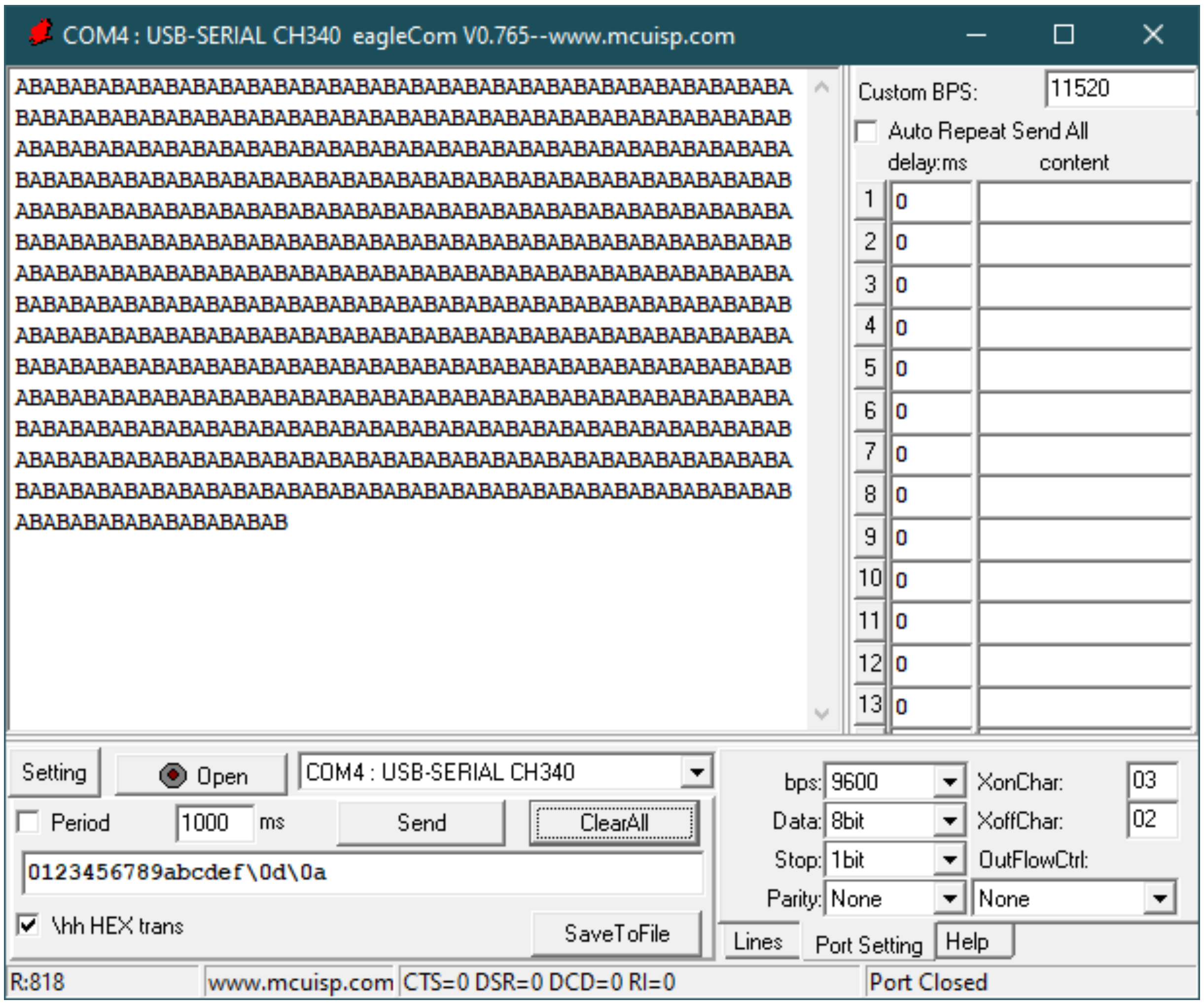
IERG3810 Lab 2 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) Report

HU, Han 1155107763 Session C No board number (board bought by myself)

Experiment 2.1 Setup USART2 to 9600pbs, 8N1.

