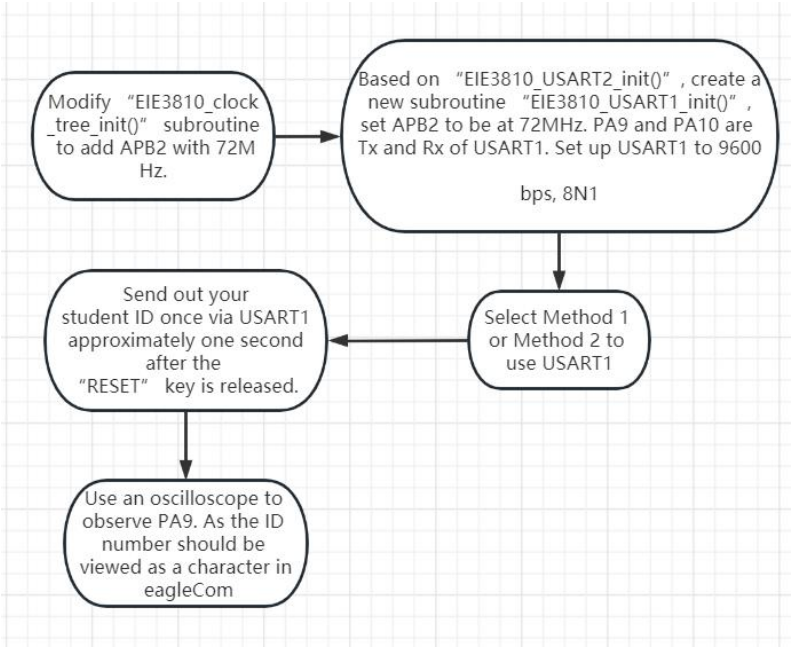
1.3.1 Why is there an error of serial communication in step 8?

Because the delay between “transmit A” and “transmit B” is too short, which leads to covering A with B. Namely, we just put A in the Data Register and “A” has not sent, while “B” is already put on Data Register. So “B” can cover “A”. But there are long delay after transmitting B (like Delay(1000000) in the code). Therefore, B can not be covered by A.

2. Experiment 2

2.1 Design

2.1.1 Designed program flowchart



2.1.2 Explanation of source code

```
int main(void)
{
  EIE3810_clock_tree_init();
  EIE3810_USART1_init(72, 9600);
  //USART_print(1,"1234567890"); //This line will b

  Delay(10000000);
  USART1->DR = 0x31;//1 on data register of USART1
  Delay(50000);
  USART1->DR = 0x32;//2
  Delay(50000);
  USART1->DR = 0x31;//1
  Delay(50000);
  USART1->DR = 0x30;//0
  Delay(50000);
  USART1->DR = 0x39;//9
  Delay(50000);
  USART1->DR = 0x30;//0
  Delay(50000);
  USART1->DR = 0x34;//4
  Delay(50000);
  USART1->DR = 0x30;//0
  Delay(50000);
  USART1->DR = 0x36;//6
  Delay(50000);


}
```

In main function, first set clock tree and then initialize USART1.(discuss details later)
Secondly, we send my student ID and use some delay between them to make sure that every number is received by computer.

```
void EIE3810_clock_tree_init(void)
{
  u8 PLL=7;//0111 -> PLL input clock * 9
  u8 temp=0;
  RCC->CR |= 0x00010000; //Enable High Speed External oscillator: set bit16 of control register as 1
  while(!((RCC->CR>>17)&0x1));//wait for HSE oscillator ready
  RCC->CFGR &= 0xFFFDFFFF; //HSE clock is not divided
  RCC->CFGR |= 1<<16; //HSE oscillator clock selected as PLL input clock
  RCC->CFGR |= PLL<<18; //PLL input clock * 9
  RCC->CR |=0x01000000;//Enable PLL
  while(!(RCC->CR>>25));//wait for PLL locked: be stable
  RCC->CFGR &=0xFFFFFFFE;//10->PLL selected as system clock
  RCC->CFGR |=0x00000002;//PLL selected as system clock
  while(temp !=0x02) //check if system clock switch status is "10:PLL used as system clock"
  {
    temp=RCC->CFGR>>2;
    temp &= 0x03; //Assign system clock switch status to temp
  }
  RCC->CFGR &= 0xFFFFFC0F;//SYSCLK not divided
  RCC->CFGR |= 0x00000400;//HCLK divided by 2
  FLASH->ACR = 0x32;//Set FLASH with 2 wait states
  RCC->APB1ENR |= 1<<17; //Enable USART2
  RCC->APB2ENR |= 1<<14; //Enable USART1 -
}
```

