Smart Software Project

Lecture: Week 5

UART & Timer

Prof. HyungJune Lee
hyungjune.lee@ewha.ac.kr

이화여자대학교
EWHA WOMANS UNIVERSITY

Today

- Review from the last lecture
  - ATmega 2560 I/O Register
- Midterm: 2pm - 4:30pm, Mon Apr 11
- ATmega2560 microcontroller (MCU)
  - UART
  - Timer
- Announcement

Project Proposal Submission

- Project proposal per team
  - Due: 5pm on April 22, Friday
  - What to submit:
    - 1) Report hardcopy (printed report)
    - 2) Report softcopy (report file submission to Cyber Campus)
  - Where to submit:
    - HW box at "HyungJune Lee" in front of Asan 221-1
  - Language: English or Korean
  - Start discussing what your team will do with your team partner

이화여자대학교
EWHA WOMANS UNIVERSITY

Project Proposal

- Project name (in English)
- Project statement
  - Goals of your project
- Project description
  - What will your project be doing?
  - Key functions
- Contributions of your work to research or industry
- Related work
  - Any existing previous works (at least two) similar to your project
  - What's similar and different?
- System overview & architecture
  - Block diagram of main blocks
  - What each block is doing
  - How each block is connected to other blocks, i.e., interface
    - Any message is exchanging between blocks? e.g., request/reply, data, etc?

Project Proposal

- Development environment
  - Arduino Mega 2560-based SmartCAR
    (or others if any, e.g., Android device)
  - Androx Studio IDE (or others if any)
  - What kind of sensors are to be used in your project
  - Other information from the connection to Android device or other devices? (location, Internet, etc.)
- Verification procedure
  - How can you test if your project works properly as designed
  - Test cases
- What do you anticipate will be the easiest part of your project?
- What do you anticipate will be the most difficult part of your project?
- Detailed time plan
- References

Evaluation criteria

- Format requirement
  - 5 points
- Creativity
  - 5 points
- Clarity
  - 5 points
- Concreteness (of software architecture)
  - 5 points
- Implementability
  - 5 points