Result with the default delay:



```
ABABABABABABABABABABABABABABABABABABABABABABABABABABABABA
BABABABABABABABABABABABABABABABABABABABABABABABABABABABAB
ABABABABABABABABABABABABABABABABABABABABABABABABABABABABA
BABABABABABABABABABABABABABABABABABABABABABABABABABABABAB
ABABABABABABABABABABABABABABABABABABABABABABABABABABABABA
BABABABABABABABABABABABABABABABABABABABABABABABABABABABAB
ABABABABABABABABABABABABABABABABABABABABABABABABABABABABA
BABABABABABABABABABABABABABABABABABABABABABABABABABABABAB
ABABABABABABABABABABABABABABABABABABABABABABABABABABABABA
BABABABABABABABABABABABABABABABABABABABABABABABABABABABAB
ABABABABABABABABABABABABABABABABABABABABABABABABABABABABA
BABABABABABABABABABABABABABABABABABABABABABABABABABABABAB
ABABABABABABABABABABABABABABABABABABABABABABABABABABABABA
BABABABABABABABABABABABABABABABABABABABABABABABABABABABAB
ABABABABABABABABABABAB
```

COM4 : USB-SERIAL CH340 eagleCom V0.765--www.mcuisp.com

Custom BPS: 11520

Auto Repeat Send All: unchecked; delay:ms 0 for rows 1–13

COM4 : USB-SERIAL CH340; Period 1000 ms (unchecked)

