EIE3810 Microprocessor System Design Laboratory

Lab 2. Universal Synchronous-Asynchronous Receiver-Transmitter (USART)

School of Science and Engineering

The Chinese University of Hong Kong, Shenzhen

2023-2024 Term 1

1. Objectives

In Lab 2, we will spend the 2-week session to study the following:

- To study the clock tree of Cortex-M3
- To study how to setup and transmit data by USART

2. Basics

A universal synchronous-asynchronous receiver-transmitter (USART) is a type of serial interface device that can be programmed to communicate synchronously or asynchronously. Serial transmission of digital information (bits) through a single wire is less expensive than parallel transmission through multiple wires.

In synchronous mode, aside from the transmitted data, one device will generate a clock pulse and transmit it to the receiving device. Based on this clock pulse, the receiver can sample the data received, and get to know the information from the transmitter.

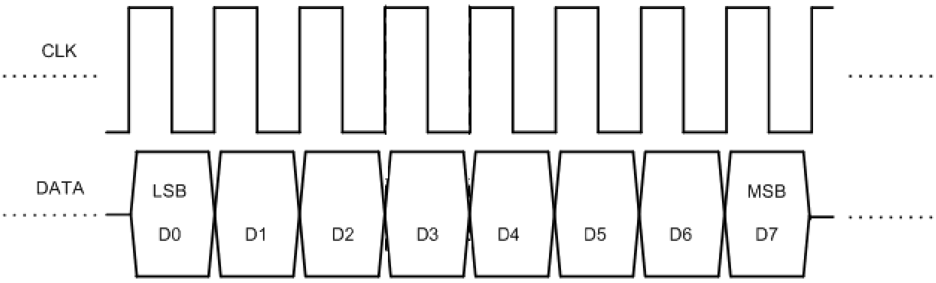


Figure 1. Synchronous serial transmission

Asynchronous serial communication, in another way, does not send out clock pulses. The two devices must set up the same baud rate (i.e., unit for symbol rate). If only "0" or "1" are transmitted at one time, the baud rate is equivalent to the bit rate. The receiver can retrieve the information from one single wire. The signal is high when the line idles. If the receiver detects a low voltage, it will start receiving the next 8[1] bits as data. After that, there can be an optional parity bit. There are two typical types of parity, i.e., even or odd[2]. The last bit is a stop bit, which is a high voltage.

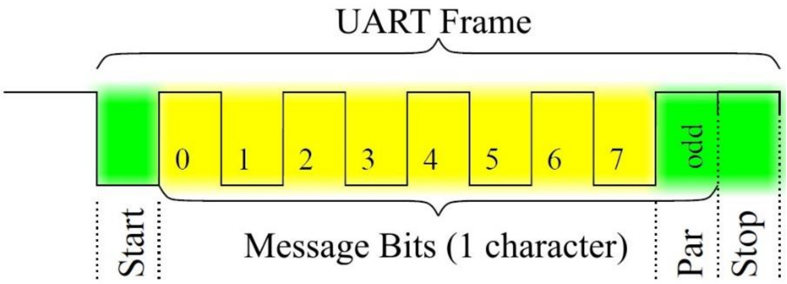


Figure 2. Asynchronous serial transmission

[1] In general, there are 8 bits data. But there can also be other lengths.

[2] In even parity, sum of the number of "1"s in data bits and the parity bit should be an even number. In odd parity, that sum should be an odd number. Hence, in Figure 2, in odd parity, the parity bit is 1, as the sum of "1"s is 5.

USART has both synchronous and asynchronous modes. Another term, UART (the universal asynchronous receiver-transmitter), can only work in an asynchronous mode. STM32F103ZET6 has both USART and UART interfaces, as shown in Figure 3.

In this lab, we will use two USARTs (USART1 and USART2).