- Total score: 25 points

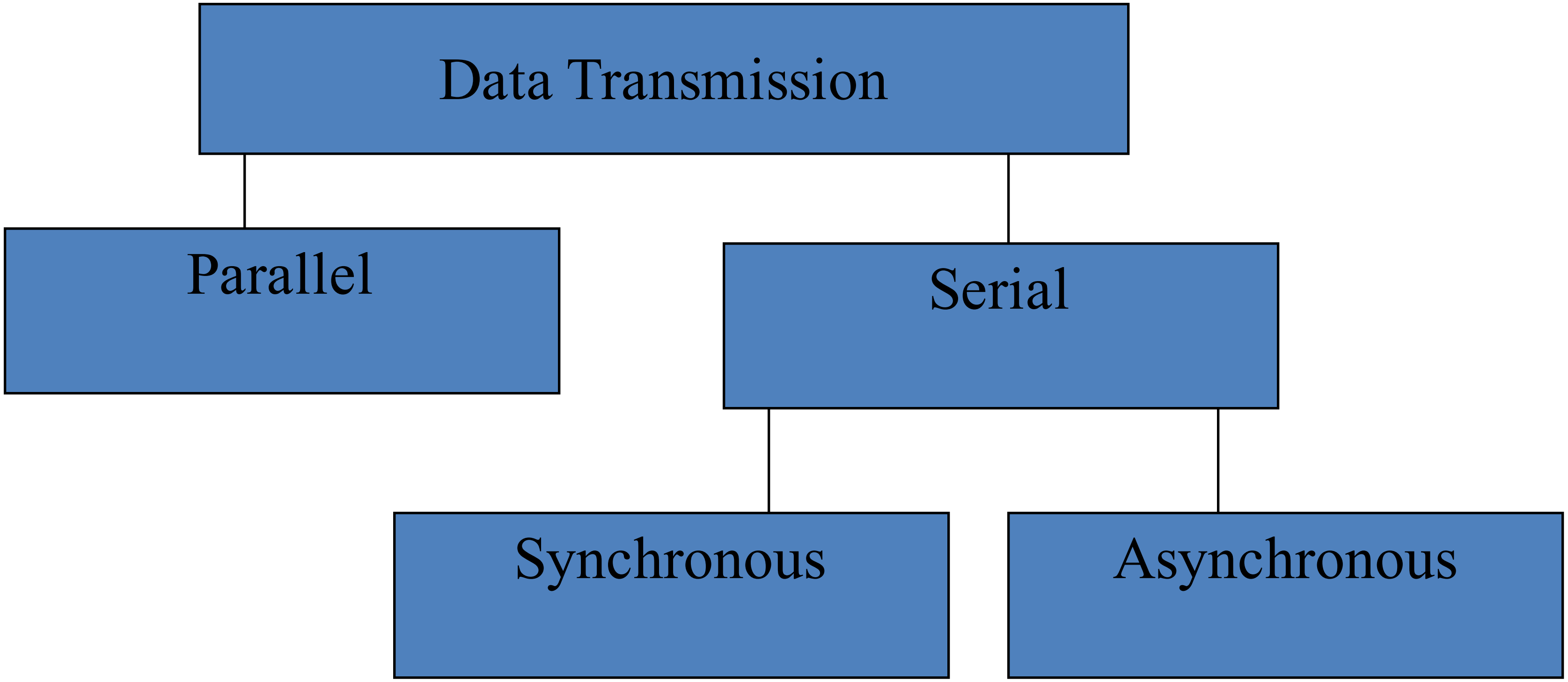
Class Schedule

Week	Lecture Contents	Lab Contents
Week 1	Course introduction	Arduino introduction: platform & programming environment
Week 2	Embedded system overview & source management in collaborative repository (using GitHub)	Lab 1: Arduino Mega 2560 board & SmartCAR platform
Week 3	ATmega2560 Micro-controller (MCU): architecture & I/O ports, Analog vs. Digital, Pulse Width Modulation	Lab 2: SmartCAR LED control
Week 4	Analog vs. Digital & Pulse Width Modulation	Lab 3: SmartCAR motor control (Due: HW on creating project repository using GitHub)
Week 5	ATmega2560 MCU: memory, I/O ports, UART	Lab 4: SmartCAR control via Android Bluetooth
Week 6	ATmega2560 UART control & Bluetooth communication between Arduino platform and Android device	Lab 5: SmartCAR control through your own customized Android app (Due: Project proposal)
Week 7	Midterm exam	
Week 8	ATmega2560 Timer, Interrupts & Ultrasonic sensors	Lab 6: SmartCAR ultrasonic sensing
Week 9	Infrared sensors & Buzzer	Lab 7: SmartCAR infrared sensing
Week 10	Acquiring location information from Android device & line tracing	Lab 8: Implementation of line tracer
Week 11	Gyroscope, accelerometer, and compass sensors	Lab 9: Using gyroscope, accelerometer, and compass sensors
Week 12	Project	Team meeting (for progress check)
Week 13	Project	Team meeting (for progress check)
Week 14	Course wrap-up & next steps	
Week 15	Project presentation & demo I (Due: source code, presentation slides, & poster slide)	Project presentation & demo II
Week 16	Final week (no final exam)	