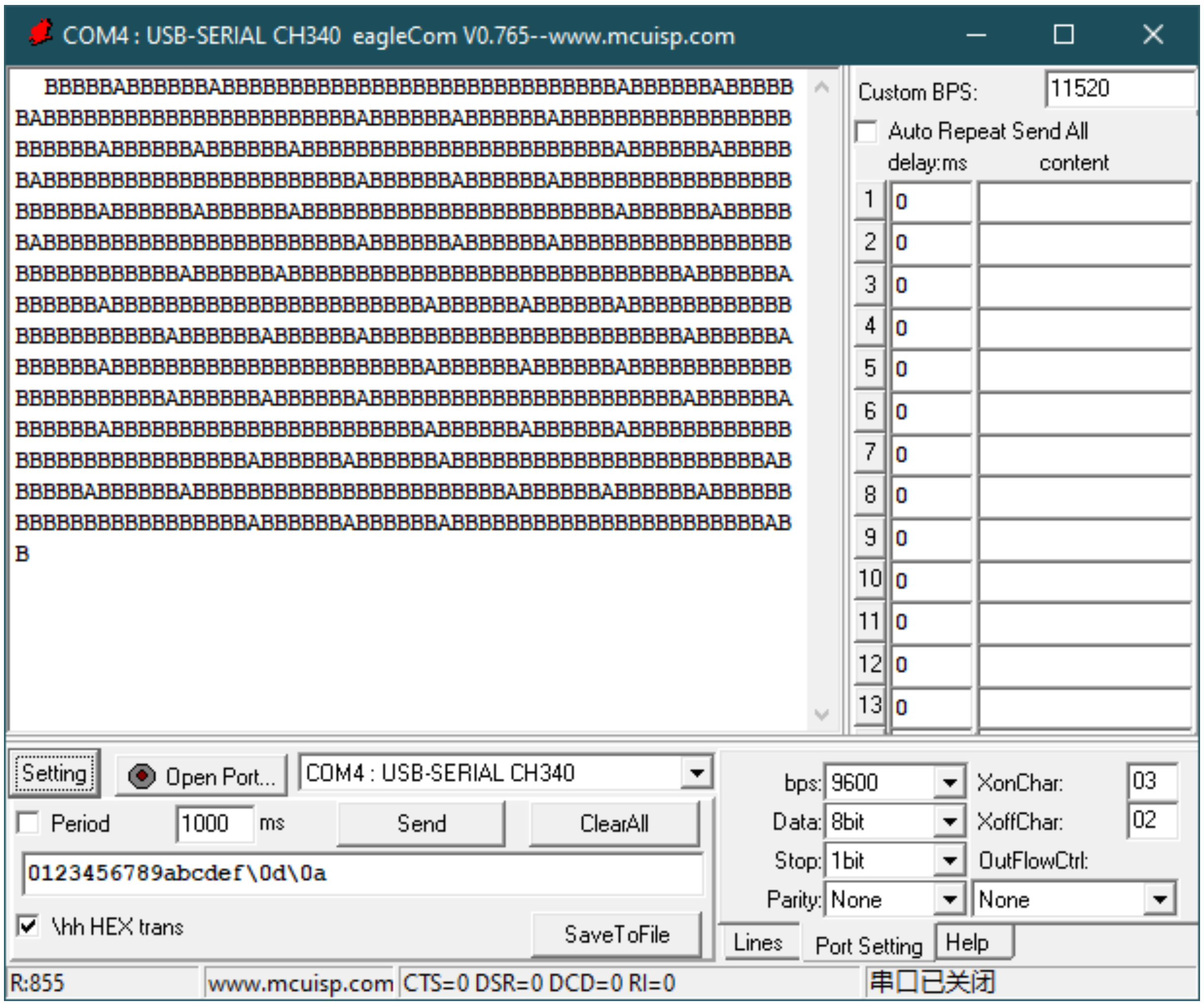
0123456789abcdef\0d\0a

\hh HEX trans: checked

bps: 9600 · Data: 8bit · Stop: 1bit · Parity: None · XonChar: 03 · XoffChar: 02 · OutFlowCtrl: None

R:818 | www.mcuisp.com | CTS=0 DSR=0 DCD=0 RI=0 | Port Closed

Result with modified delay (i.e. Modify the figure of `Delay(50000)` to `Delay(100)` in the main function.):



COM4 : USB-SERIAL CH340 eagleCom V0.765--www.mcuisp.com

```
  BBBBBABBBBBBBABBBBBBBBBBBBBBBBBBBBBBBBBBBBBABBBBBBBABBBBBB
BABBBBBBBBBBBBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBBBBBBB
BBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBBBBBBBBBBBBABBBBBBBABBBBBB
BABBBBBBBBBBBBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBBBBBBB
BBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBBBBBBBBBBBBABBBBBBBABBBBBB
BABBBBBBBBBBBBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBABBBBBBBBABBBBBBBBBBBBBBBBBBBBBBBBBBBBABBBBBBBA
BBBBBBABBBBBBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBB
BBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBBBBBBBBBBABBBBBBBA
BBBBBBABBBBBBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBB
BBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBBBBBBBBBBABBBBBBBA
BBBBBBABBBBBBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBBBBBBBB
BBBBBBBBBBBBBBBBBABBBBBBABBBBBBBABBBBBBBBBBBBBBBBBBBBBBBAB
BBBBBABBBBBBABBBBBBBBBBBBBBBBBBBBBBBABBBBBBBABBBBBBBABBBBBB
BBBBBBBBBBBBBBBBBABBBBBBABBBBBBBABBBBBBBBBBBBBBBBBBBBBBBAB
B
```

Custom BPS: 11520

Auto Repeat Send All

	delay:ms	content
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
9	0	
10	0	
11	0	
12	0	
13	0	

Setting | Open Port... | COM4 : USB-SERIAL CH340

Period 1000 ms | Send | ClearAll

0123456789abcdef\0d\0a

[✓] \hh HEX trans | SaveToFile

bps: 9600 | XonChar: 03

Data: 8bit | XoffChar: 02

Stop: 1bit | OutFlowCtrl:

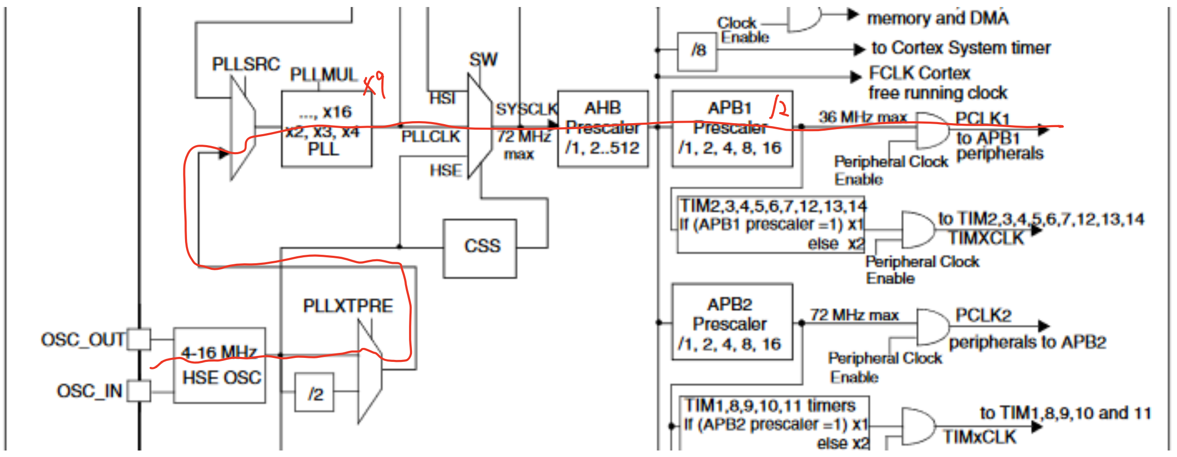
Parity: None | None

Lines | Port Setting | Help

R:855 | www.mcuisp.com | CTS=0 DSR=0 DCD=0 RI=0 | 串口已关闭

- We can see from the output of eagleCom that we cannot receive correct two alphabets that are displayed in the results of the default delay. Instead, we can observe that quite more 'B's are being received at the PC side than 'A's.
- This error happens because the delay time set is too small as mentioned in the lab manual, it will take about 1ms for a character transmitted at 9600bps by USART. However, `Delay(100)` only takes 0.0125ms to complete and therefore the 'A' may not be successfully sent before the program sends 'B' and the transmitted data will be corrupted as shown in the screenshot. Since we have a `Delay(1000000)` (0.8s) after each transmission of "AB", the number of successfully transmitted 'B's are obviously more than 'A's.

Clock tree initiation



During the initiation of the clock tree, we first modify the Clock configuration register (RCC_CFGR) and Clock   control register (RCC_CR) to clear previous settings.

```
RCC->CFGR &= 0xF8FF0000;
RCC->CR &= 0xFEF6FFFF;
```

The first line clears the MCO, ADCPRE, PPRE2, PPRE1, HPRE, SWS and SW but hold USBPRE, PLLMUL, PLLXTPRE and PLLSRC.

The second line clears PLLON, CSSON and HSEON and hold other bits.

Then, we begin to modify certain bits of the RCC_CFGR and RCC_CR to initialize the clock and set the clock as the mode that we need.

```
RCC->CR|=0x00010000; //HSEON=1
while(!(RCC->CR>>17)); //Check HSERDY
RCC->CFGR=0x00000400; //PPRE1=100
RCC->CFGR|=PLL<<18; //PLLMUL=111
RCC->CFGR|=1<<16; //PLLSRC=1
FLASH->ACR|=0x32; //set FLASH with 2 wait states
RCC->CR|=0x01000000; //PLLON=1
while(!(RCC->CR>>25)); //check PLLRDY
RCC->CFGR|=0x00000002; //SW=10
while(temp!=0x02){ //check SWS
    temp=RCC->CFGR>>2;
    temp&=0x03;
}
```

First, we enable the HSE clock and begin to check the external high-speed clock ready flag to see whether the HSE oscillator is ready. We will proceed to next steps until it is ready.

Then, we set the division factor for APB low-speed and high-speed prescaler (APB1 & APB2) to fulfil the requirements of the PCLK for APB peripherals. Here, since we will use the maximum of the SYSCLK (72MHz. explained later), PCLK1 (36MHz) and PCLK2 (72MHz), PPRE1 and PPRE2 are set according to HCLK /2 (100) and HCLK not divided (000). This value assignment also clears the USBPRE, PLLMUL, PLLXTPRE and PLLSRC and we need to set them.

