Introduction to Electric Vehicles

Serial Communication

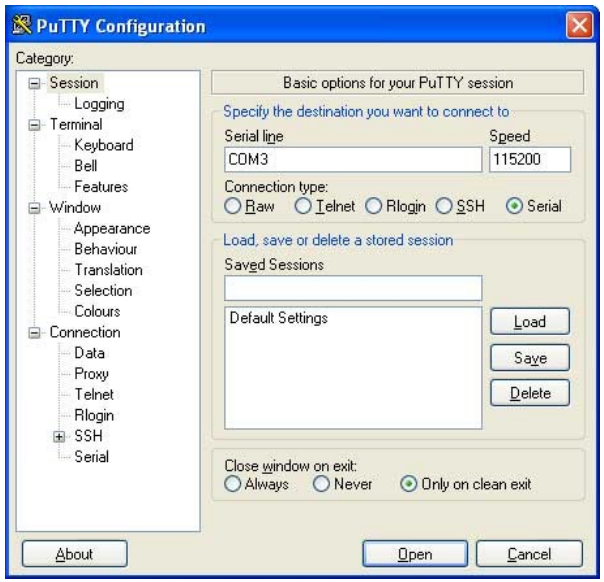
Joonki Hong

joonki@cad4x.kaist.ac.kr

KAIST

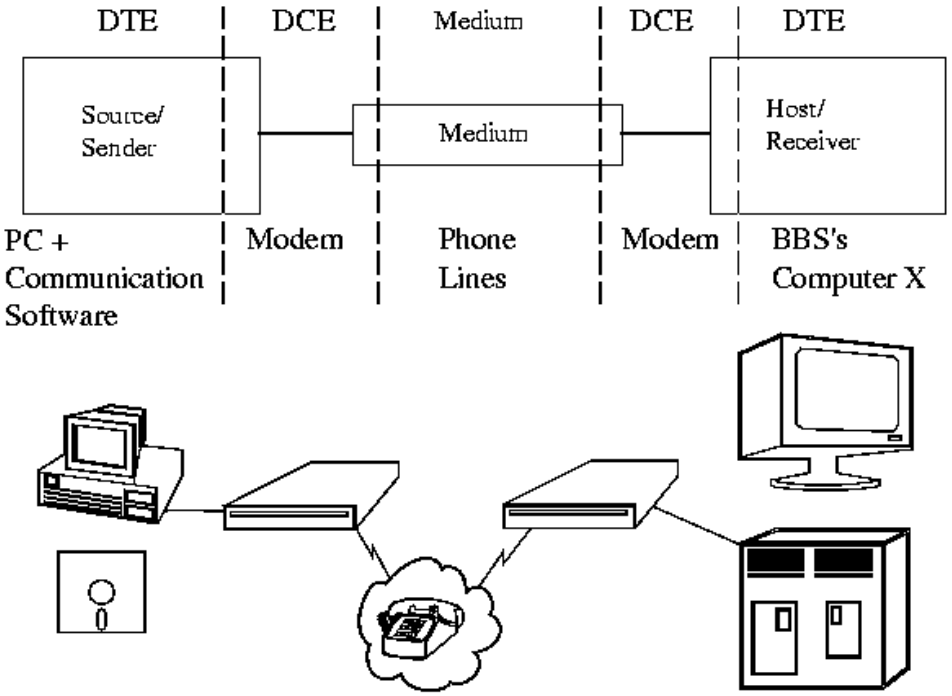
Terminal and Terminal Emulator

- Dummy terminal
  - Device which consists of keyboard and a monitor
  - No intelligence
- Terminal emulator
  - Executed on the same machine or on a different one
    - via telnet
    - via ssh
    - via dial-up
  - Supports
    - Local echo
  - Why we use terminal emulator?
    - Embedded system has limited resources
      - No display device
    - Send result to host
      - Show result in a terminal emulator