Today

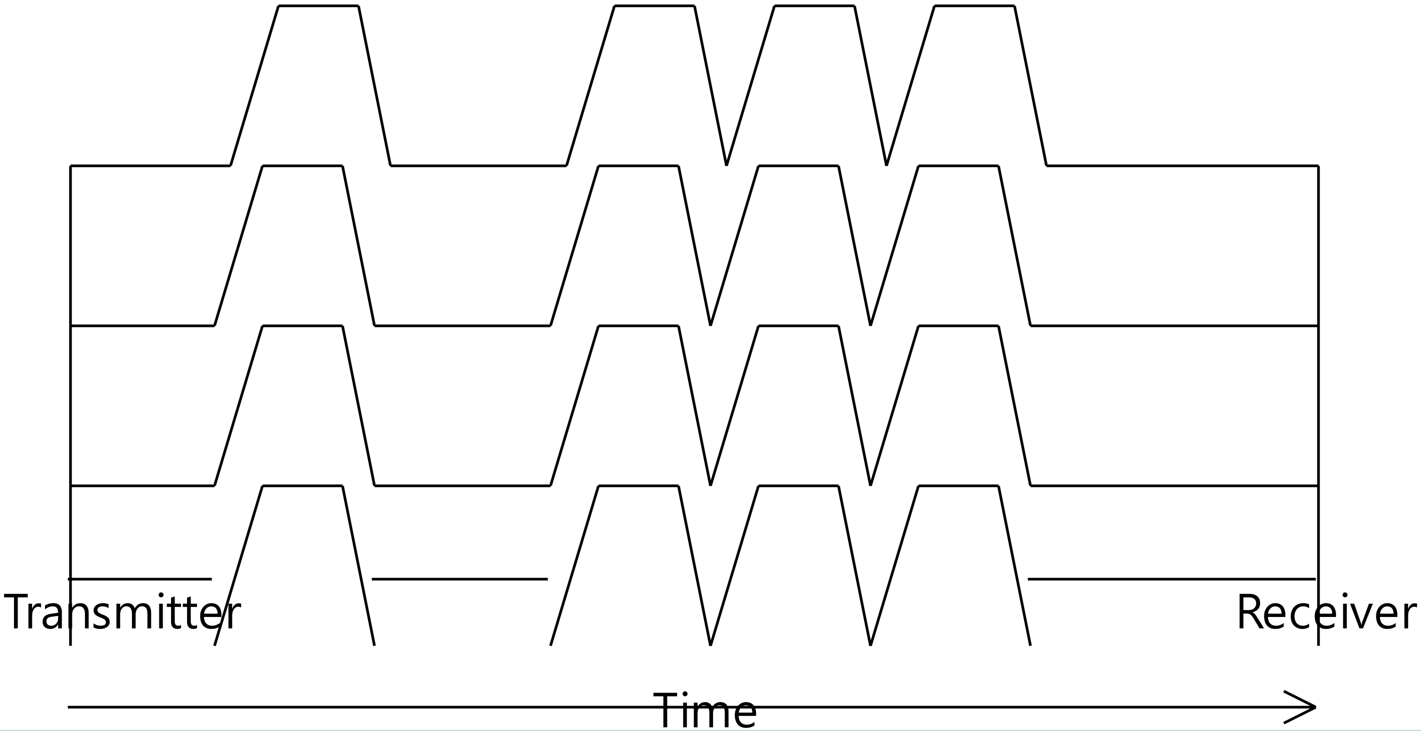
- Review from the last lecture
  - ATmega 2560 I/O Register
- Midterm: 2pm - 4:30pm, Mon Apr 11
- ATmega2560 microcontroller (MCU)
  - **UART**
  - Timer
- Announcement

Data Transmission Tree



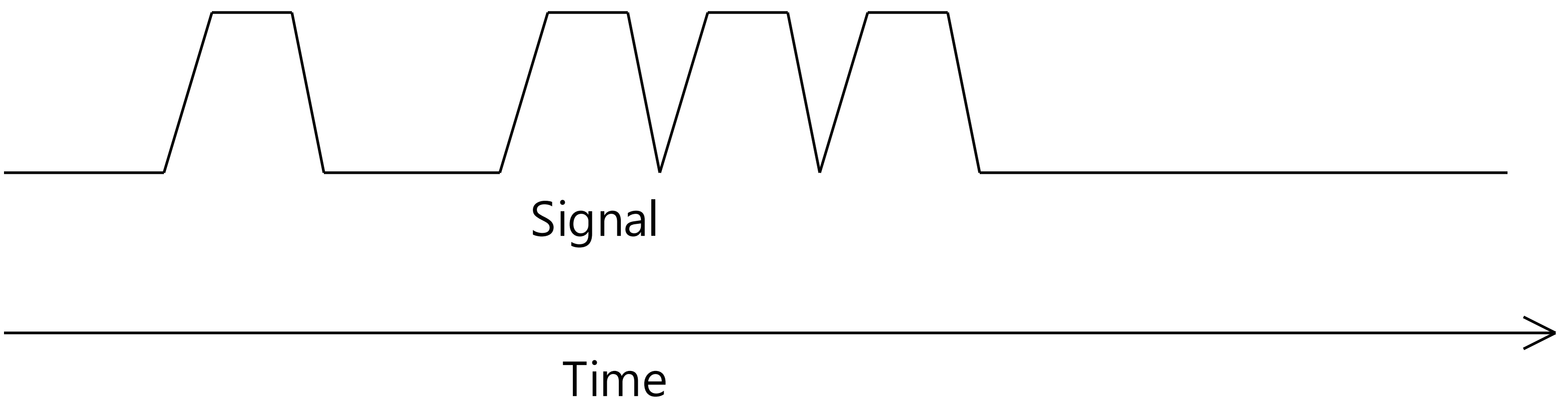
Definition: Parallel

- Data is sent and received more than one bit at a time
- Transmission on multiple wires
- Many lines of communication, synchronized bursts of data