As mentioned before, since we use the maximum of both PCLKs, the SYSCLK used here should be the maximum 72MHz. To generate such a frequency from our 8MHz source, we use a combination of PLLXTPRE=0 (division factor /1), PLLSRC=1 (HSE oscillator clock selected as PLL input clock), and PLLMUL=100 (x9 PLL) to get a 72MHz PLLCLK. Then we enable PLL and set SW to 10 to let PLLCLK be selected as system clock. After setting the PLLON and SW, corresponding check are performed to ensure that the clock settings are correct.

USART2 Initiation

```
float temp;
u16 mantissa;
u16 fraction;
temp=(float)(pclk1*1000000)/(bound*16);
mantissa=temp;
fraction=(temp-mantissa)*16;
mantissa<<=4;
mantissa+=fraction;
```

First, we calculate the bit rate and decide the mantissa and fraction accordingly, these number will be used later in the Baud rate register (USART_BRR).

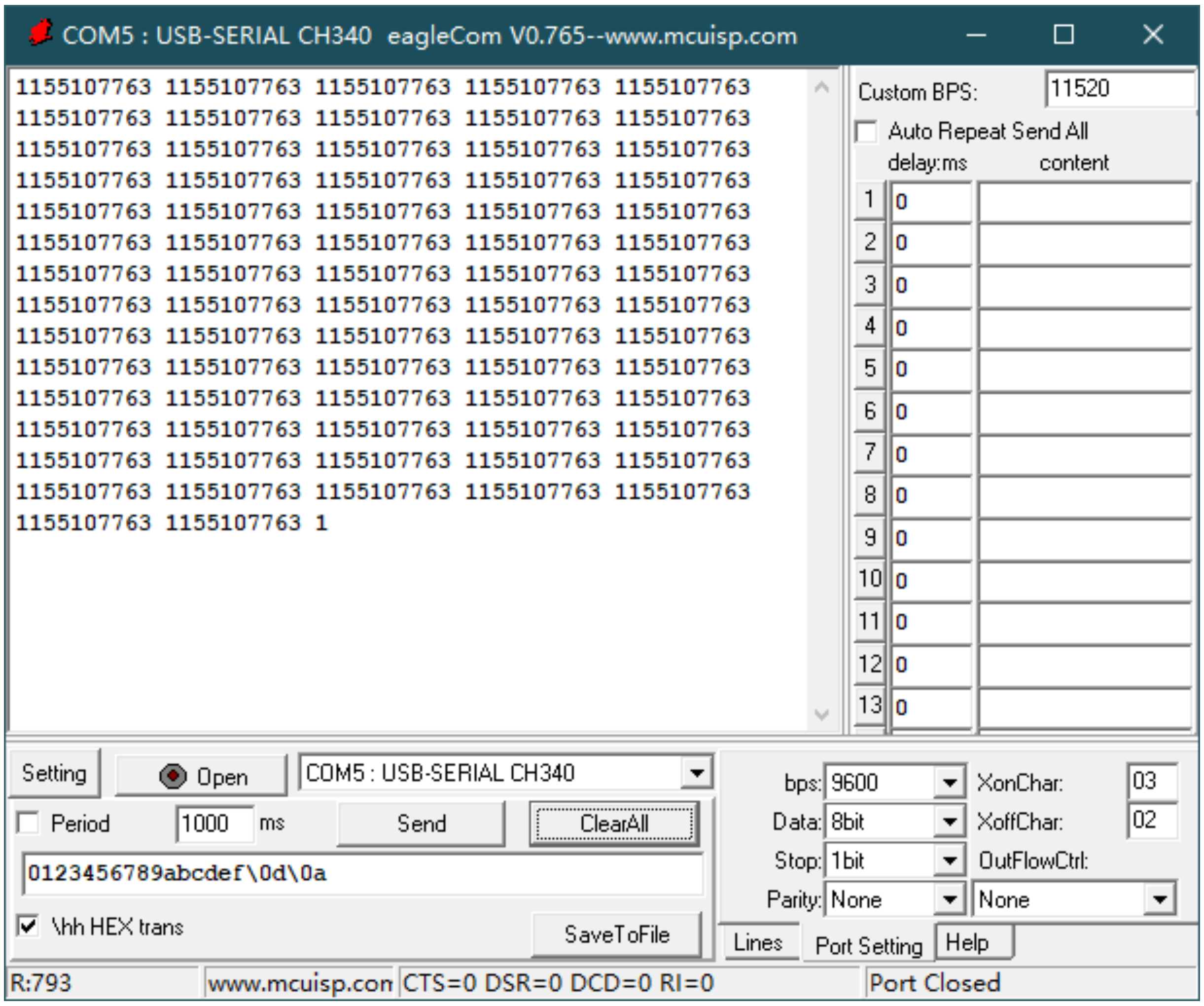
```
RCC->APB2ENR |= 1<<2; //enable port A clock
RCC->APB1ENR |= 1<<17; //enable USART2 clock
GPIOA->CRL &= 0xFFFF00FF; //set PA2, PA3 Alternate Function
GPIOA->CRL |= 0x00008B00; //set PA2, PA3 Alternate Function
RCC->APB1RSTR |= 1<<17; //reset USART2
RCC->APB1RSTR &=~(1<<17); //clear the reset status
USART2->BRR = mantissa; //set the mantissa and fraction
USART2->CR1 |= 0x2008; //enable USART and transmitter
```

