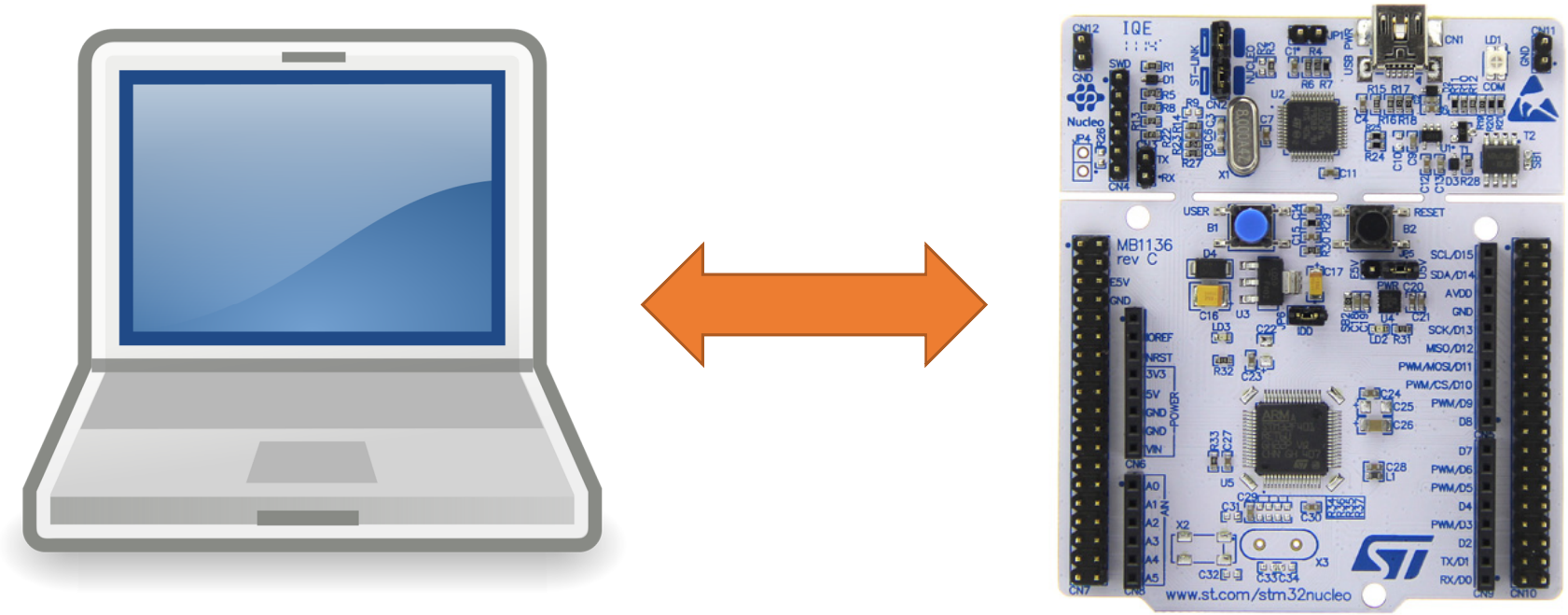
Lesson 6
USART

Lecturer: Harvard Tseng