Definition: Serial

- Data is sent and received one bit at a time
- Transmission on single wire
- One line of communication, long string of data



Serial Types: RS232, SCI, and SPI

- RS232
  - Typical computer COM port
- SCI
  - Serial Communication Interface, uses the universal asynchronous receiver/transmitter or UART
- SPI
  - Serial Peripheral Interface, part of Port B

Why Serial?

- Fewer wires translates to
  - Lower cost
  - Simpler set-up

Definition: Synchronous

- Sender and receiver have their clocks synchronized
- Transmissions occur at specified intervals

- Advantage:
  - Faster

Definition: Asynchronous

- Devices are not synchronized
- Transmissions happen at unpredicted intervals

- Advantages:
  – Simpler
  – More robust

Why Asynchronous?

- Disadvantage:
  - Slower due to overhead

- Advantages:
  - Simpler
  - Cheaper
  - Information can be sent when ready

Please Note:

- Both synchronous and asynchronous must have agreed upon bit transfer rate

FYI Term: "UART"

- Universal
- Asynchronous
- Receiver-
- Transmitter

"…a computer component that handles asynchronous serial communication."

www.webopedia.com

UART Character Frame

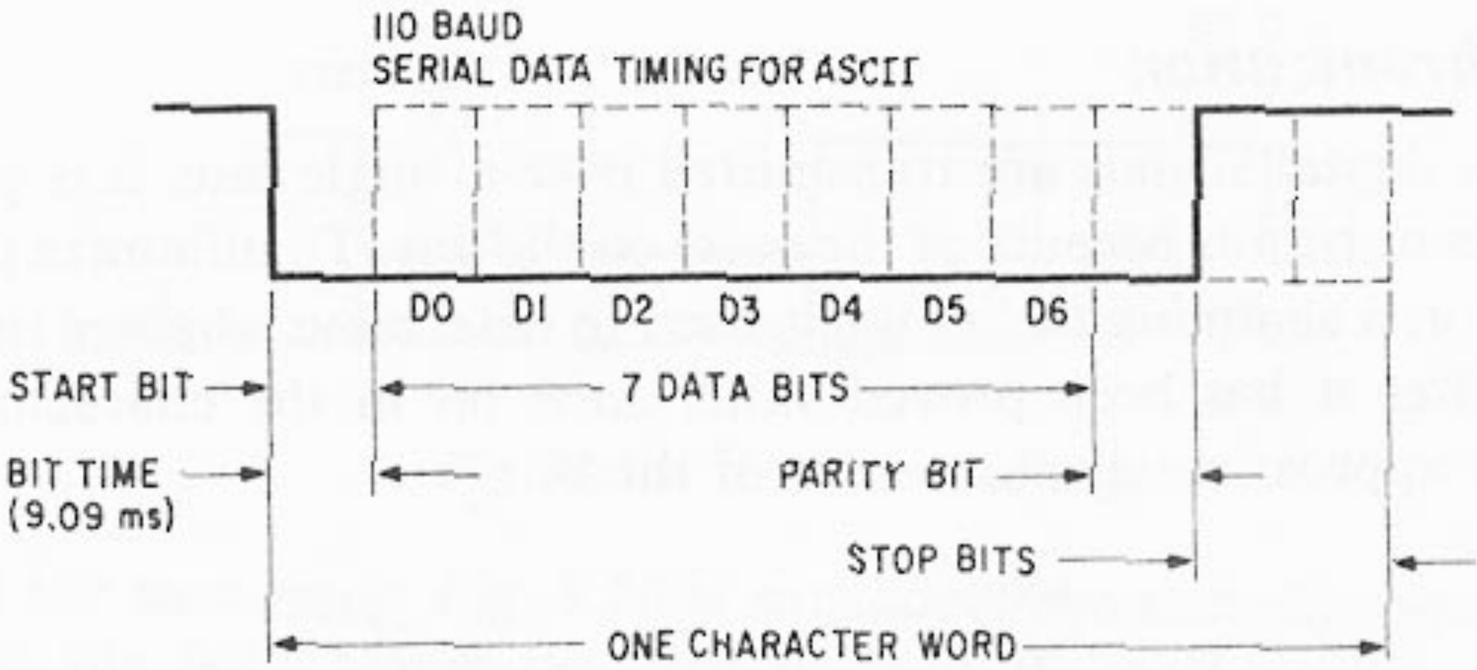


Fig. 9.23 Role of stop, start, and parity bits

UART Character Frame Example

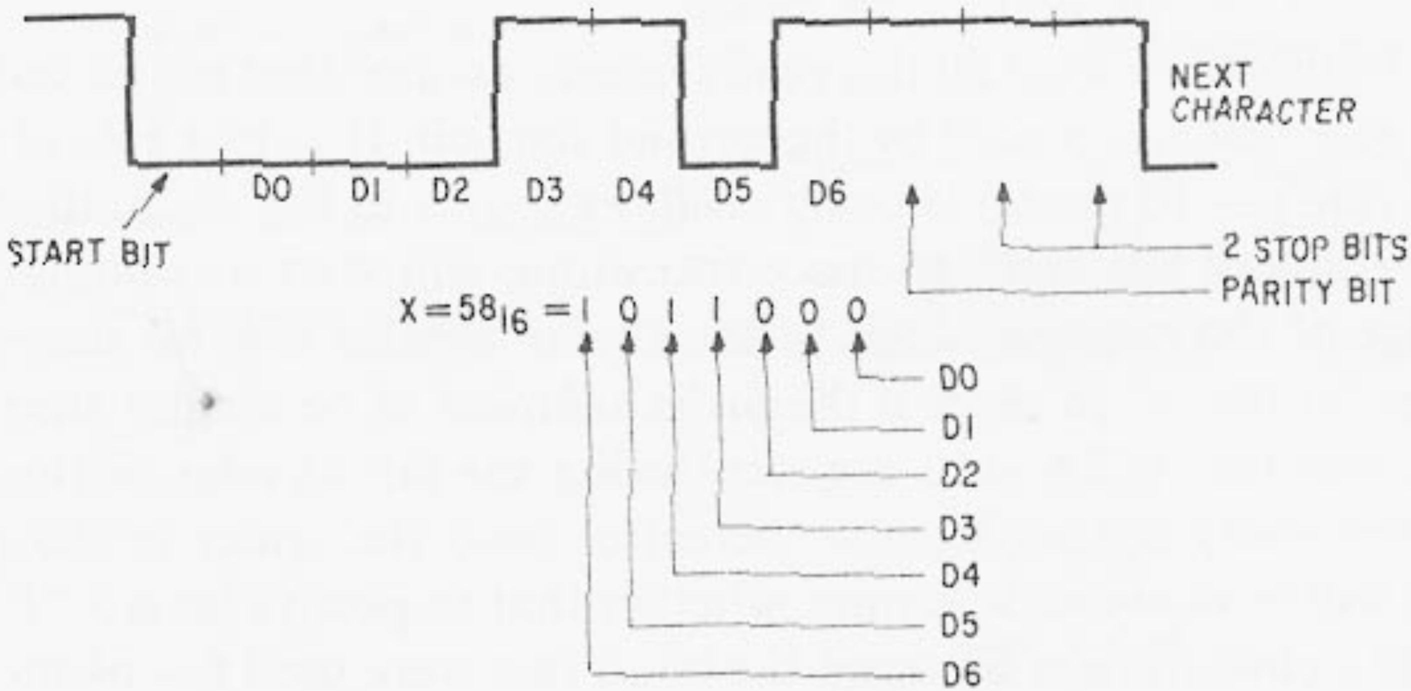


Fig. 9.24 Pulse train

Definitions

- Start Bit
  - Signals the beginning of the data word
  - A low bit after a series of high bits
- Data Bits
  - The meat of the transmission
  - Usually 7 or 8 bits
- Parity Bit
  - An error check bit placed after the data bits
  - Can be high or low depending on whether odd parity or even parity is specified
- Stop Bit/s
  - One or two high bits that signal the end or the data word
- Data Word
  - Start Bit, Data Bits, Parity Bit, & Stop Bit/s