Then we set corresponding register bits to enable and configure the port used (here port A and USART2 are used). The clock is first enabled and then set the used port PA2 & PA3 as alternate function output. USART2 are then reset for using and finally the calculated mantissa and fraction are set.

After the last step, which is enabling the USART and transmitter, we can transmit data from the USART2.

Experiment 2.2 Setup USART1 to 9600pbs, 8N1 with 72MHz.

Result using USART1 and sending my CUID:


COM5 : USB-SERIAL CH340 eagleCom V0.765--www.mcuisp.com

```
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1155107763 1155107763 1155107763
1155107763 1155107763 1
```

Custom BPS: 11520
Auto Repeat Send All

	delay:ms	content
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
9	0	
10	0	
11	0	
12	0	
13	0	

Setting | Open | COM5 : USB-SERIAL CH340
Period 1000 ms | Send | ClearAll
0123456789abcdef\0d\0a
[✓] \hh HEX trans | SaveToFile

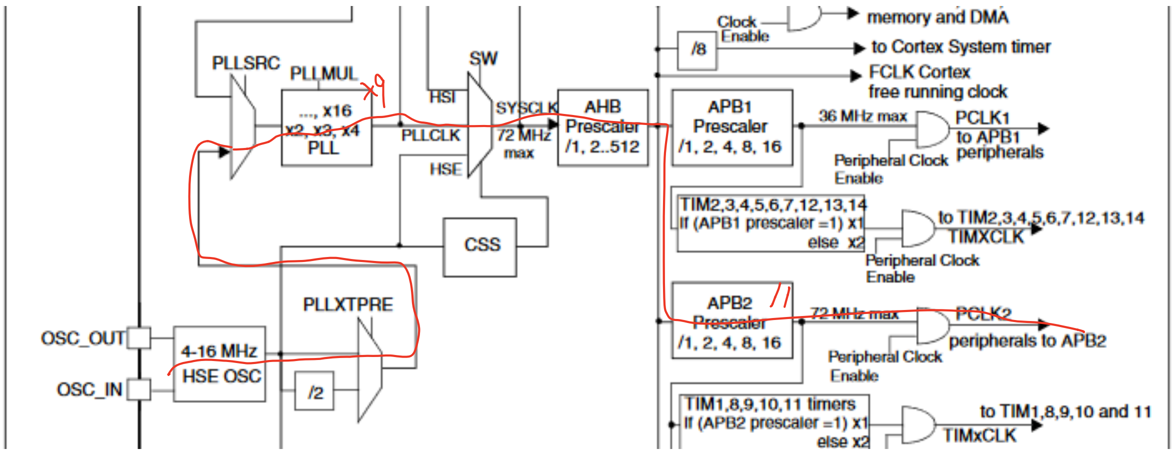
bps: 9600 | XonChar: 03
Data: 8bit | XoffChar: 02
Stop: 1bit | OutFlowCtrl:
Parity: None | None

Lines | Port Setting | Help

R:793 | www.mcuisp.con | CTS=0 DSR=0 DCD=0 RI=0 | Port Closed

- In this experiment, the USART1 is used instead of USART2 and high speed APB is used and the clock for the peripherals is the 72MHz PCLK2.