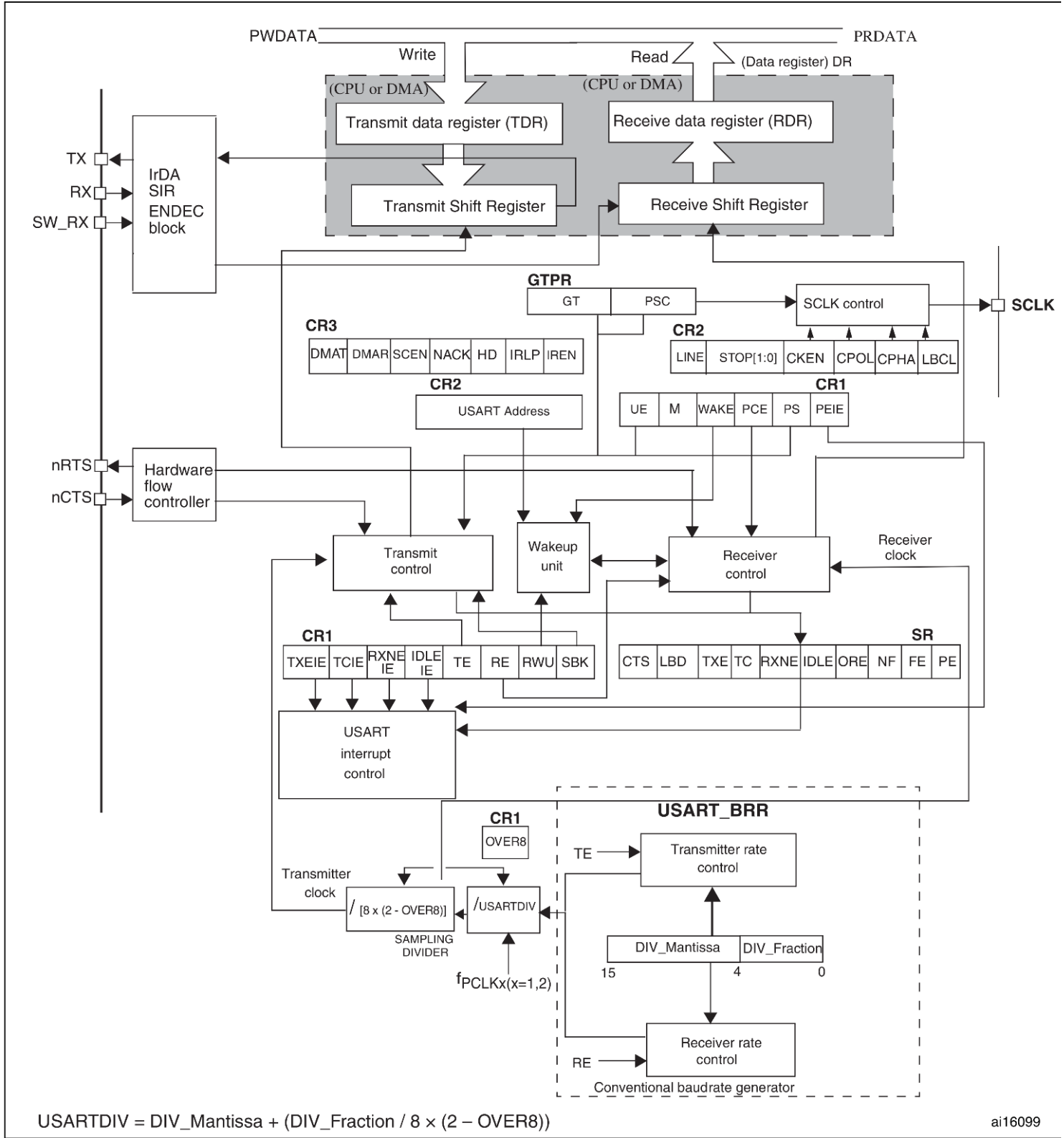
Communication is important!