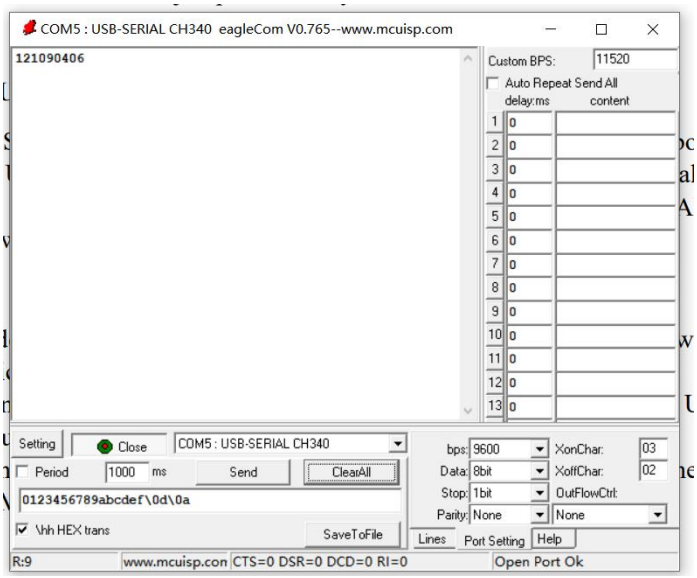
Enable HSE and use PLL to output a clock with nine times frequency i.e. 9x8=72(MHz)

Select PLL as system clock, and divide HCLK by 2. Set FLASH with 2 wait states to deal with too quick frequency. In the end, enable USART1 and USART2.