Clock tree initiation



Here PCLK2 is used. Therefore, there is no big difference between this initiation process and the one in experiment 2.1, except the prescaler of APB2 should be set differently (division factor /1).

```
RCC->CFGR &= 0xF8FF0000;
RCC->CR &= 0xFEF6FFFF;
RCC->CR|=0x00010000; //HSEON=1
while(!(RCC->CR>>17)); //Check HSERDY
RCC->CFGR=0x00000400; //PPRE1=100, PPRE2=000
RCC->CFGR|=PLL<<18; //PLLMUL=111
RCC->CFGR|=1<<16; //PLLSRC=1
```

```
FLASH->ACR|=0x32; //set FLASH with 2 wait states
RCC->CR|=0x01000000; //PLLON=1
while(!(RCC->CR>>25)); //check PLLRDY
RCC->CFGR|=0x00000002; //SW=10
while(temp!=0x02){ //check SWS
    temp=RCC->CFGR>>2;
    temp&=0x03;
}
```

However, we can use the same code for the clock tree initiation as the value set to make PPRE1=100 also ensure that PPRE2=000, which satisfies the need for the prescaler of APB2.

USART1 initiation

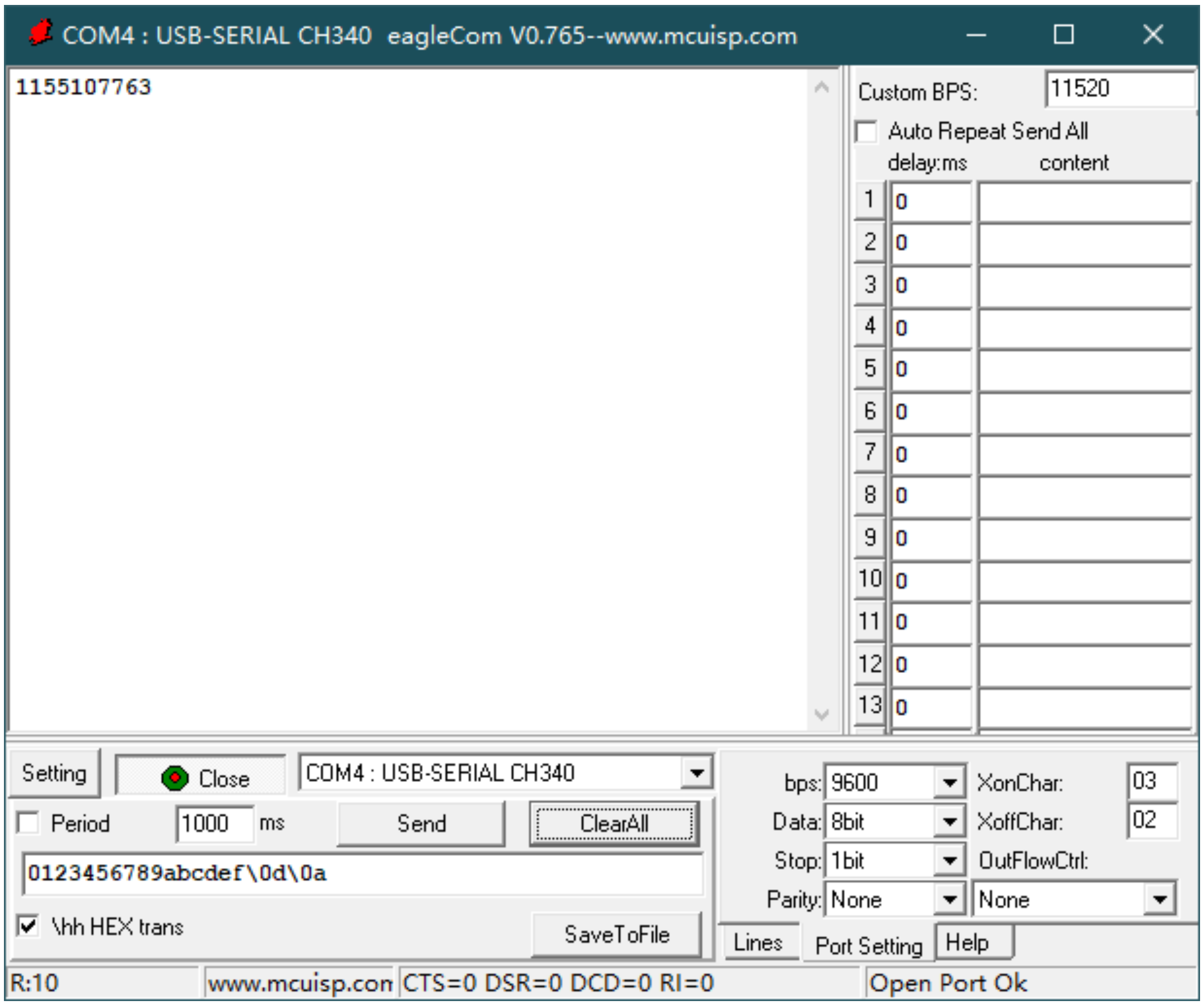
The structure of the code is exactly the same as the USART2 initiation. But since USART1 is used here, the register setting should changed to the one for USART1 and the port used here is PA9 and PA10. We should make changes accordingly.

```
temp=(float)(pclk2*1000000)/(bound*16);
mantissa=temp;
fraction=(temp-mantissa)*16;
mantissa<<=4;
mantissa+=fraction; //calculate mantissa and fraction
RCC->APB2ENR |= 1<<2; //enable port A clock
RCC->APB2ENR |= 1<<14; //enable USART1 clock
GPIOA->CRH &= 0xFFFFF00F; //set PA9, PA10 Alternate Function
GPIOA->CRH |= 0x000008B0; //set PA9, PA10 Alternate Function
RCC->APB2RSTR |= 1<<14; //reset USART1
RCC->APB2RSTR &=~(1<<14); //clear the reset
USART1->BRR = mantissa; //set the mantissa and fraction
USART1->CR1 |= 0x2008; //enable USART and transmitter
```