- MCU to MCU
- MCU to PC



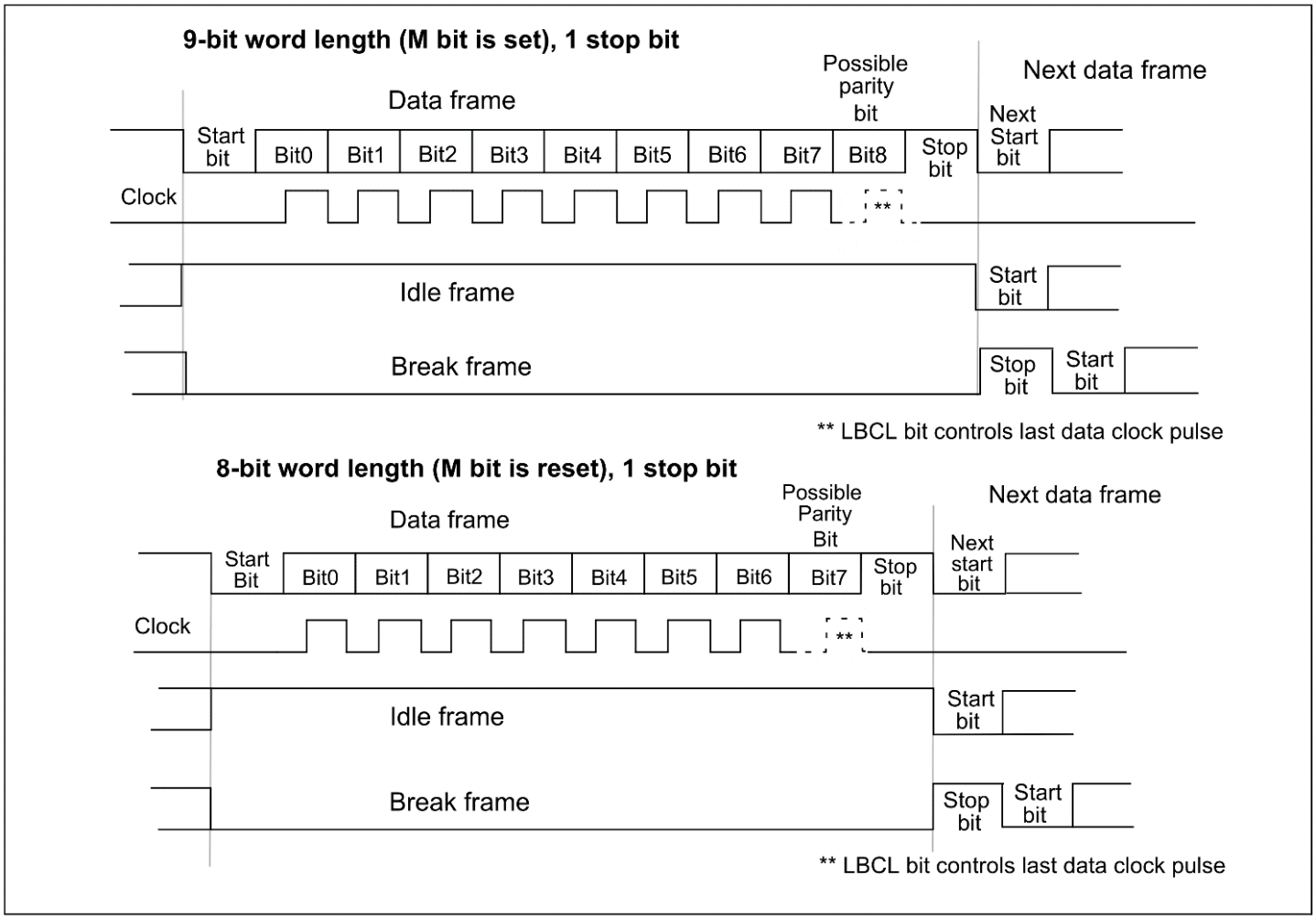
USART

Universal Synchronous Asynchronous Receiver Transmitter



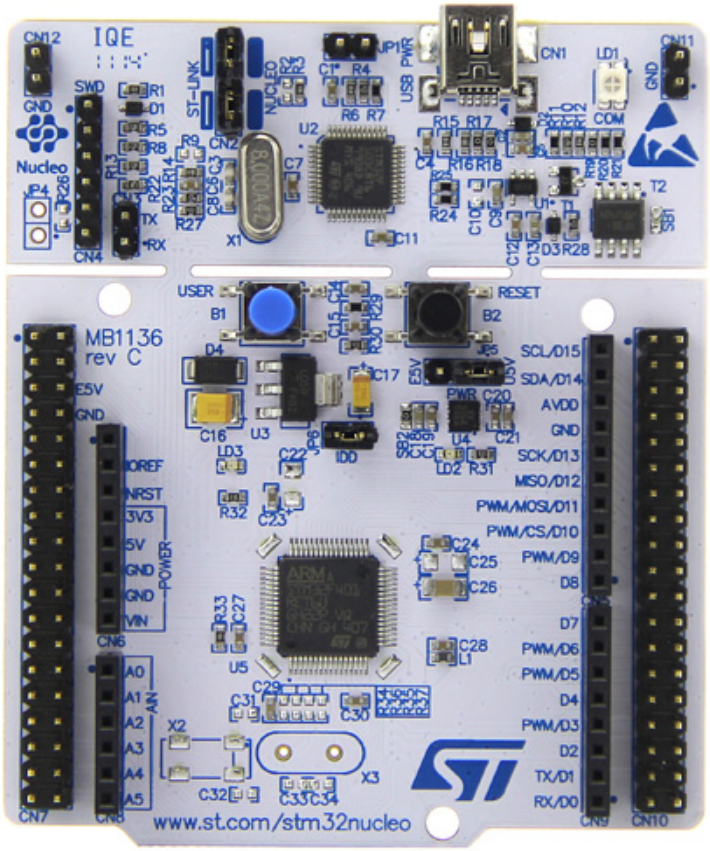