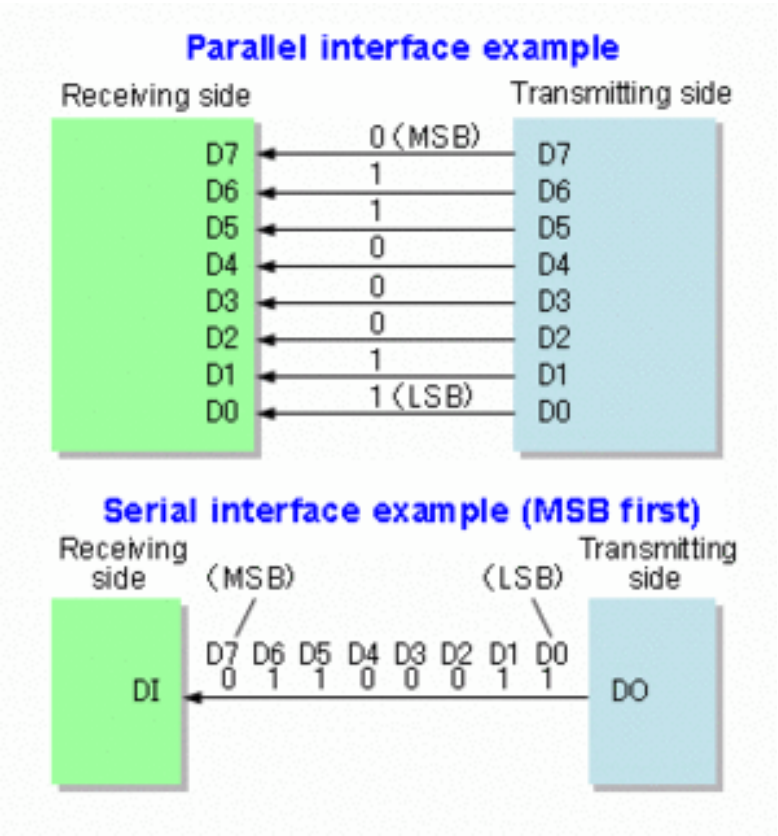
DTE and DCE

- Data terminal equipment (DTE)
  - End instrument that converts user information into signal or reconverts received signals
- Data circuit-terminating equipment (DCE)
  - Device between DTE and a data transmission circuit