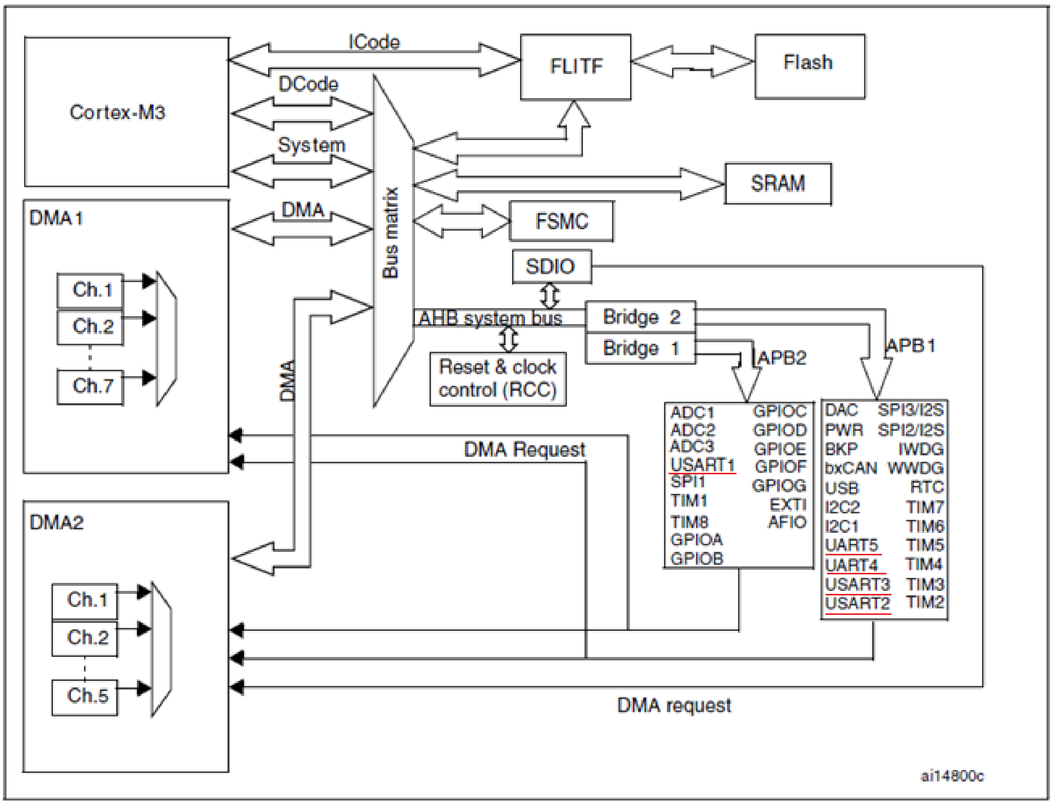


Figure 3. The system architecture of STM32F103ZET6

USART1 is connected with a USB-to-Serial chip CH340 on board (Figure 4 and Figure 5). The driver for CH340 has already been installed on the computer. When you download codes, you are using CH340.

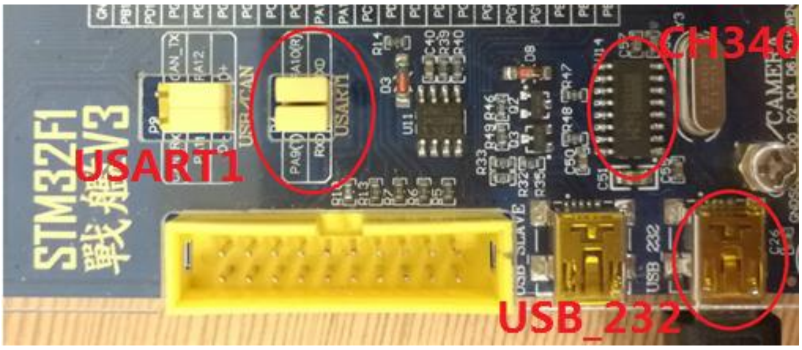


Figure 4. Locations of USART1, CH340, and USB_232

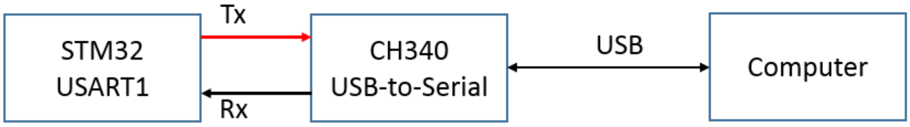


Figure 5. USART1 connection with computer

USART2 is connected with a chip SP3232, which is a level converter chip to convert the microprocessor’s signal levels to RS-232 signal level, and then connect to another device through a standard DB9 female port (COM port, or RS232 serial port). For connection with the computer, you can check Figure 6 and 7. On the workbench, you have one USB-to-Serial cable (one end is USB and the other is male RS232).

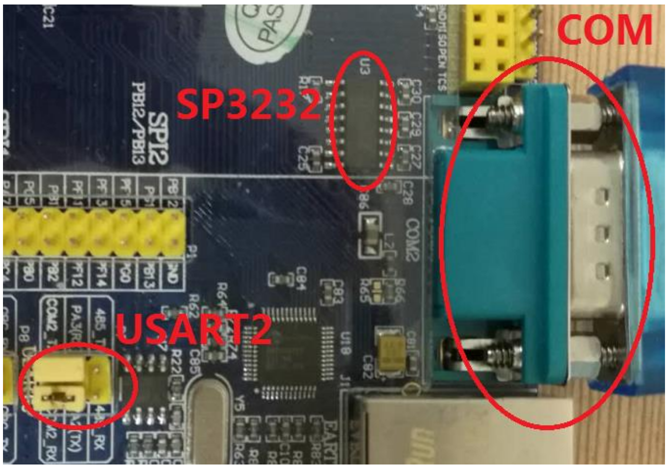
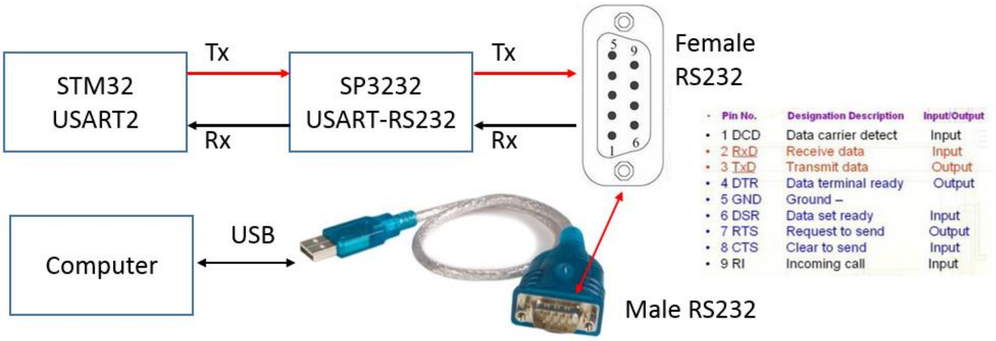