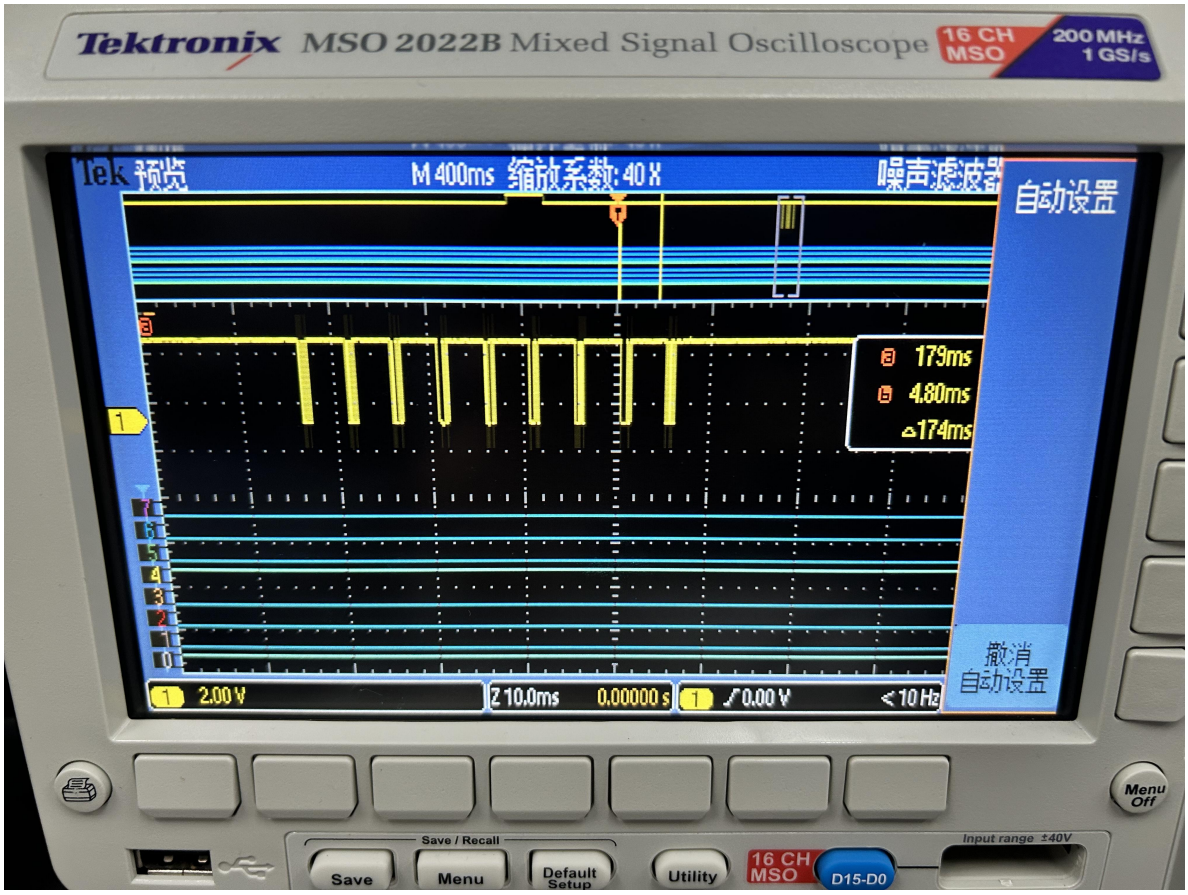
```
void EIE3810_USART1_init(u32 pclk1, u32 baudrate)
{
  //USART2
  float temp;
  u16 mantissa;
  u16 fraction;
  temp=(float) (pclk1*1000000)/(baudrate*16);
  mantissa=temp;
  fraction=(temp-mantissa)*16;//separate USARTDIV into two parts -
  mantissa<<=4;
  mantissa+=fraction;
  RCC->APB2ENR |= 1<<2; //reset IO port A
  GPIOA->CRH &= 0xFFFFF00F; //PA9 max speed 50MHz,Alternate Function Output Push-pull;
                            //PA10 Input with pull-up/pull-down
  GPIOA->CRH |= 0x000008B0; //PA9 max speed 50MHz,Alternate Function Output Push-pull;
                            //PA10 Input with pull-up/pull-down
  RCC->APB2RSTR |= 1<<14; //reset USART1
  RCC->APB2RSTR &= ~(1<<14); //No effect
  USART1->BRR=mantissa;//the USART Divider -
  USART1->CR1=0x2008; //bit0 = 0:no break character is transmitted
                      //bit1 = 0:Receiver in active mode
                      //bit2 = 0:Receiver is disabled
                      //bit3 = 1:Transmitter is enabled
                      //bit4 = 0:IDLE Interrupt is inhibited
                      //bit5 = 0:RXNE Interrupt is inhibited
                      //bit6 = 0:Transmission complete interrupt is inhibited
                      //bit7 = 0:TXE interrupt is inhibited
                      //bit8 = 0:PE interrupt is inhibited
                      //bit9 = 0:Even parity
                      //bit10 = 0:Parity control disabled
                      //bit11 = 0:Idle Line
                      //bit12 = 0:Start bit, 8Data bits,n Stop bit
                      //bit13 = 1:USART enabled
}
```

First, calculate USART Divider, and then set PA9 as Alternate Function output push-pull, PA10 as pull-up input. Reset USART1 and set USART Divider as the value calculated. In the end, set Transmit mode.

2.2 Result

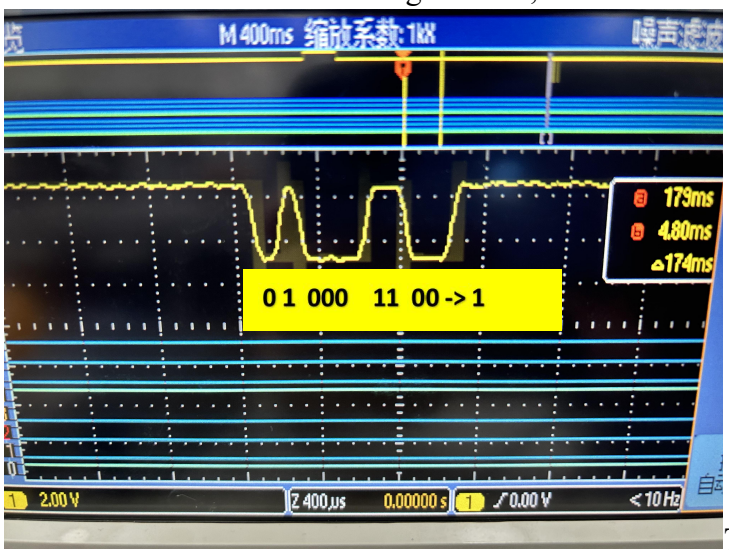


In the oscilloscope, we have:

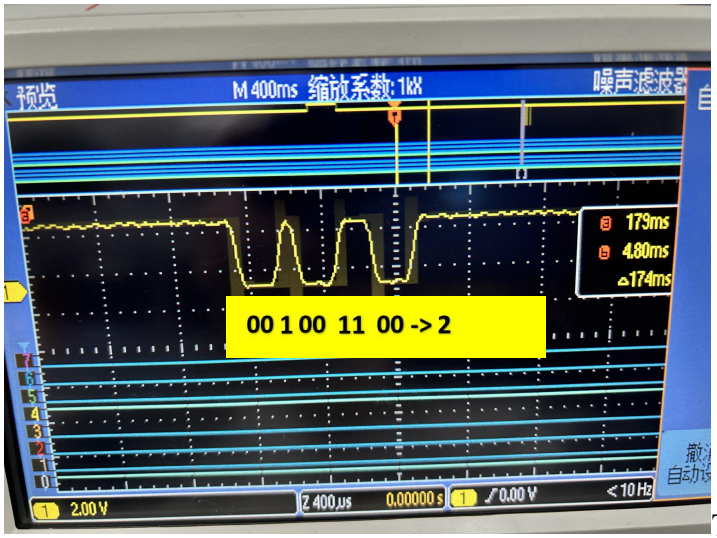


We can ordinate proper scale to see one by one.

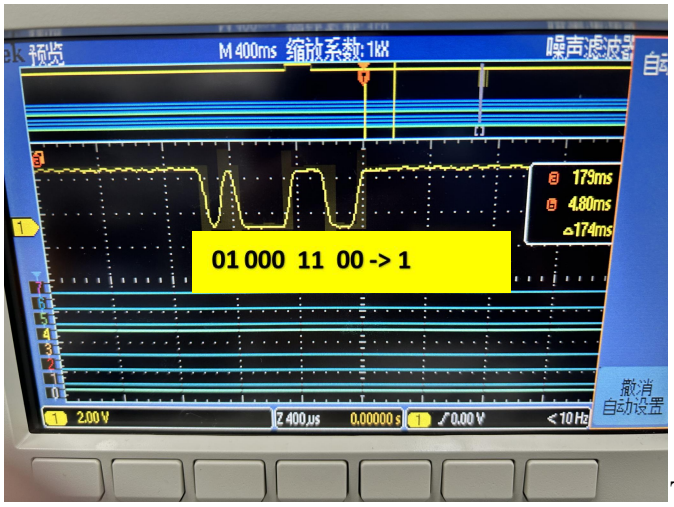
Notice: we need to read from right to left, and the last bit (which is 0) is stop bit.



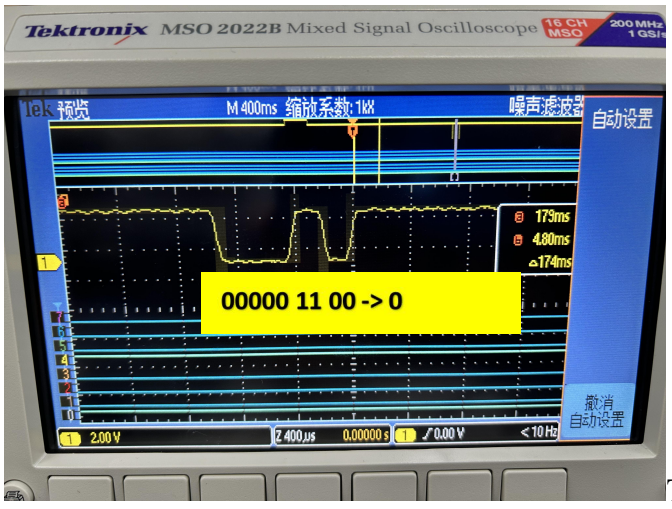
This is 1. (0x31=0b0011 0001)