Communication

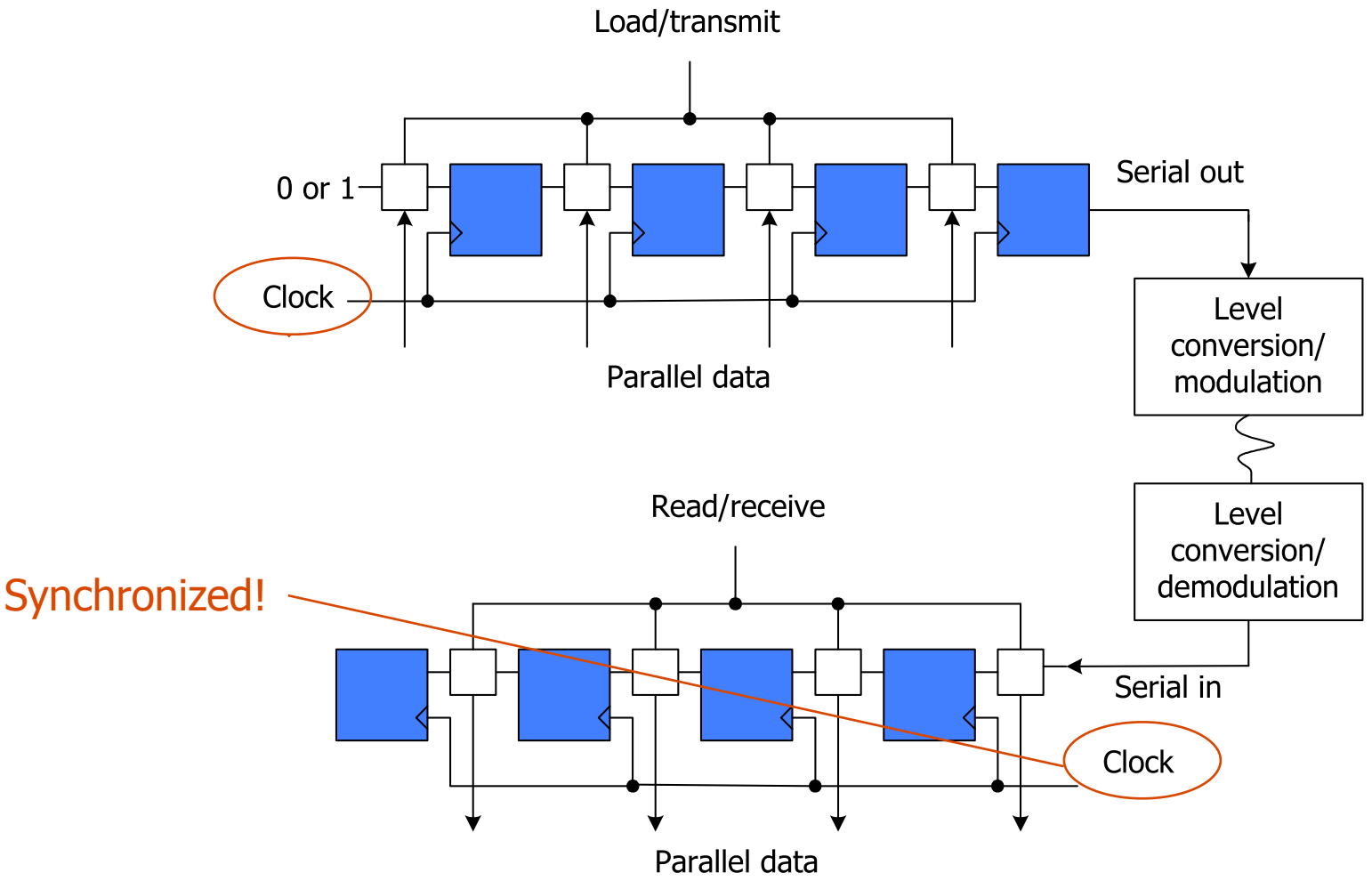
- Communication
  - Data transmission between two different domains
- Parallel communication
  - Transmit and receive a byte or word data simultaneously with parallel link
- Serial communication
  - Process of sending data one bit at a time
    - Converte parallel data to serial data
    - Transmit serialized data
    - Reconvert serialized data to parallel data
  - Pros
    - Low wire cost
    - Low signal conditioning cost
      - Crosstalk is less of an issue
  - Cons
    - Speed penalty
      - overcome by high bit rate