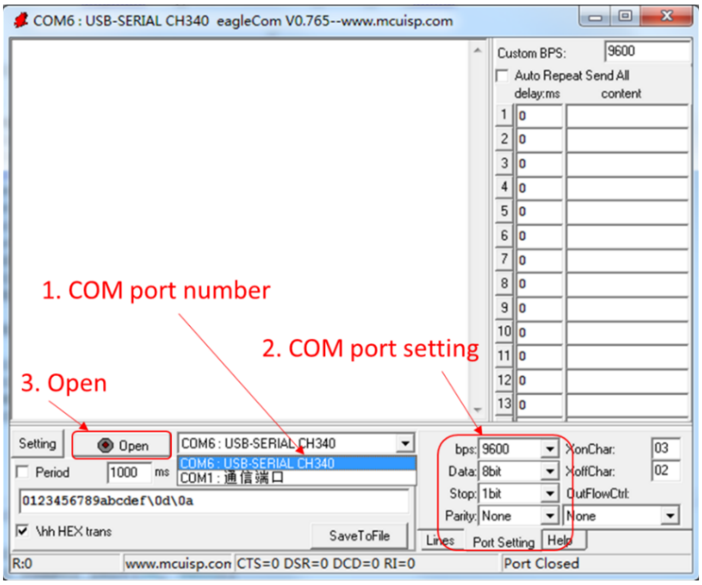
Figure 6. Locations of USART2, SP3232, and COM



Pin No.	Designation Description	Input/Output
1 DCD	Data carrier detect	Input
2 RxD	Receive data	Input
3 TxD	Transmit data	Output
4 DTR	Data terminal ready	Output
5 GND	Ground –	
6 DSR	Data set ready	Input
7 RTS	Request to send	Output
8 CTS	Clear to send	Input
9 RI	Incoming call	Input

Figure 7. USART2 connection with computer

In computer, the application eagleCom.exe can be used to receive/ transmit serial data


COM6 : USB-SERIAL CH340 eagleCom V0.765--www.mcuisp.com

1. COM port number

2. COM port setting

3. Open

Custom BPS: 9600

□ 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 COM6 : USB-SERIAL CH340

COM6 : USB-SERIAL CH340
COM1 : 通信端口

□ Period 1000 ms

0123456789abcdef\0d\0a

☑ \hh HEX trans

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

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

Figure 8. eagleCom.exe interface

In this lab, we will use the asynchronous serial protocol of the USART port. The lab handout will guide you on how to setup USART2. You can follow the configuration of USART2 and set up USART1 by yourself.

There are several steps in programming for USART, including A) clock-tree initialization, B) USART initialization, and C) data transmission.

A) Initialize clock tree

As shown in Figure 3, USART1 and USART2 are connected to APB2 and APB1 buses, respectively. APB1 and APB2 connect to two different nodes of the clock tree (Figure 9).

We will use an 8MHz crystal as an external clock source, which is connected to OSC_IN and OSC_OUT. Thus, the high-speed external clock signal (HSE) should be chosen. We will first use USART2 and then USART1.

The target is to setup 36MHz as PCLK1 for APB1 (for USART2). The following register bits should be set.