USART Character Description

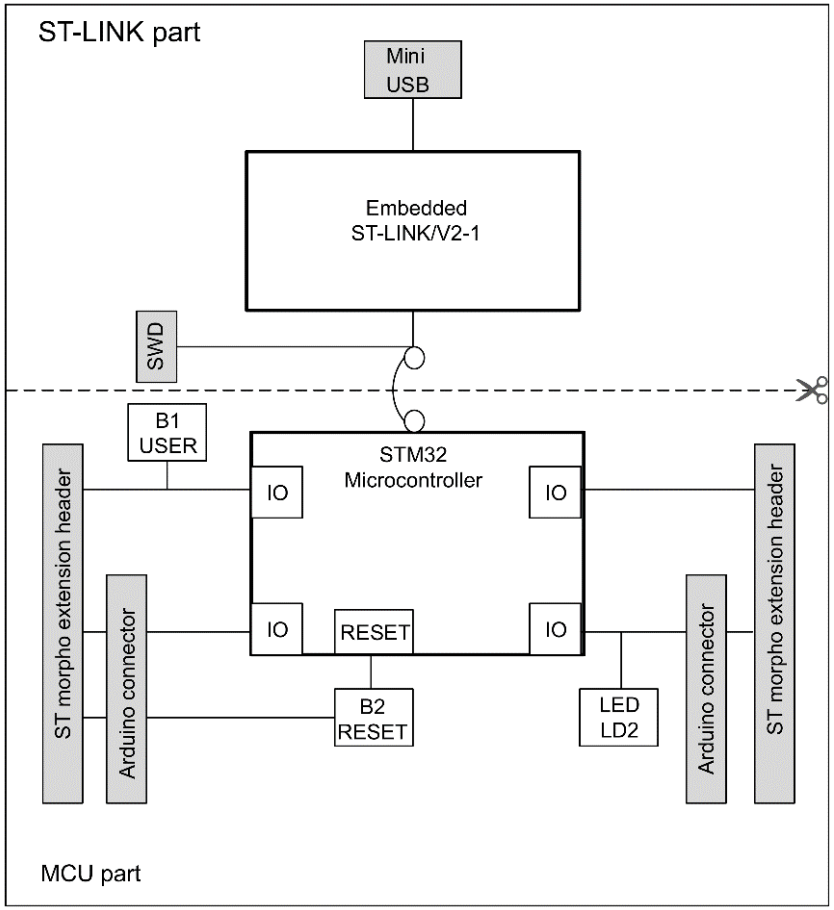
- A data word length can be either 8 or 9 bits.



Development Board

- USART2 is connected to ST-LINK using PA2, PA3.