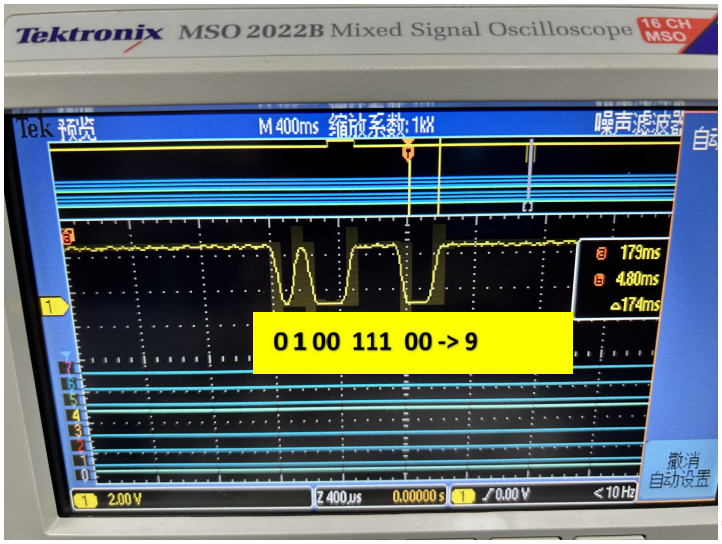
This is 2.(0x32=0b0011 0010)



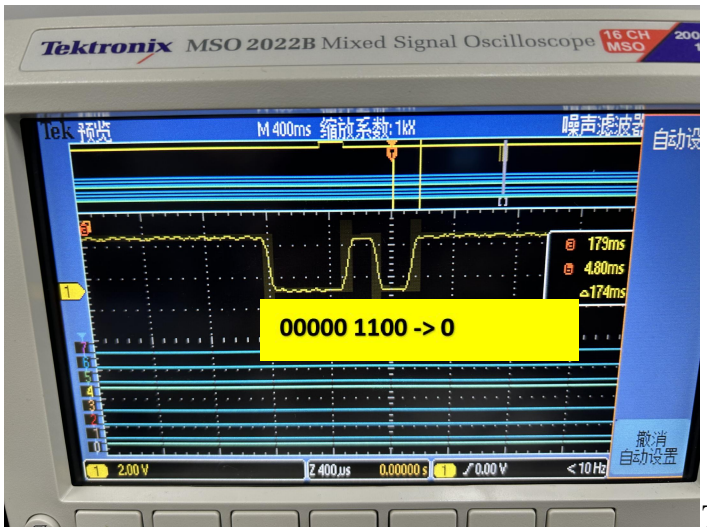
This is 1.(0x31=0b0011 0001)



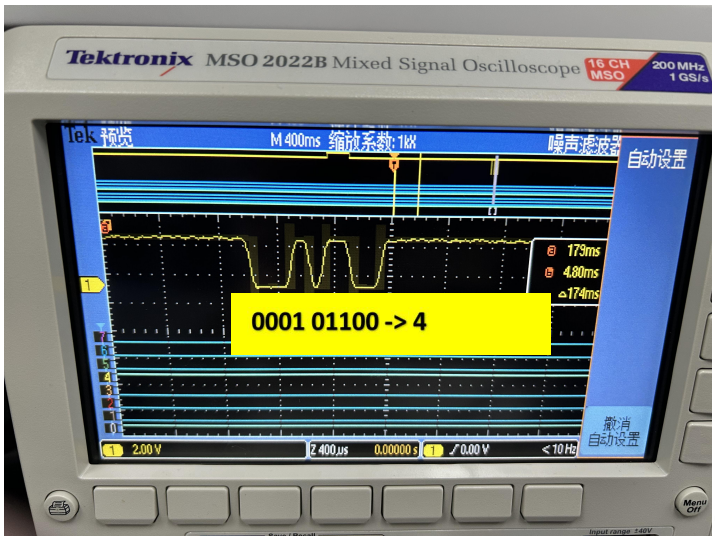
This is 0.(0x30=0011 0000)