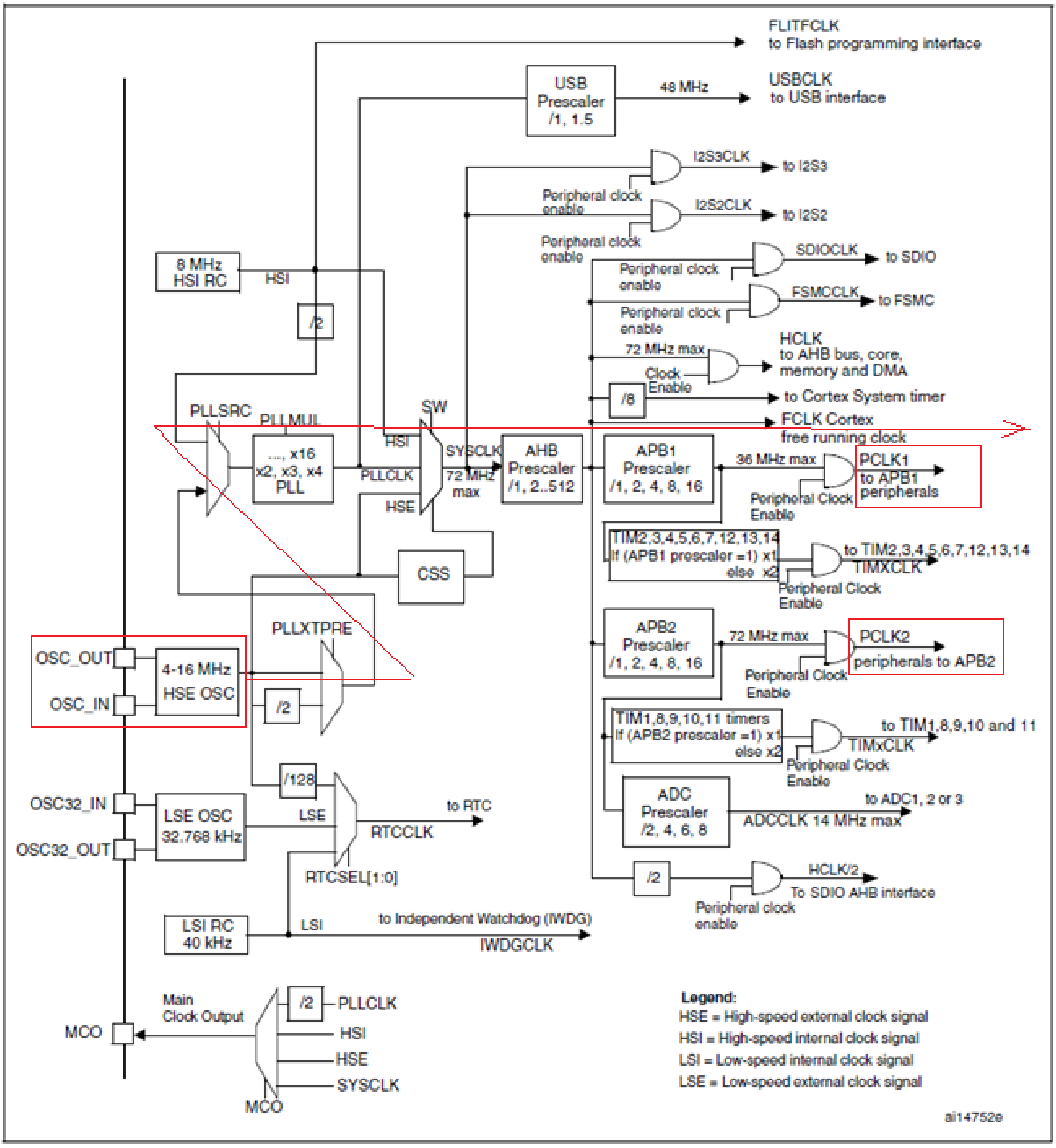


Figure 9. Clock tree

RCC_CR (Reset and Clock Control: Clock configuration register, pages 98-99 in RM0008)

- HSEON (HSE clock enable): turn on HSE clock.
- HSERDY (External high-speed clock ready flag): check if the HSE clock is ready.
- PLLON (PLL enable): turn on PLL.
- PLLRDY (PLL clock ready flag): check if PLL is locked.

RCC_CFGR (Reset and Clock Control: Clock configuration register, pages 100-102 in RM0008)

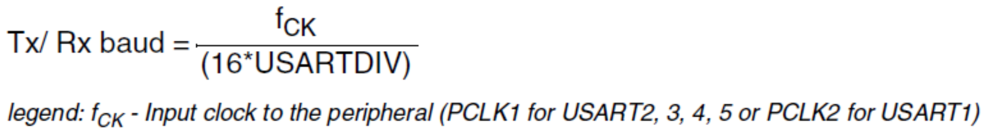
- PLLXTPRE (HSE divider for PLL[3] entry): HSE clock not divided.
- PLLSRC (PLL entry clock source): HSE oscillator clock selected as PLL input clock
- PLLMUL (PLL multiplication factor): PLL input clock x 9, so that the PLLCLK can reach 72MHz (8MHz x 9).
- SW (System clock switch): PLL selected as the system clock. Then SYSCLK is 72MHz (maximum).
- SWS (System clock switch status): Check which clock source is used as the system clock.
- HPRE (AHB Prescaler): SYSCLK not divided. This will generate HCLK as 72MHz (maximum).
- PPRE1 (APB low-speed prescaler for APB1): HCLK divided by 2. This will generate 36MHz for PCLK1, but it is not output yet.

RCC_APB1ENR (Reset and Clock Control: APB1 peripheral clock enable register, pages 114-116 in RM0008)

- USART2EN (USART2 clock enable): USART2 clock enabled. This will output the 36MHz of PCLK1 for USART2.

B) Initialize USART

After setting up the clock tree, we have to set up the baud rate of the serial port. If we set the baud rate to 9600 bps (bit per second), it takes 1/9600≈0.104 ms for each bit of transmission. Reference can be taken from pages 798-800 in RM0008.



$$\text{Tx/ Rx baud} = \frac{f_{CK}}{(16*\text{USARTDIV})}$$

legend: f_{CK} - Input clock to the peripheral (PCLK1 for USART2, 3, 4, 5 or PCLK2 for USART1)

9600 bps is one typical baud rate. f_{CK} is 36MHz as we set up for PCLK1.

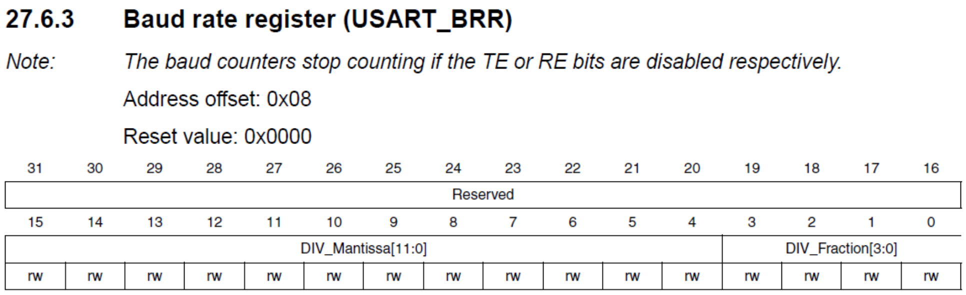
USARTDIV=36,000,000/9,600/16=0d234.375[4]

To setup USARTDIV in the USARTx_BRR register, USARTDIV is separated into two parts:

- DIV_Mantissa = 0d234=0xEA;
- DIV_Fraction = 16*0d0.375 = 0x06;

[3] PLL is Phase-Lock Loop, which provides synchronous clock with external clock.