MCU Pinouts

- Alternate function mapping

Port		AF00	AF01	AF02	AF03	AF04	AF05	AF06	AF07	AF08	AF09
		SYS_AF	TIM1/TIM2	TIM3/ TIM4/ TIM5	TIM9/ TIM10/ TIM11	I2C1/I2C2/ I2C3	SPI1/SPI2/ I2S2/SPI3/ I2S3/SPI4	SPI2/I2S2/ SPI3/ I2S3	SPI3/I2S3/ USART1/ USART2	USART6	I2C2/ I2C3
Port A	PA0	-	TIM2_CH1/ TIM2_ETR	TIM5_CH1	-	-	-	-	USART2_ CTS	-	-
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_ RTS	-	-
	PA2	-	TIM2_CH3	TIM5_CH3	TIM9_CH1	-	-	-	USART2_ TX	-	-
	PA3	-	TIM2_CH4	TIM5_CH4	TIM9_CH2	-		-	USART2_ RX	-	-
	PA4	-	-	-	-	-	SPI1_NSS	SPI3_NSS/ I2S3_WS	USART2_ CK	-	-
	PA5	-	TIM2_CH1/ TIM2_ETR	-	-	-	SPI1_SCK	-	-	-	-
	PA6	-	TIM1_BKIN	TIM3_CH1	-	-	SPI1_ MISO	-	-	-	-
	PA7	-	TIM1_CH1N	TIM3_CH2	-	-	SPI1_ MOSI	-	-	-	-
	PA8	MCO_1	TIM1_CH1	-	-	I2C3_SCL	-	-	USART1_ CK	-	-

USART Setting

RCC	• Enable GPIOA & USART2
GPIO	• Set PA2, PA3 as alternate mode
NVIC	• IRQ setting • Enable USART2_IRQn
USART	• Register setting • Enable USART