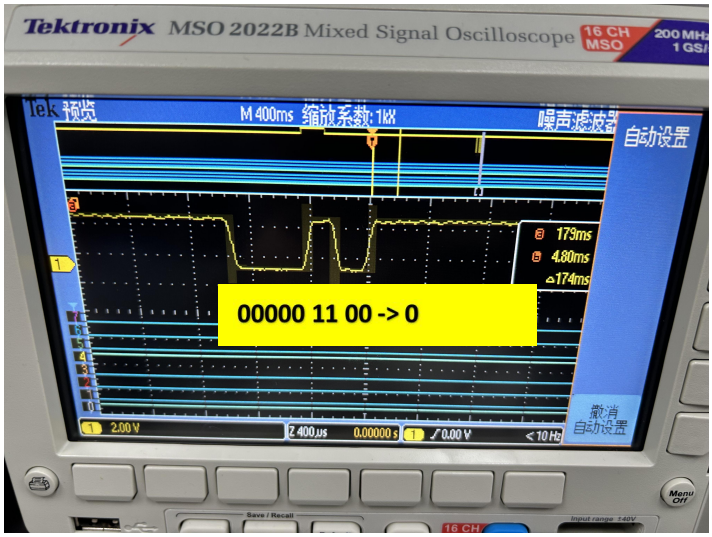
This is 9.(0x39=0b0011 1001)



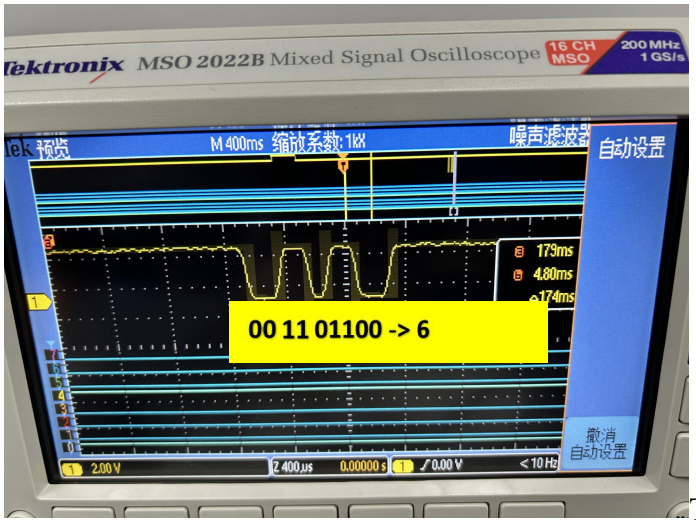
This is 0.(0x30=0b0011 0000)



This is 4.(0x34=0b0011 0100)



This is 0.(0x30=0b0011 0000)



This is 6.(0x36=0b0011 0110)

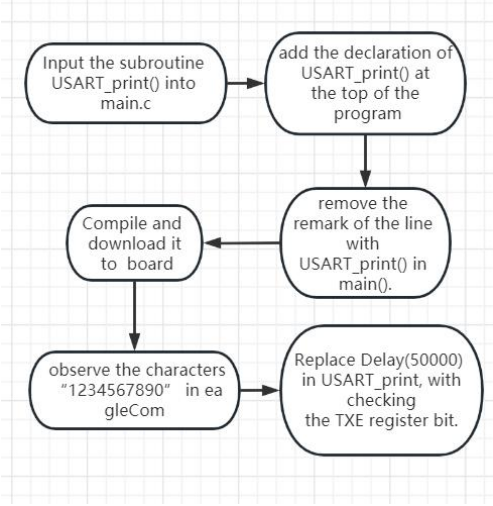
2.3 Questions

There is no question.

