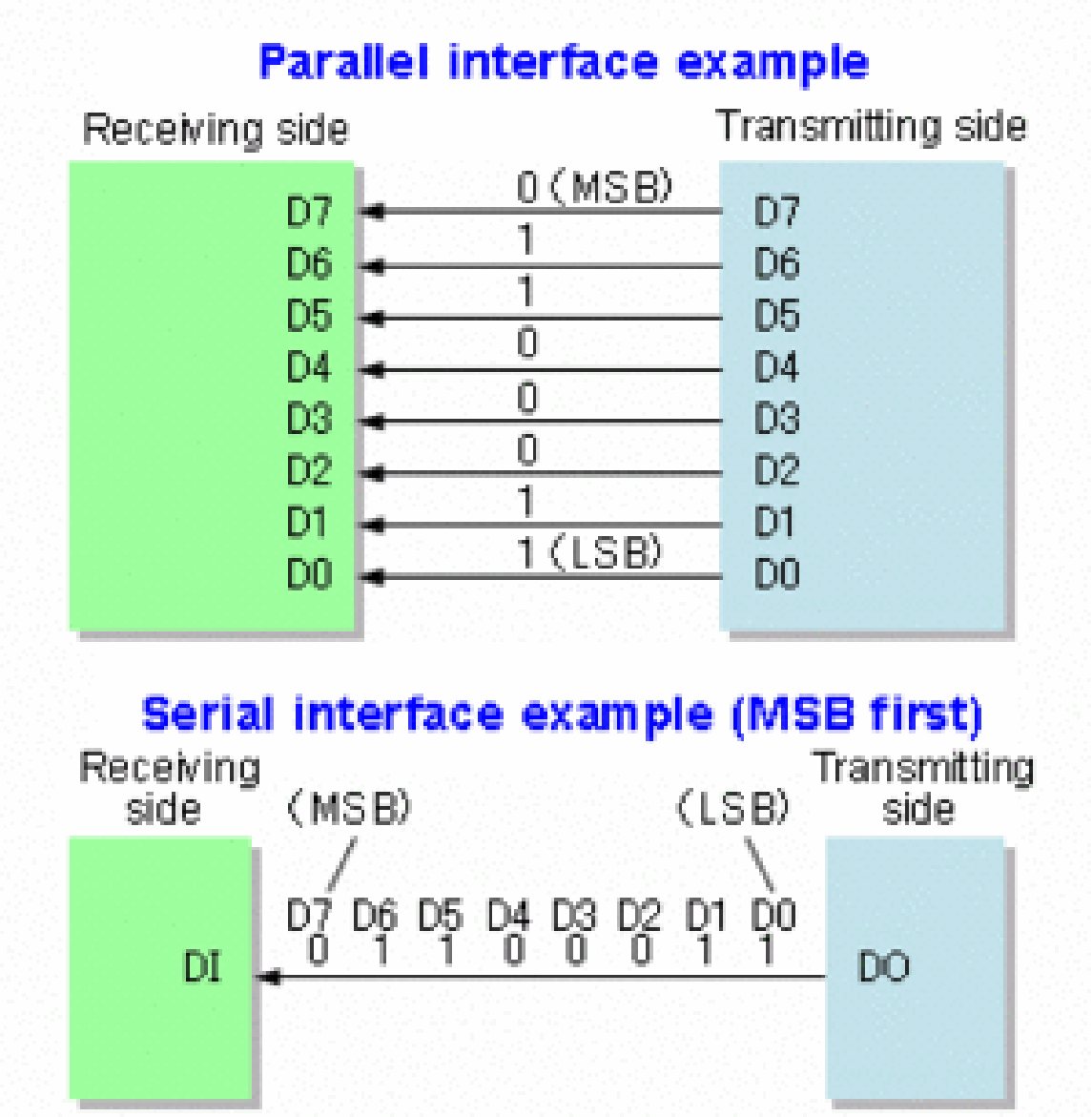
American International University-Bangladesh

Serial Communications Interfaces

Data Transmission

- Data transmission can be performed two ways.

1. **Parallel Communications**, where several bits of data are transmitted/received as a whole, on a link with several parallel channels.
2. **Serial Communications**, where data is transmitted/received bit by bit through a single channel.



Serial Data Communication

❑ Advantage of serial communication:

- ***Smaller number of communication lines*** is required compared to parallel communication.
  - 2 lines (transmit & receive) are required in ***asynchronous full duplex*** serial comm.
  - 3 lines (transmit, receive & clock) are required in ***synchronous*** serial communication.

❑ Disadvantage of serial communication:

- ***More time*** is required to transmit/receive compared to parallel communication.