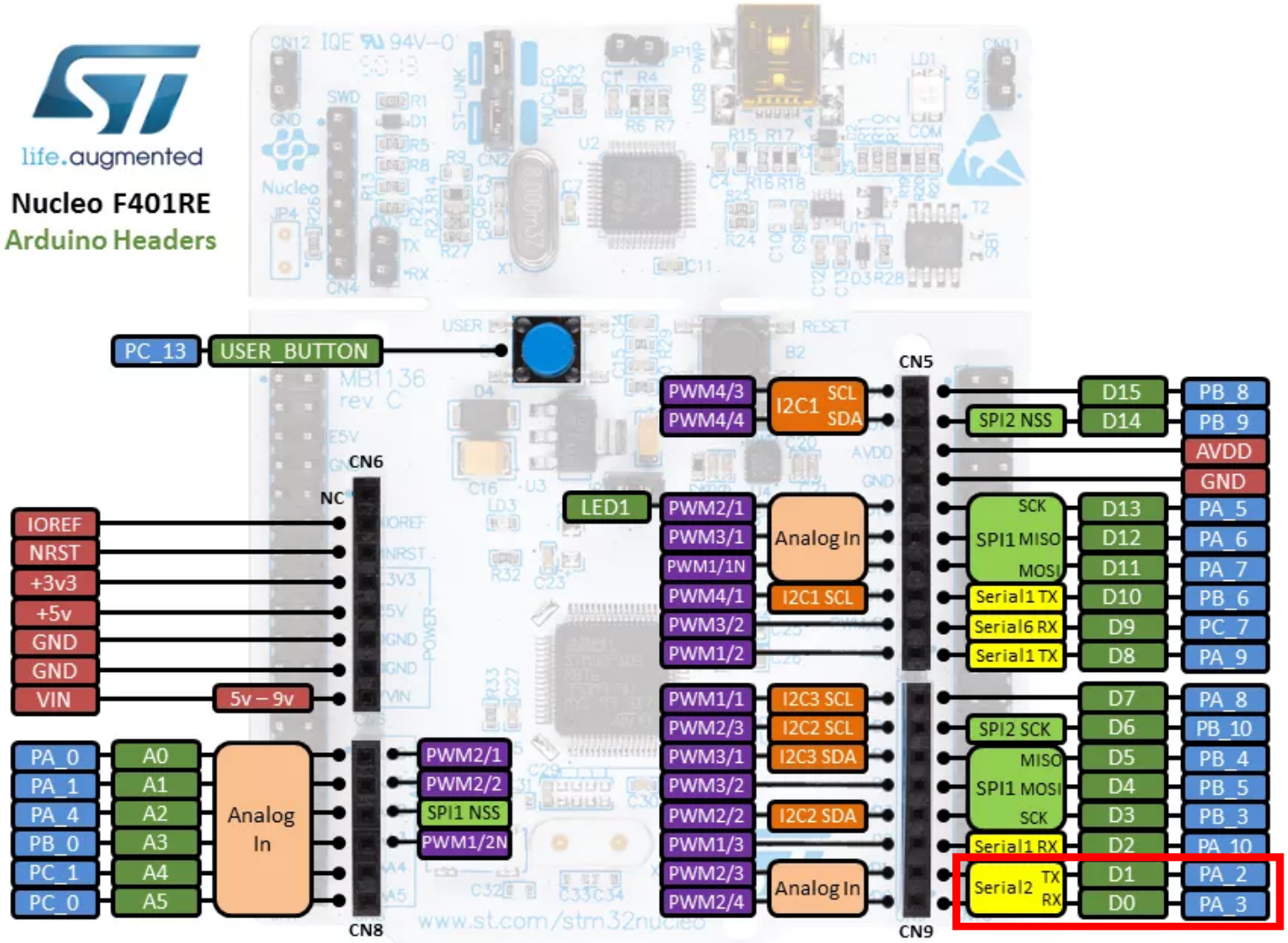
RCC Setting

- RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
- RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

GPIO Setting

- GPIO_PinAFConfig(GPIOA, GPIO_PinSource2, GPIO_AF_USART2);
- GPIO_PinAFConfig(GPIOA, GPIO_PinSource3, GPIO_AF_USART2);
-
- GPIO_InitTypeDef GPIO_InitStruct;
- GPIO_InitStruct.GPIO_Pin = GPIO_Pin_2 | GPIO_Pin_3;
- GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF;
- GPIO_Init(GPIOA, &GPIO_InitStruct);

Board pinout



USART Configuration

- BaudRate : 9600, 19200, …
- WordLength : 8bits, 9bits.
- StopBits : 0.5, 1, 1.5, 2.
- Parity : No, Even, Odd.
- Mode : Rx, Tx.
- HardwareFlowControl : None, RTS, CTS, RTS_CTS.

Using std library

- void USART_Init(USART_TypeDef* USARTx, USART_InitTypeDef* USART_InitStruct);
- void USART_Cmd(USART_TypeDef* USARTx, FunctionalState NewState);
- void USART_SendData(USART_TypeDef* USARTx, uint16_t Data);
- uint16_t USART_ReceiveData(USART_TypeDef* USARTx);
- FlagStatus USART_GetFlagStatus(USART_TypeDef* USARTx, uint16_t USART_FLAG);

Status Register (USART_SR)

- TXE: Transmit data register empty
  - 0: Data is not transferred to the shift register
  - 1: Data is transferred to the shift register
- RXNE: Read data register not empty
  - 0: Data is not received
  - 1: Received data is ready to be read