- Note that the corresponding bits for PA9 and PA10 locate in GPIOA_CRH register (instead of CRL).
- Also the corresponding register bits for clock enabling and reset of USART1 are all in APB2ENR and APB2RSTR (instead of APB1ENR and APB1RSTR).

Experiment 2.3 Send a string to USART with checking TXE bit

Result when checking TXE bit: Here I transmit my own CUID through USART2.


COM4 : USB-SERIAL CH340 eagleCom V0.765--www.mcuisp.com

```
1155107763
```

Custom BPS: 11520
Auto Repeat Send All
delay:ms content

	delay:ms	content
1	0	
2	0	
3	0	
4	0	
5	0	
6	0	
7	0	
8	0	
9	0	
10	0	
11	0	
12	0	
13	0	

Setting | Close | COM4 : USB-SERIAL CH340
Period 1000 ms | Send | ClearAll
0123456789abcdef\0d\0a
[✓] \hh HEX trans | SaveToFile
bps: 9600 | XonChar: 03
Data: 8bit | XoffChar: 02
Stop: 1bit | OutFlowCtrl:
Parity: None | None
Lines | Port Setting | Help
R:10 | www.mcuisp.con | CTS=0 DSR=0 DCD=0 RI=0 | Open Port Ok

The details of the modified `USART_print` function:

```
void USART_print(u8 USARTport, char* st){
    u8 i=0;
    while (st[i]!=0x00){
        if(USARTport==1){
            USART1->DR=st[i];
            while((USART1->SR &=0x00000080)>>7!=1);
        }
        if(USARTport==2){
            USART2->DR=st[i];
            while((USART2->SR &=0x00000080)>>7!=1);
        }
        if(i==255)
            break;
        i++;
    }
}
```

- Here the checking procedure is done by looping and conduct bitwise operations to check the 7th bit of USART_SR is 1 or not. If it doesn't equal to 1, the data has not been transmitted to the shift register and we should wait until the transmission is done.