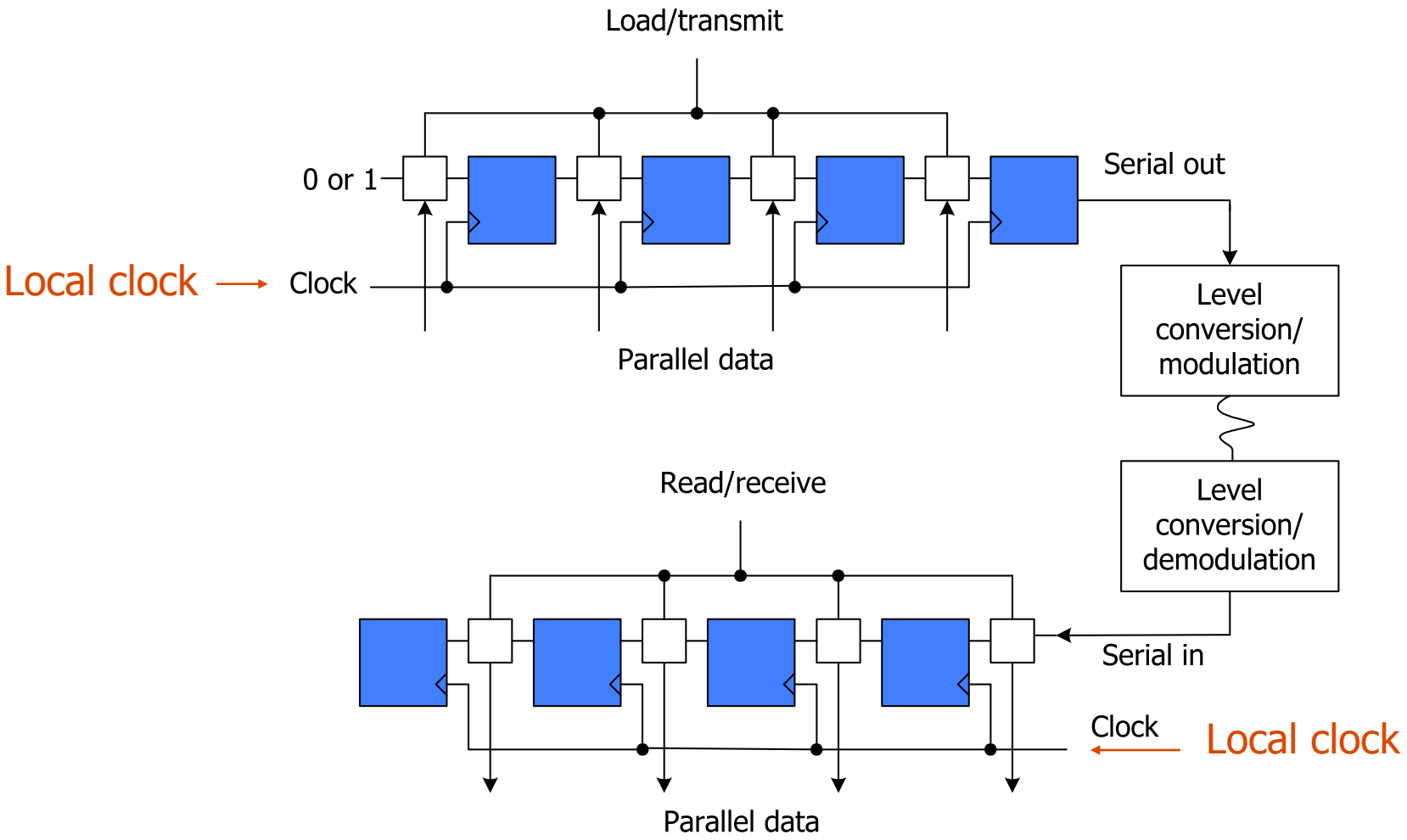
Serial communication

- Transmit bits in a single channel
  - Simplex communication
  - Half-duplex
  - Full-duplex
- A sequence of bit can be packet or character
  - ASCII code
    - 7 bit for 128 characters
  - Binary code
    - Fixed length or not
- Synchronous and asynchronous communication

Synchronous communication



Asynchronous Communication

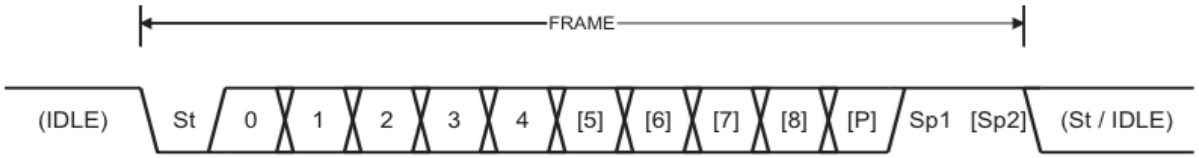


UART (1)

- Universal asynchronous receiver and transmitter
  - converts bytes of data and transmit the individual bits
  - Highly flexible serial communication device
  - Commonly used in conjunction with communication standards such as RS-232
- Why we use the UART?
  - Primitive to easy to implement on chip
  - Simple to write code in embedded system
    - No OS support required
    - No external device required like PHY
    - Almost every PC has UART port
    - Good for debugging
      - Print message

UART (2)

- Frame format
  - 1 start bit
  - 5, 6, 7, 8 or 9 data bits
  - no, even or odd parity bit
  - 1 or 2 stop bits