[4] 0d means that it is decimal number.



27.6.3 Baud rate register (USART_BRR)

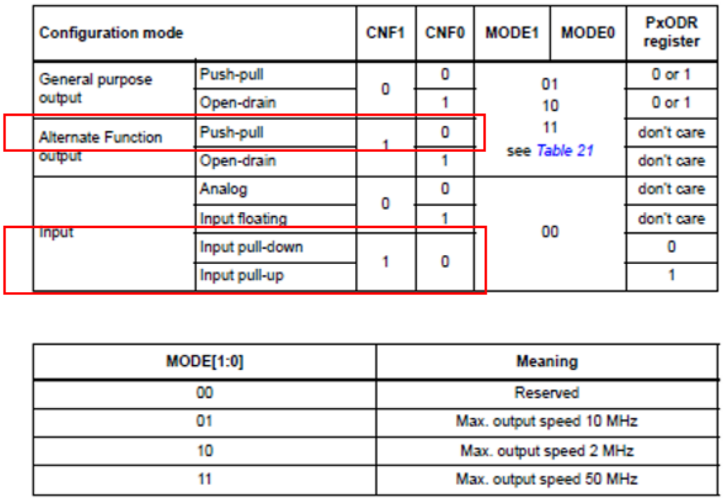
Note: The baud counters stop counting if the TE or RE bits are disabled respectively.

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

After the baud rate, you need to configure the pins PA2 and PA3 for USART2. PA2 is Tx, so it should be output (Max speed can be set as 50MHz); PA3 is Rx, so it should be input. Refer to Lab 1 about the setting of I/O. When setting up PA2, the mode is Alternate Function output Push-Pull, rather than GPIO. For setting PA3, “0” and “1” will be on the signal wire, and there is no floating signal which has uncertain voltage. Hence, either Input Pull-up or Input Pull-down is acceptable.



Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	01 10 11 see Table 21		0 or 1
	Open-drain		1			0 or 1
Alternate Function output	Push-pull	1	0			don't care
	Open-drain		1			don't care
Input	Analog	0	0	00		don't care
	Input floating		1			don't care
	Input pull-down	1	0			0
	Input pull-up					1

MODE[1:0]	Meaning
00	Reserved
01	Max. output speed 10 MHz
10	Max. output speed 2 MHz
11	Max. output speed 50 MHz

Figure 10. The configuration mode for PA

During this USART initialization process, you need to use several registers.

RCC_APB2ENR and GPIOA_CRL

- To enable GPIOA, you can refer to Lab 1.

RCC_APB1RSTR (APB1 peripheral reset register, page 108-110 in RM0008)

- USART2RST (USART2 reset): Reset USART2

USART_BRR (Baud rate register. For USARTx the name should be USARTx_BRR. Page 820 in RM0008)

- DIV_Mantissa: mantissa of USARTDIV
- DIV_Fraction: fraction of USARTDIV

USART_CR1 (control register 1. For USARTx the name should be USARTx_CR1. Page 821 in RM0008)

- UE (USART enable): enable USART.
- M (Word length): 1 start bit, 8 data bit, 1 stop bit.
- PCE (Parity control enable): disable parity.
- TE (Transmitter enable): enable transmitter.

C) Transmit data

The next step is to transmit data. The data is put into the data register of USARTx (USARTx_DR) for transmission. Refer to page 820 of RM0008.

For example, the coding below sends a byte 0b01000001 (0x41) to the data register of USART2, and then transmit out:

USART2->DR = 0x41;

LSB (Least Significant Bit) will be sent first, so on the pin of Tx, what you can observe by oscilloscope is “10000010”, rather than “01000001”.

If we set the baud rate to 9600 bps (bit per second). For each bit of transmission, it takes 1/9600≈0.104 ms. For each frame (generally containing 10 bits, if there is no parity bit), it needs to have about 1.04ms.

But CPU operates at a significantly faster speed than communication in USARTx. If CPU sends another frame of data to USARTx_DR, while the previous transmission is still on the way, it will corrupt the previous data. Hence, the CPU needs to wait, while USARTs are busy with data transmission.

There are different ways to wait. Using interrupt is an efficient way, enabling the CPU to do another thing. We will learn to interrupt in later labs. In this lab, we will use two other methods: “delay a fix time period” and “busy wait”.

For “delay a fixed time period”, it adopts the looping function Delay(), which we use in Lab 1.

For “busy wait”, it repeatedly checks whether the transmission job is completed. If yes, new data will be sent. Although it still wastes time and computing power, it is better than Delay(). (Experiment 3)

3. Experiments

3.1 Experiment 1: Setup USART2 to 9600bps, 8N1

Procedures:

1) Create a new project or copy Lab 1 project for this experiment.
2) Download main.c from Blackboard, and add it into your project. For “FLASH->ACR =0x32”, two wait states should be set if 48 MHz < SYSCLK ≤ 72 MHz. (page 59 of RM0008)
3) Compile your project and download it into the project board.
4) Connect a USB-to-Serial cable to the board “COM2” DB-9-female port (near the upper-right corner of the board, refer to Figure 6) and the computer.
5) Close the FlyMCU program, then run the serial interface program “eagleCom.exe” (Figure 8). Select the suitable COM port for this program. This COM port number is assigned by computer, rather than COM2 in the project board. You can check the eagleCom interface when unplug and re-plug the USB-to-Serial cable.
6) In eagleCom, select 8N1 (i.e. bps: 9600, Data: 8 bit, Parity: None, Stop: 1 bit). Press “Open”. Then, press RESET on the project board. You will see two alphabets shown on the eagleCom interface repeatedly.
7) Use an oscilloscope to measure PA2 (DON NOT forget the reference GND). To measure PA2 and keep the connection between PA2 and COM2_RX, you need to add a stacking header between the

pins and jumper cap (Figure 11). Get it from TA or technician. Compare your observation with Figure 2, and describe your observations in your report.