BAUD RATE

BIT RATE

Baud Rate

- Baud Rate = bits transferred/second
- baud rate INCLUDES start, stop, and parity

- "bit rate" refers to JUST data bits transferred per second (may include parity)
- baud rate > bit rate

ATmega2560 USART Port

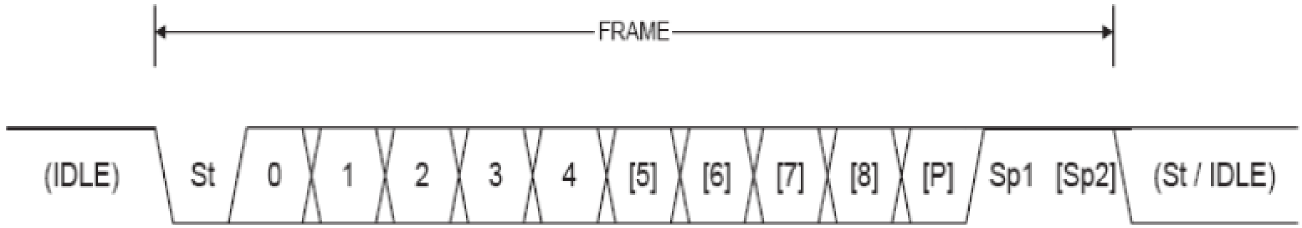
- Send: 8-bit parallel data are converted to serial data, and sent
- Receive: Serial data are converted to parallel data that enter CPU

- 4 USART(Universal Synchronous and Asynchronous Receive and Transmitter) ports
  - USART0, USART1, USART2, USART3
  - Full-Duplex
  - Support both Synchronous mode and Asynchronous mode

- Asynchronous transmission mode
  - Interrupt
    - TX complete
    - TX Data Register Empty
    - RX Complete

ATmega2560 USART Port

- ATmega2560 USART Data Frame Format
  - USART data frame



- Start bit
  - 1bit ‘0’ is automatically generated upon transmission
- Data bit
  - 5, 6, 7, 8, 9 bits
- Parity bit
  - 0 or 1 parity bit (odd parity or even parity)
- Stop bit
  - 1 or 2 stop bits ‘1’ are automatically generated upon transmission

ATmega2560 USART Port API

- Arduino Serial Port API
  - Serial.begin(speed)
    - Set baud rate for USART
      - TX/RX speed
        - » 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200
    - To use USART, you should use this function to initialize the serial port
    - For each USART,
      - For USART0, Serial.begin(speed)
      - For USART1, Serial1.begin(speed)
      - For USART2, Serial2.begin(speed)
      - For USART3, Serial3.begin(speed)

Reference: http://arduino.cc/en/Reference/Serial

ATmega2560 USART Port API

- Serial.end()
  - Disable the USART

- Serial.available()
  - Get the number of bytes (characters) available for reading from the serial port
  - The data has already arrived and stored in the serial receive buffer

- Serial.read()
  - Read incoming 1 byte data from the serial port
  - After reading, the data is erased in the serial receive buffer

- Serial.peek()
  - Read incoming 1 byte data from the serial port
  - After reading, the data is not erased in the serial receive buffer
    - After reading data using peek(), if you use read(), the returned data are the same as before

- Serial.write(val)
  - Write binary data and send them to the serial port
  - Val: a single byte or a string consisting of a series of bytes
  - Return: the number of bytes written and sent