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

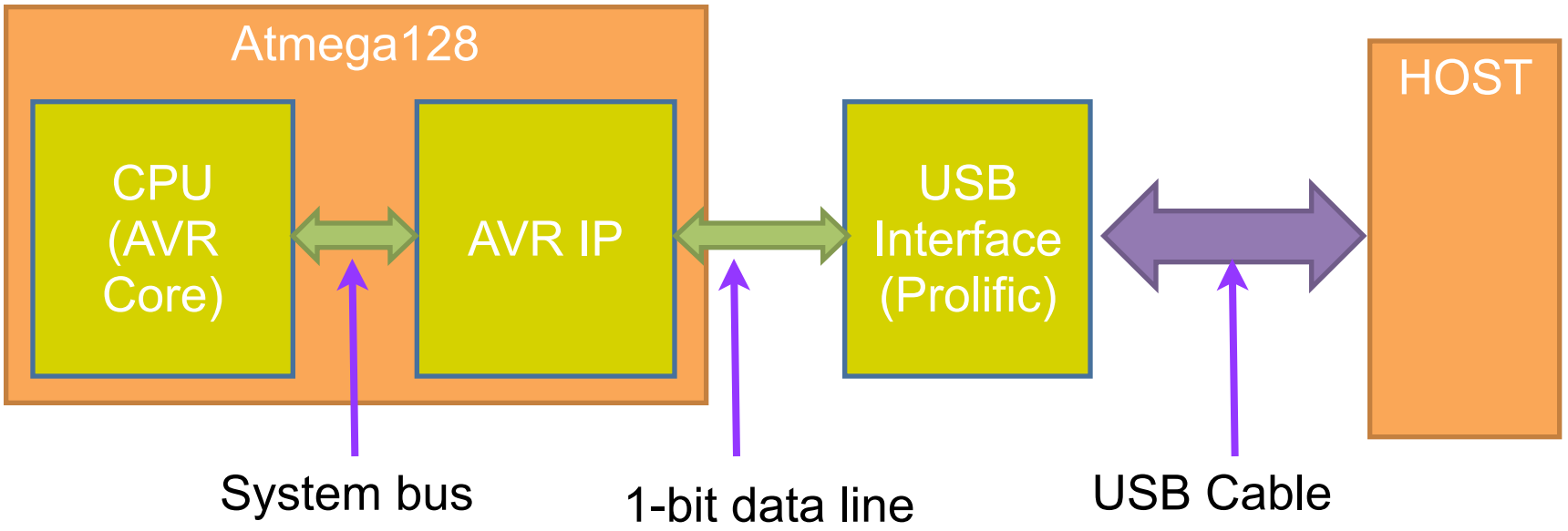
UART (3)

- Frame
  - Bit time
    - The period of time required to serially transmit or receive 1 bit of data (1 cycle of the baud rate frequency)
  - Start bit
    - The bit time of a logic 0 that indicates the beginning of a data frame
    - A start bit begin with a 1-to-0 transition, and is preceded by at least 1 bit time of logic 1
  - Stop bit
    - 1 bit time of logic 1 that indicates the end of a data frame
  - Break
    - A frame in which all of the data bits, including the stop bit, is logic 1
  - Parity bit
    - Parity bit is calculated by doing an exclusive-or of all the data bits.

UART (4)

- Frame error
  - An error condition that occurs when the stop bit of a received frame is missing
  - A framing error is always present on the receiver side when the transmitter is sending BREAKs
- Parity error
  - An error condition that occurs when the calculated parity of the received data bits in a frame does not match the parity bit received on the receiver side
- Overrun error
  - An error condition that occurs when the latest character received is ignored to prevent overwriting a character already present in the UART buffer

HW Block Diagram



What to do

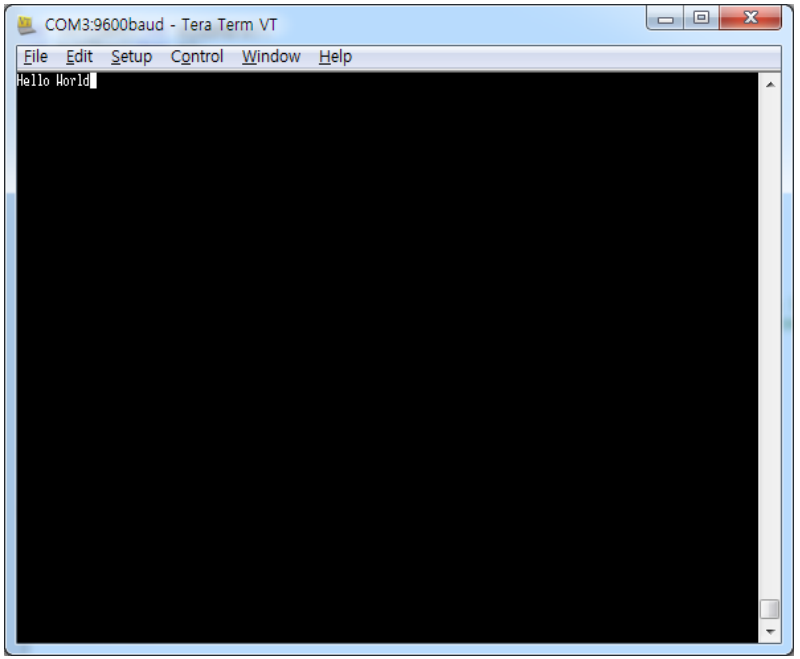
- Setup UART register - Read USART chapter on data sheet pp. 177 - 203
  - Baud rate: 9600 bps
  - Stop bit: 1bit
  - Parity bit: none
  - Flow control: none
  - Rx enable
  - Tx enable
- Write getChar
  - Wait until buffer is empty
  - Get one character
- Write putChar
  - Wait until buffer is empty
  - Put one character
- Write “Hello world!” to terminal