8) Modify “Delay(50000)” to “Delay(100)” in the main function. Describe what has happened (check eagleCom), in your report.
9) Describe the procedures of subroutine “EIE3810_clock_tree_init()” in your report.
10) Describe the procedures of subroutine “EIE3810_USART2_init()” in your report.

[Demonstration] Demonstrate to instructor, TA, or technician, that your program works.

[In Report] Include your test result in the eagleCom. Also, include the requirement in steps 7), 8), 9) and 10).

[In Source Code] For each line marked “Add comments”, understand the function of that code and describe it briefly in the code comments.

[Question] Why is there an error of serial communication in step 8?

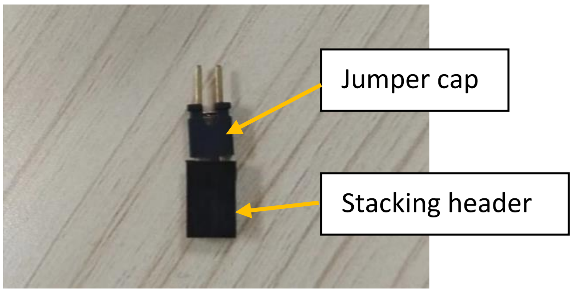


Figure 11. The header and jumper hat that you need to measure PA2

3.2 Experiment 2: Setup USART1 to 9600bps, 8N1 with 72 MHz

Two USART ports, i.e., USART1 and USART2, of STM32 are connected to the project board. We have studied the setting of USART2 in Experiment 1. USART1 connects to a USB-to-Serial chip on the board and provides a USB interface to download codes. If you want to use USART1 for communication, there are two ways.

Method 1:

- Use USB_232 for code downloading first by FlyMCU. After the code has been downloaded, FlyMCU will automatically release the occupation of this COM port.
- Open eagleCom.exe and open the same COM port, which FlyMCU occupied before. USART1 will be used for communication through USB-232 now.
- If you want to re-download the code, you need to close this port in eagleCom.exe. Otherwise, it cannot be open by FlyMCU.

Method 2:

- Use USB_232 solely for USART1 communication.
- Connect ST-Link (Figure 12) to the JTAG port of the project board. Use one more USB cable to connect to another USB port on the computer.

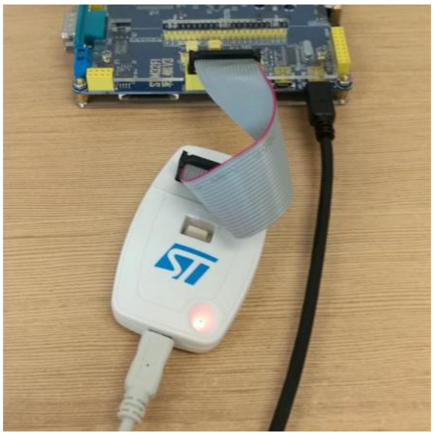
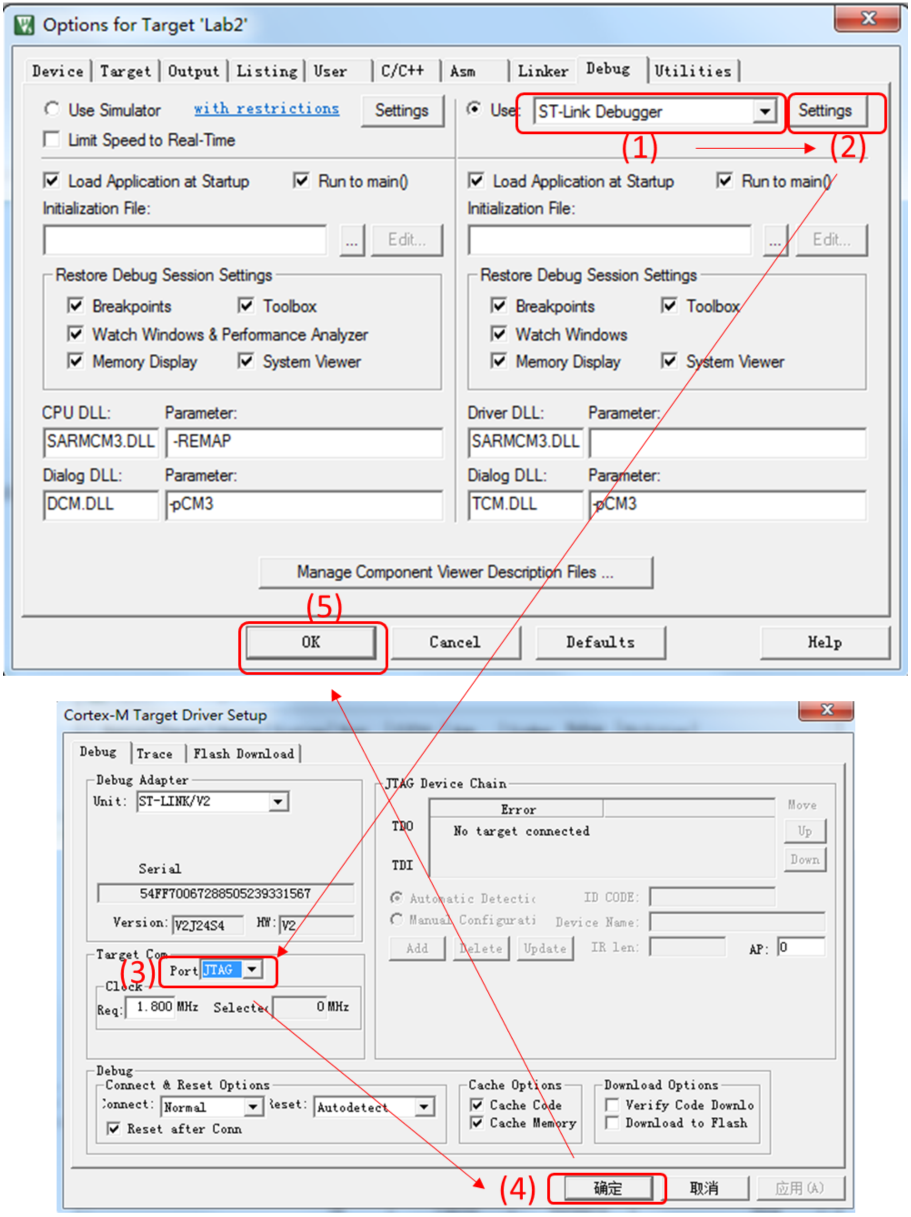