ATmega2560 USART Port API

- Serial.print(val, format)
  - Print data to USART as human-readable ASCII text
    - » val: string or data to send via serial
    - » format: number base (for integer type) or
      number of decimal places (for floating point type)
  - Return: the number of bytes written
  - Serial.print(val, format) Example
    - » Serial.print(78) ; → "78" will be sent
    - » Serial.print('N'); → "N" will be sent
    - » Serial.print("Hello world."); → "Hello world." will be sent
    - » Serial.print(78, BIN); → "1001110" will be sent in ASCII
    - » Serial.print(78, OCT); → "116" will be sent in ASCII
    - » Serial.print(78, DEC); → "78" will be sent in ASCII
    - » Serial.print(78, HEX); → "4E" will be sent in ASCII
    - » Serial.print(1.23456); → "1.23" will be sent
      (rounded to the 2^{nd} decimal place by default)
    - » Serial.print(1.23456, 0) ; → "1" will be sent
      (rounded to the 0^{th} decimal place)
    - » Serial.print(1.23456, 2) ; → "1.23" will be sent
      (rounded to the 2^{nd} decimal place by default)
    - » Serial.print(1.23456, 4) ; → "1.2346" will be sent
      (rounded to the 4^{th} decimal place by default)

ATmega2560 USART Port API

- Serial.println(val, format)
  - Print data to the serial port as human-readable ASCII text followed by a carriage return character ('₩r') and a newline character('₩n')

- Serial.flush()
  - Wait for the transmission of outgoing serial data to complete
  - If Serial.write() is called, data will be transmitting in background
  - Sometimes, you need to wait for TX to be completed
    » Serial.write();
    » Serial.flush(); //Wait for the TX to be completed at here

- void serialEvent()
  - Is triggered upon receiving data from USART
    » On USART0, serialEvent() is called
    » On USART1, serialEvent1() is called
    » On USART2, serialEvent2() is called
    » On USART3, serialEvent3() is called
  - Inside serialEvent() function, Serial.read() should be used to read data

SmartCAR UART Port Configuration

- UART0 port is used for both program port to PC and Bluetooth port to Bluetooth wireless
  - Cannot be used at the same time
- UART0 can be used for debugging on your Arduino program

UART No.	Name	Port / Number	Etc
UART0	RXD0	PE0 / -	Program port Bluetooth port
	TXD0	PE1 / -	
UART1	RXD1	PD2 / 19	Ultrasonic sensor
	TXD1	PD3 / 18	
UART2	RXD2	PH0 / 17	Extension board 1
	TXD2	PH1 / 16	
UART3	RXD3	PJ0 / 15	Extension board 2
	TXD3	PJ1 / 14	

- Baud rate should be set to 115,200 bps

SmartCAR UART Example

- UART_Echo.cpp

```
#include "UART_Echo.h"

unsigned char text[] = "\r\n Welcome! Arduino Mega 2560 \r\n UART0 Test Program.\r\n";

void setup()
{
    int i=0;
    Serial.begin(115200);
    while(text[i] !='\0')
      Serial.write(text[i++]);
    Serial.print("ECHO >>");
}

void loop()
{
    if(Serial.available() > 0)
      Serial.write(Serial.read());
}
```

SmartCAR Example Code Analysis

- Global variable
  - text[]
    - Initial serial communication message

```
unsigned char text[] = "\r\n Welcome! Arduino Mega 2560 \r\n UART0 Test Program.\r\n";
```

- setup()
  - Serial.begin(speed): set the baud rate for serial port
    - 115200 bps
  - while(): use Serial.write(val) for text[] to write 1 byte ASCII value out of text[]
    - Execute until all bytes in text[] are sent

```
    int i=0;
    Serial.begin(115200);
    while(text[i] !='\0')
      Serial.write(text[i++]);
    Serial.print("ECHO >>");
```

SmartCAR Example Code Analysis

- loop()
  - Serial.available(): check the number of bytes
    - Serial.available() > 0
      - if there is any data that are received in the receive buffer,
      - Then, Serial.write(Serial.read())
    - Serial.read(): read 1 byte from the receive buffer
    - Serial.write(): write the received value back to the serial port (echo)