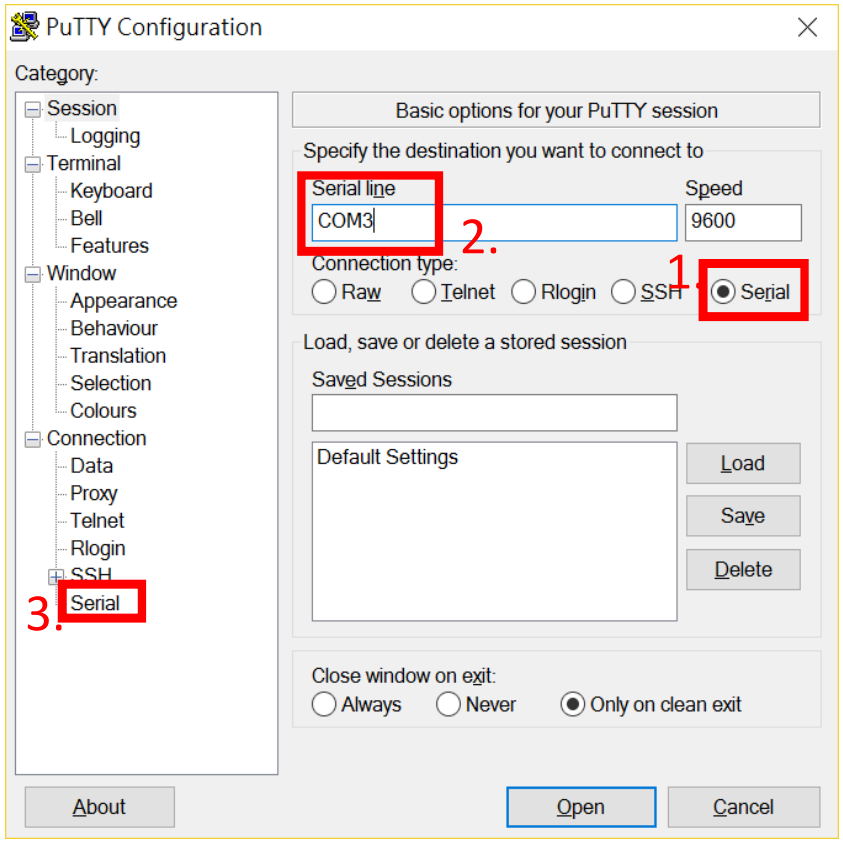
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

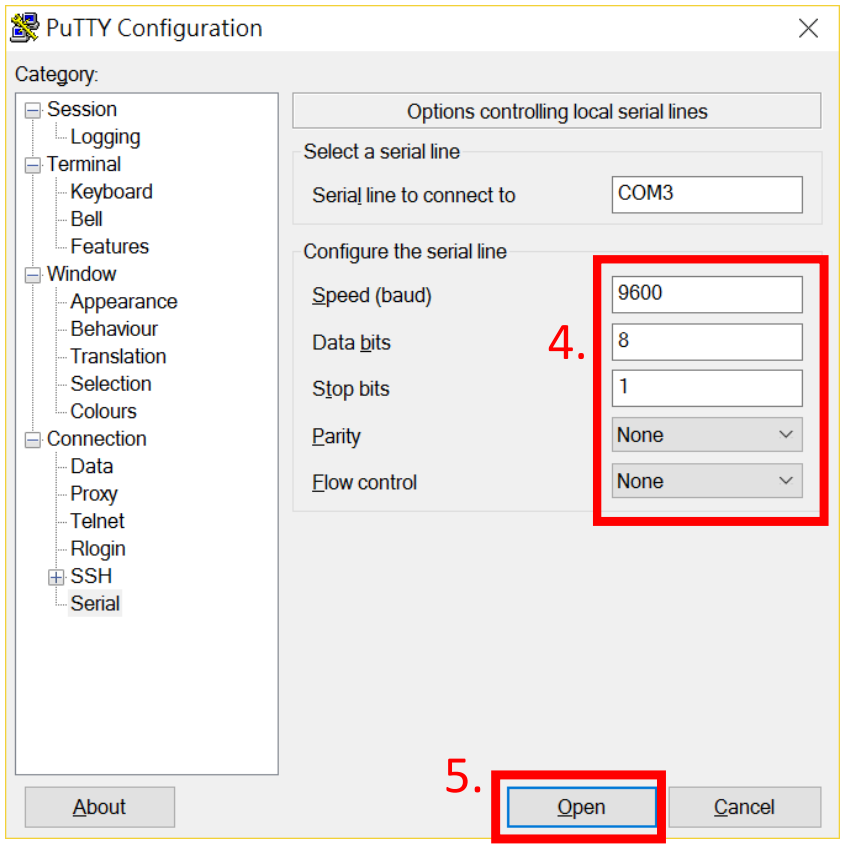
PuTTY

Free and open-source terminal emulator.

Serial console and network file transfer application.