Figure 12. ST-Link connection

- Click , and set as shown in Figure 13.



Options for Target 'Lab2'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

○ Use Simulator with restrictions Settings
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display ☑ System Viewer
CPU DLL: SARMCM3.DLL Parameter: -REMAP
Dialog DLL: DCM.DLL Parameter: -pCM3

● Use: ST-Link Debugger Settings
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows
☑ Memory Display ☑ System Viewer
Driver DLL: SARMCM3.DLL Parameter:
Dialog DLL: TCM.DLL Parameter: pCM3

Cortex-M Target Driver Setup

Debug | Trace | Flash Download

Debug Adapter
Unit: ST-LINK/V2
Serial 54FF70067288505239331567
Version: V2J24S4 HW: V2
Target Com
Port JTAG
Clock
Req: 1.800 MHz Selecte 0 MHz

JTAG Device Chain
TDO Error
No target connected
TDI
● Automatic Detecti ID CODE:
○ Manual Configurati Device Name:
IR len: AP: 0

Debug
Connect & Reset Options
Connect: Normal Reset: Autodetect
☑ Reset after Conn
Cache Options
☑ Cache Code
☑ Cache Memory
Download Options
□ Verify Code Downlo
□ Download to Flash

Figure 13. ST-Link setup

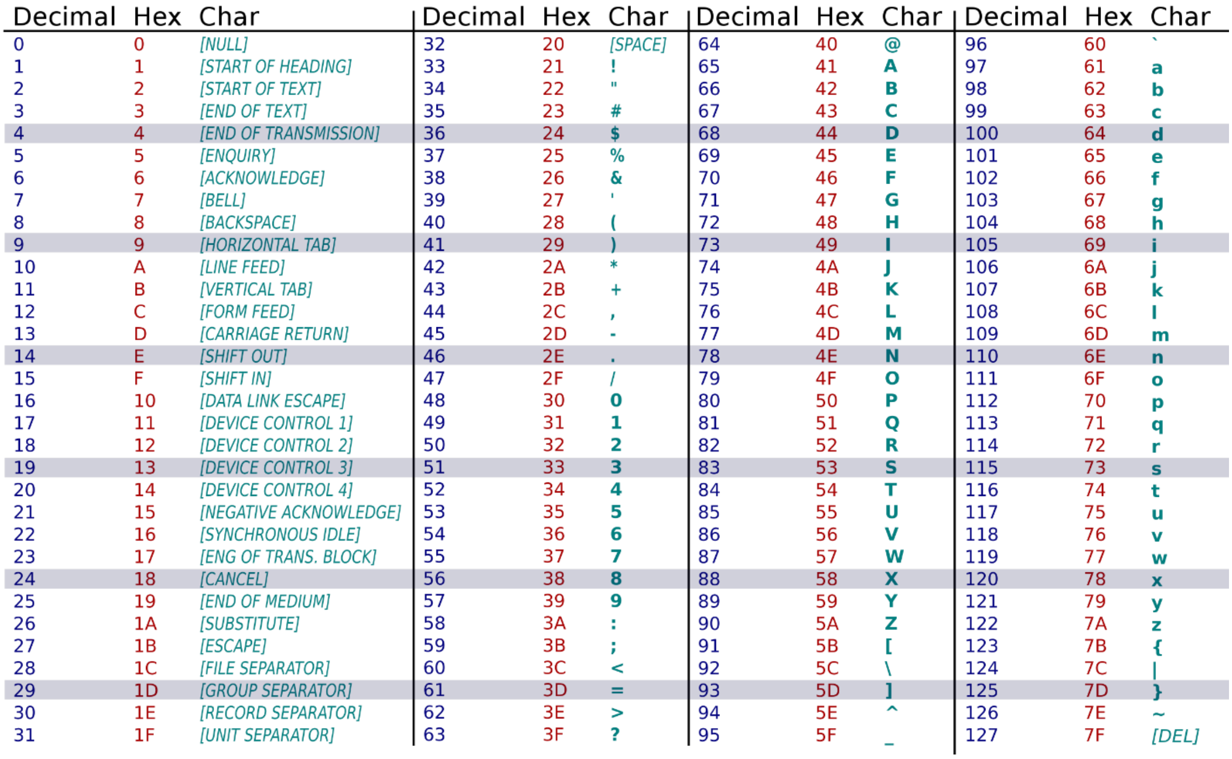
- After rebuilding by , you can directly download the code by clicking LOAD. You do not need to use FlyMCU.exe.