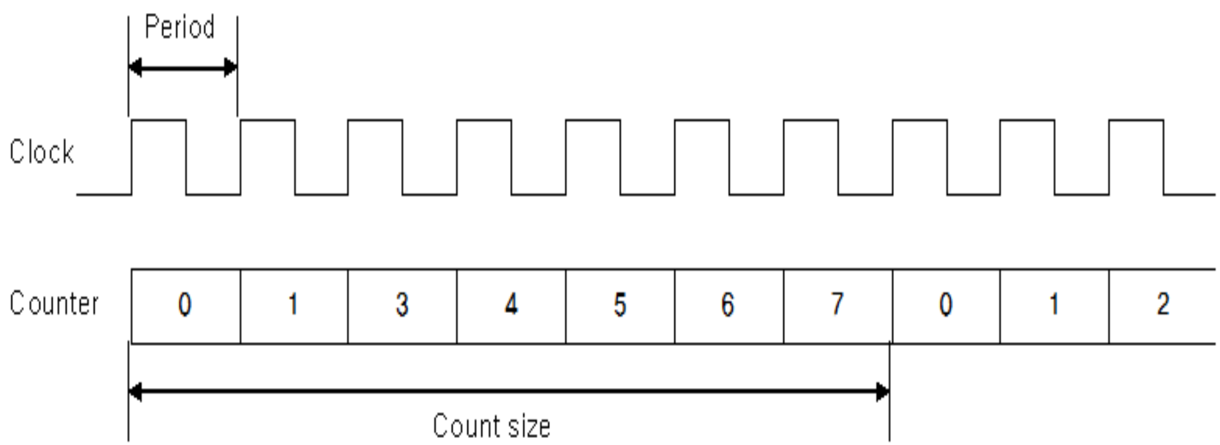
```
if(Serial.available() > 0)
  Serial.write(Serial.read());
```

Today

- Review from the last lecture
  - ATmega 2560 I/O Register
- Midterm: 2pm - 4:30pm, Mon Apr 11
- ATmega2560 microcontroller (MCU)
  - UART
  - **Timer**
- Announcement

Timer: Clock & Counter

- Clock
  - Fundamental “Clock” in computer system
  - Generated by an oscillator crystal
  - Period: one cycle of ‘1’ and ‘0’
  - Counter: counts the number of clock cycles



ATmega2560 Timers

- 0 ~ 5: Total 6 timers

- Timer0, Timer2
  - 8 bit counter (count 0~255)

- Timer1, Timer3, Timer4, Timer5
  - 16 bit counter (count 0~65535)

ATmega2560 Timers

- Timerx function (x =0~5)
  - Timerx library function: use timer and counter x
  - Timer0 & Timer2
    - 8-bit timer: count 0 ~ 255, and then when it comes to 0, it will generate a timer overflow interrupt
  - All other timers, Timer1, Timer3, Timer4, & Timer5
    - 16-bit timer: count 0 ~ 65535, and then when it comes to 0, it will generate a timer overflow interrupt

ATmega2560 Timers

- 3 Functions
  - Timerx::set(unsigned long us, void (*f)())
    - Set an timer period & Interrupt Service Routine function
      - us: interrupt period in micro second
      - void(*f)(): Interrupt Service Route to execute when an timer overflow interrupt occurs
  - Timerx::start()
    - Start the timer
  - Timerx::stop()
    - Stop the timer

ATmega2560 Timers

```
#include <Timer2.h>

#define FRONT_LED   10

int LED_state = 0;

void Timer2_ISR() {
    digitalWrite(FRONT_LED, LED_state);
    if (LED_state)
        LED_state = 0;
    else
        LED_state = 1;
}

void setup() {
    pinMode(FRONT_LED, OUTPUT);
    Timer2::set(1000000, Timer2_ISR);
    Timer2::start();
}

void loop() {
}
```

- What is this program doing?

Delay vs. Timer Approach

```
#define FRONT_LED    10

int LED_state = 0;

void setup() {
    pinMode(FRONT_LED, OUTPUT);
}

void loop() {
    digitalWrite(FRONT_LED, LED_state);
    delay(1000);

    if (LED_state)
        LED_state = 0;
    else
        LED_state = 1;
}
```

```
#include <Timer2.h>

#define FRONT_LED    10

int LED_state = 0;

void Timer2_ISR() {
    digitalWrite(FRONT_LED, LED_state);
    if (LED_state)
        LED_state = 0;
    else
        LED_state = 1;
}

void setup() {
    pinMode(FRONT_LED, OUTPUT);
    Timer2::set(1000000, Timer2_ISR);
    Timer2::start();
}

void loop() {
}
```

- Which one is better in terms of what?

Course Announcement

- For lab session, we will cover
  - SmartCAR Bluetooth Communication with Android device

- Next week, we will have midterm
  - 2pm - 4:30pm, Mon Apr 11
  - Closed book & closed classnote
  - Coverage: lecture & lab over week 1 ~ this week