PuTTY Configuration



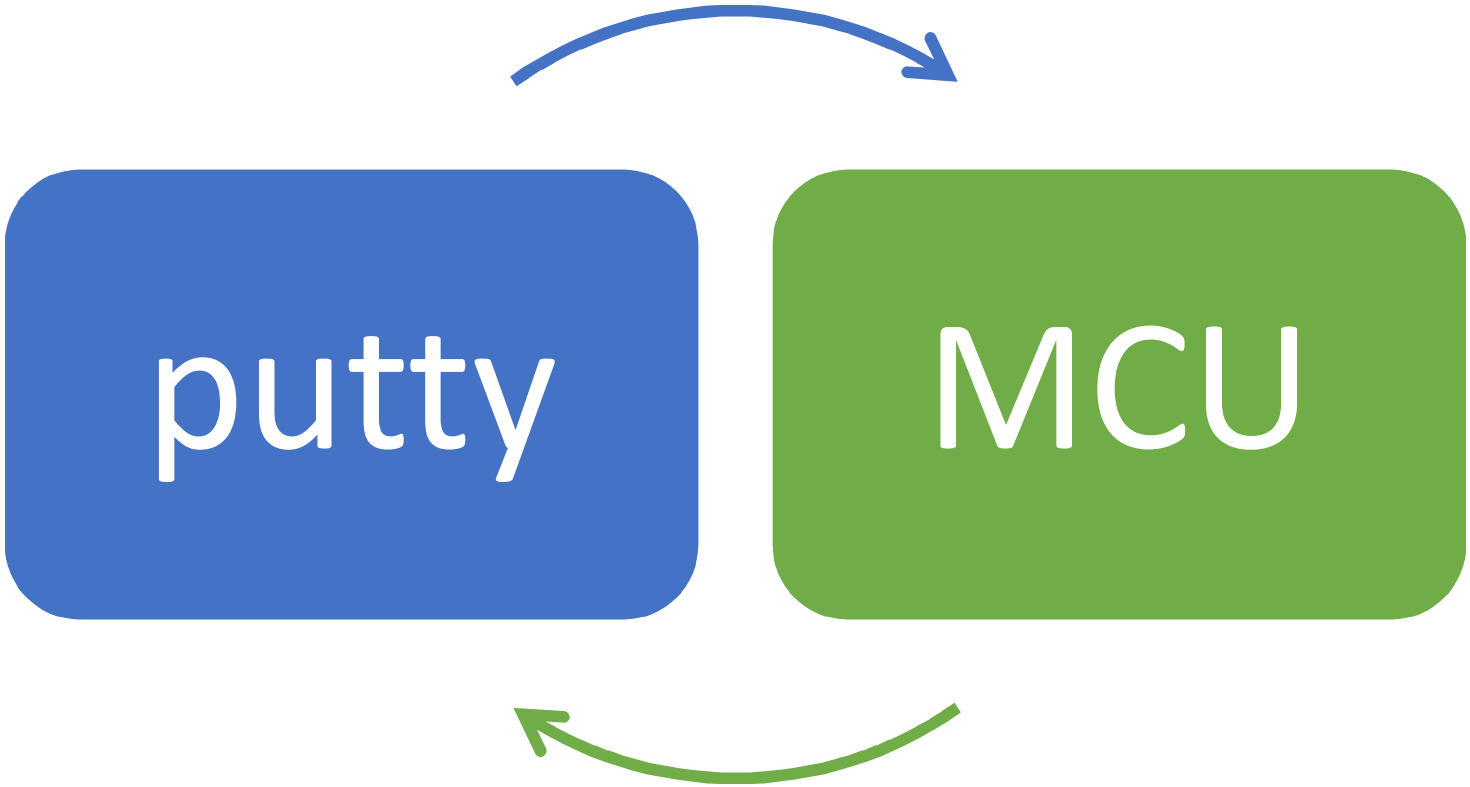


Special ASCII

- 0x0A(\n) Line Feed
  - Change to next line.
- 0x0C(\f) Form Feed
  - Next page.
- 0x0D(\r) Carriage Return
  - Go back to first character.

Exercise 1

- Use polling method to send each character one at a time.
- Create a function called putty_Begin(int baud_rate).



Exercise 2

- Change LED frequency 1~9Hz by putty.
- If button pressed, LED turn off and clear scrollback.
  - Hints: Send special ASCII to clear.