Procedures:

1) Modify “EIE3810_clock_tree_init()” subroutine to add APB2 with 72MHz.
2) Based on “EIE3810_USART2_init()”, create a new subroutine “EIE3810_USART1_init()”, and set APB2 to be at 72MHz. PA9 and PA10 are Tx and Rx of USART1. Set up USART1 to 9600 bps, 8N1 (Data: 8 bit, Parity: None, Stop: 1 bit).
3) Select Method 1 or Method 2 to use USART1.
4) Send out your CUHK(SZ) student ID once via USART1 approximately one second after the “RESET” key is released. Use an oscilloscope to observe PA9. As the ID number should be viewed as a character in eagleCom, you can take a reference on the following ASCII table (Figure 14) on what data to put into the data register of USART1.

[Demonstration] Demonstrate, to instructor, TA, or technician, that your program works.

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



Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Figure 14. ASCII Table

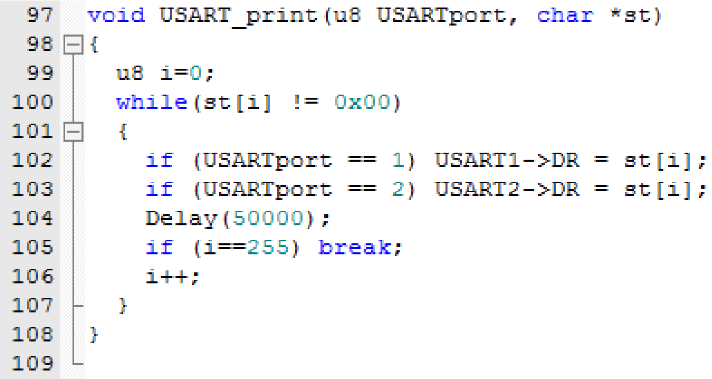
3.3. Experiment 3: Send a string to USART with checking TXE bit

In Experiment 2, we send characters one by one, using Delay(). When the data has been transferred to the shift register, TXE in USART_SR register will be set to “1”. Checking this bit is better than using Delay() for sending the next character. For more information on this register, check page 819 of RM0008.

To indicate the end of a series of characters (string), a character value 0x00 is used at the end of a string. It means that the string “ABC” in Hex format is represented by “0x41 0x42 0x43 0x00”. Check the location of 0x00, then we can know the end of the string. A subroutine named “USART_print()” is shown in Figure 18. It can copy a string to USART1 or USART2.

Procedure:

1) Input the subroutine USART_print() into main.c (Figure 15).
2) Remember to add the declaration of USART_print() at the top of the program.
3) Remove the remark of the line with USART_print() in main().
4) Compile and download it to your board. Try to understand the function of USART_print().
5) You should observe the characters “1234567890” in eagleCom.
6) Replace Delay(50000) in USART_print, with your codes, which checks the TXE register bit.



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

Figure 15. USART_print()

[Demonstration] Demonstrate, to instructor, TA, or technician, that your program works.

[In Report] Include your test result in the eagleCom. Draw a flowchart of USART_print() after your revision in step 5.

3.4. Experiment 4: Create your CLOCK and USART libraries for the project board

In Lab 1, you created some library files for keys, LEDs, and buzzers. Now make some new library files for the project board USART.

Procedure:

1) Modify your program from Experiment 3, and create files EIE3810_Clock.c and EIE3810_USART.c. Move the functions of clock_tree and USART to those files.
2) Create files EIE3810_Clock.h and EIE3810_USART.h. Add the declarations of corresponding functions to them. Store those .c and .h file to the folder “Board”. Hints: both .c files should include “stm32f10x.h”
3) Add those .c files to Manage Project Items.
4) Only keep “main()” and “Delay()” in “main.c”.
5) Approximately one second after the “RESET” key is released, send “1234567890” from USART1, and your student ID from USART2 once.
6) Compile and download. Open 2 eagleCom to monitor USART1 and USART2 simultaneously.

[Demonstration] Demonstrate, to instructor, TA, or technician, that your program works.

[In Report] Include your test result in the eagleCom.

4. Lab Report and Source Code

Submit the report softcopy and your code (complete project folder of each experiment) in zip format to Blackboard by the deadline below:

- L01: 15:00, Tuesday, October 17, 2023
- L02: 09:00, Friday, October 20, 2023

Each day of late submission will result in 10% deduction in the report and source code raw marks.