Terminal Program

- You may use
  - Teraterm
    - http://ttssh2.sourceforge.jp/
  - Realterm
    - http://realterm.sourceforge.net/
  - Hyperterminal
  - Minicom
  - ZOC6

Homework (Project) #1

- Printed report is required
  - Until next Friday class (March 27th)
  - No delayed paper accepted
- Contents should include
  - Source code
  - Detailed description of setting register
    - Why you set those registers
    - Why you put those values into those register
  - Result
    - Screen shot of terminal emulator



Result_Joonki (Source code & description)

- USART register description
  - USART0 I/O data register - UDR0
    - Bit 7:0 - RxBn7:0: Receive data buffer (read access)
    - Bit 7:0 - TxBn7:0: Transmit data buffer (write access)

Bit	7	6	5	4	3	2	1	0	
	RXB0[7:0]								UDR0 (Read)
	TXB0[7:0]								UDR0 (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

  - USART0 baud rate registers - UBRR0K and UBRR0H
    - Bit 15:12 - Reserved bits
    - Bit 11:0 - UBRRn11:0: USARTn baud rate register

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR0[11:8]				UBRR0H
	UBRR0[7:0]								UBRR0L
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Result_Joonki (Source code & description)

- USART register description
  - USART0 control and status register A - USCR0A
    - Bit 7 - RxCn: USARTn receive complete
    - Bit 6 - TxCn: USARTn transmit complet
    - Bit 5 - UDREn: USARTn data register empty
    - Bit 4 - FEn: Frame error
    - Bit 3 - DORn: Data overRun
    - Bit 2 - UPEn: USARTn parity error
    - Bit 1 - U2Xn: Double the USARTn Transmission speed
    - Bit 0 - MPCMn: Multi-processor Communication Mode

Bit	7	6	5	4	3	2	1	0	
	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	UCSR0A
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Result_Joonki (Source code & description)

- USART register description
  - USART0 control and status register B - USCR0B
    - Bit 7 - RxCIEnn: Rx complete interrupt enable
    - Bit 6 - TxCIEn: Tx complete interrupt enable
    - Bit 5 - UDRIEn: USARTn data register empty interrupt enable
    - Bit 4 - RXENn: Receiver enable
    - Bit 3 - TXENn: Transmitter enable
    - Bit 2 - UCSZn2: Character size
    - Bit 1 - RXB8n: Receive data Bit 8
    - Bit 0 - TXB8n: Transmit data bit 8

Bit	7	6	5	4	3	2	1	0	
	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	UCSR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Result_Joonki (Source code & description)

- USART register description
  - USART0 control and status register C - USCR0C
    - Bit 7 - Reserved bit
    - Bit 6 - UMSELn: USARTn mode select
    - Bit 5:4 - UPMn1:0: Parity mode
    - Bit 3 - USBSn: Stop bit select
    - Bit 2:1 - UCSZn1:0: Character size
    - Bit 0 - UCPOLn: Clock polarity

Bit	7	6	5	4	3	2	1	0	
	–	UMSEL1	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPO1L	UCSR1C
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Result_Joonki (Result)

- Screenshot of terminal emulator