3. Experiment 3

3.1 Design

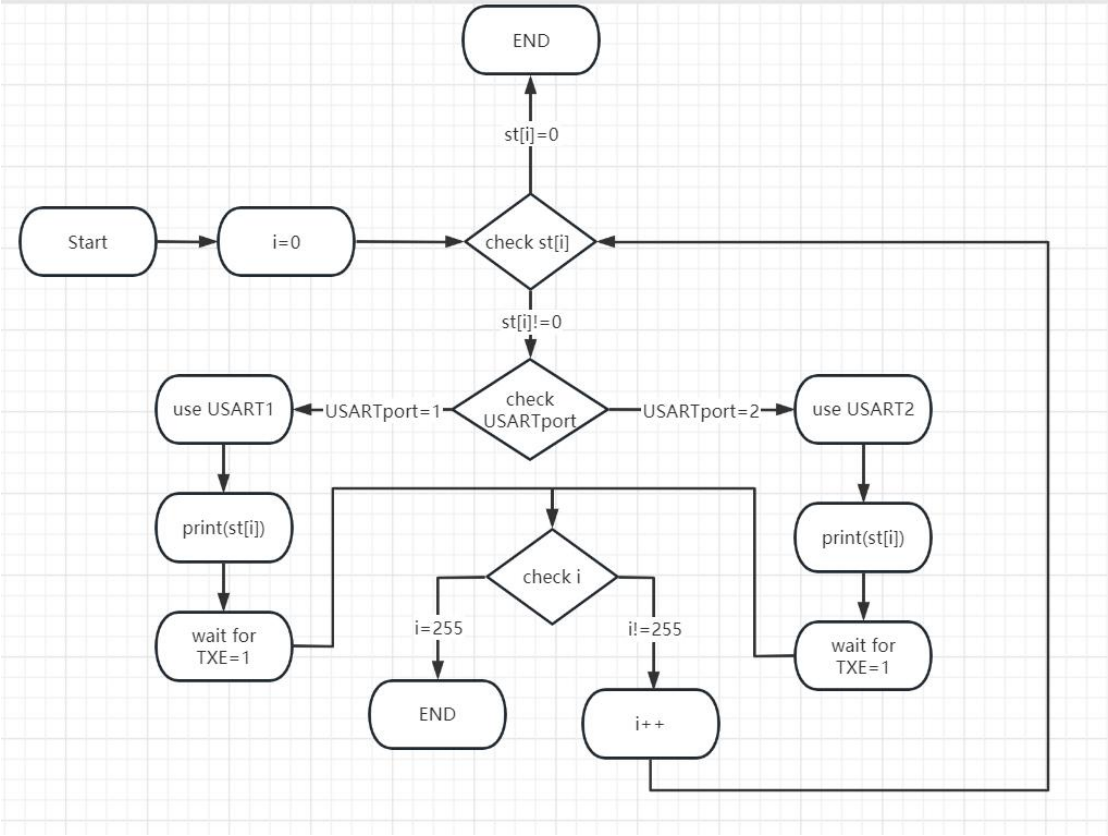
3.1.1 Designed program flowchart



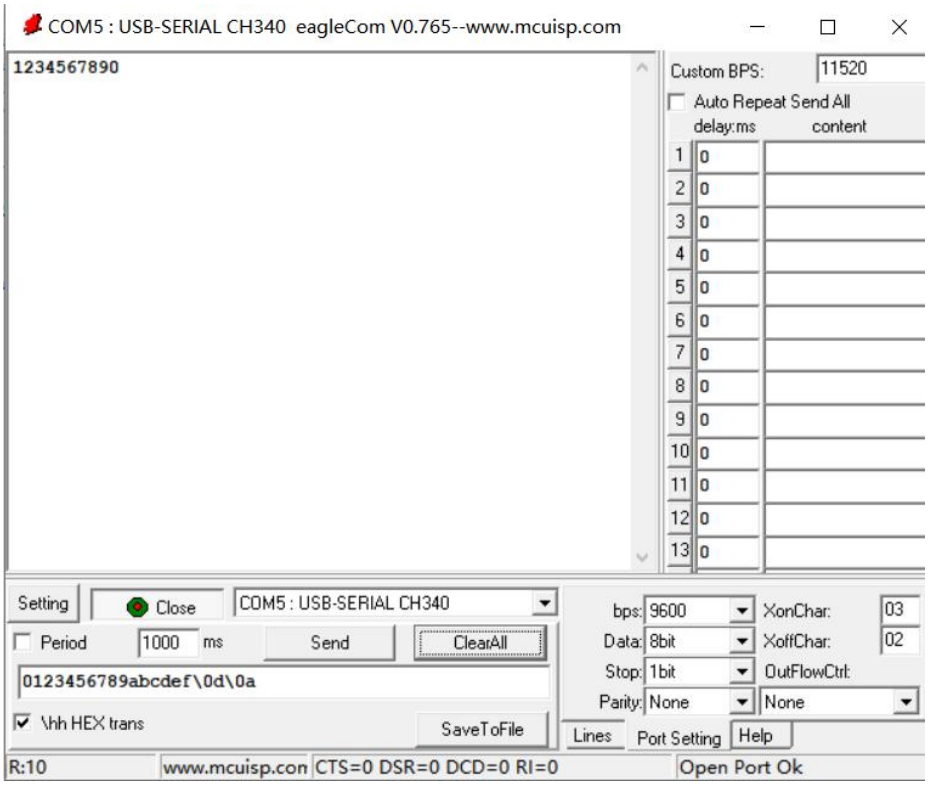
3.1.2 Explanation of source code

```
void USART_print(u8 USARTport, char*st)
{
  u8 i=0;
  while(st[i] !=0x00)
  {
    if (USARTport == 1)
    {
      USART1->DR=st[i];
      while((USART1->SR >> 7 & 1) != 1);//check TXE
    }
    if (USARTport == 2)
    {
      USART2->DR=st[i];
      while((USART2->SR >> 7 & 1) != 1);//check TXE
    }
    //Delay(50000);
    if (i==255) break;
    i++;
  }
}
```

The flowchart:



3.2 Result



We can see 1234567890 on the eagleCom

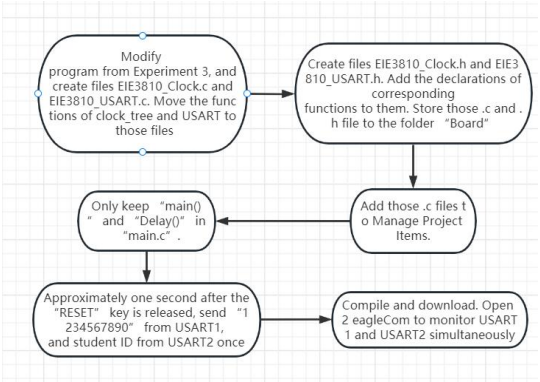
3.3 Questions

There is no question.

4. Experiment 4

4.1 Design

4.1.1 Designed program flowchart

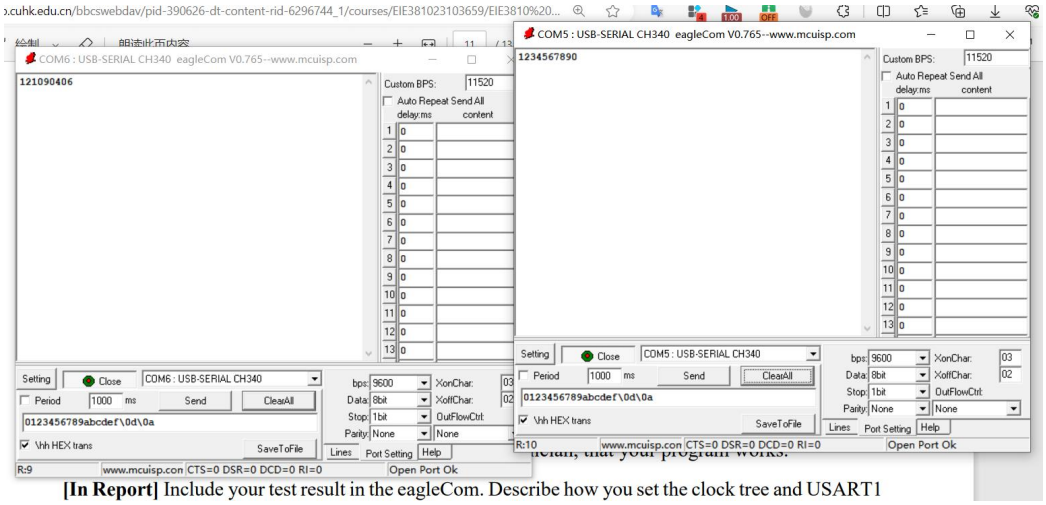


4.1.2 Explanation of source code

```
int main(void)
{
  EIE3810_clock_tree_init();
  EIE3810_USART1_init(72, 9600);
  EIE3810_USART2_init(36, 9600);
  Delay(10000000);
  USART_print(1,"1234567890");
  USART_print(2,"121090406");
}
```

The settings as before, we print 1234567890 and student ID on USART1 and USART2 respectively.

4.2 Result



[In Report] Include your test result in the eagleCom. Describe how you set the clock tree and USART1

We can see student ID and 1234567890 on two eagleCom

4.3 Questions

There is no question.

5. Conclusion

In this experiment, I has learned how to communicate by USART. I successfully transmit characters and numbers to my computer using eagleCom. And I also know how to choose proper port to realize the function of communicating. This is kind of like computer network, we can connect different devices.