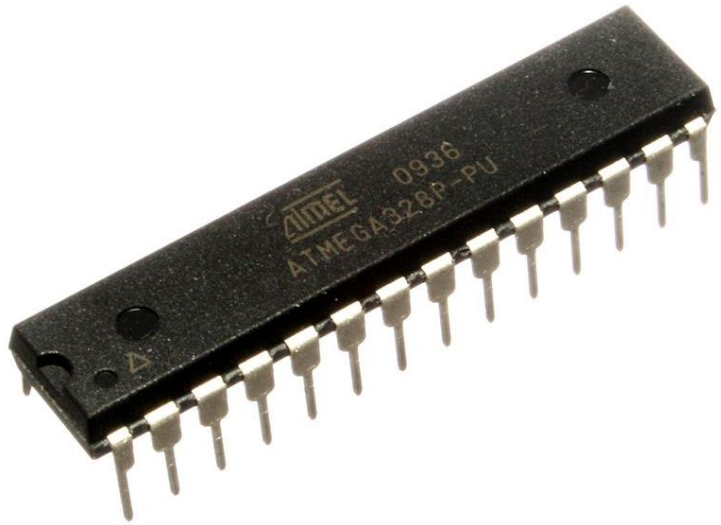
Types of Serial Communication

ATmega328 has 3 types of serial communication interfaces:

1. Universal Synchronous Asynchronous Receiver & Transmitter (USART).

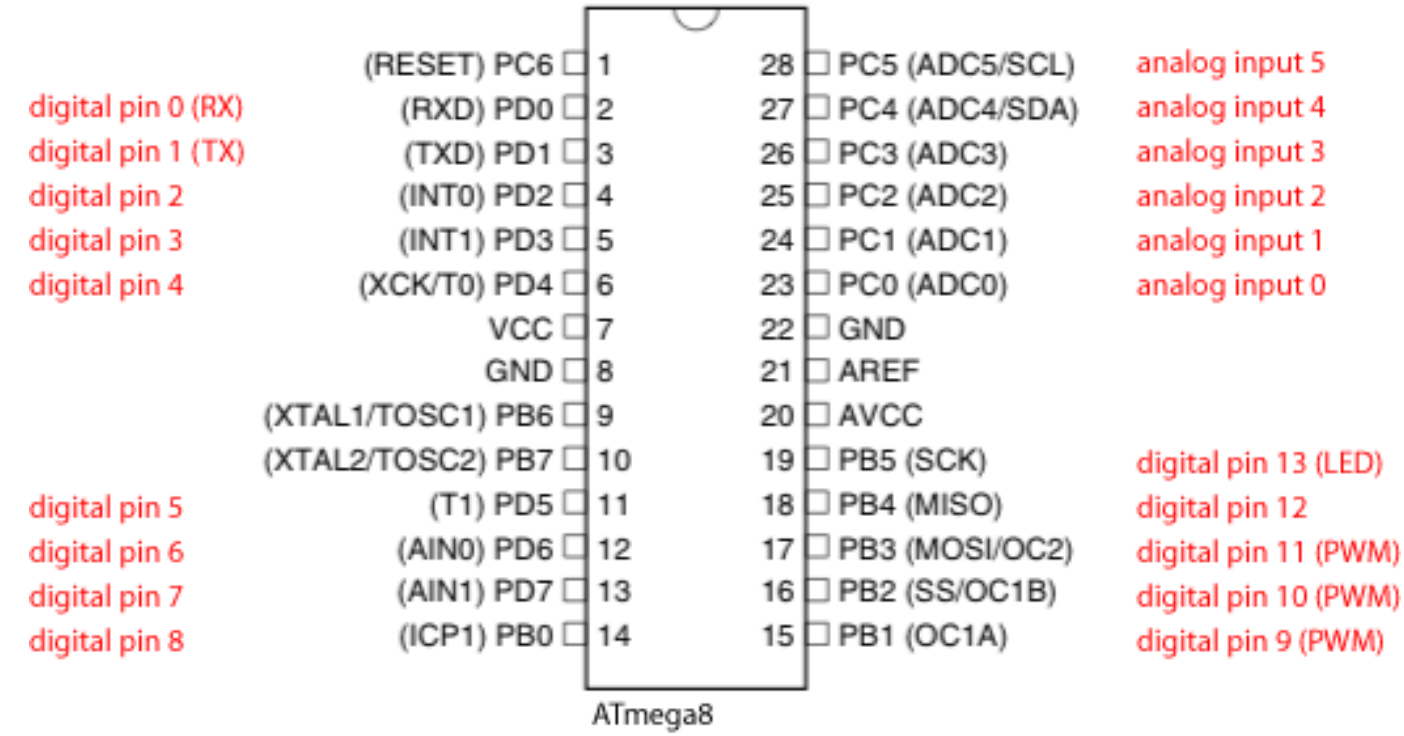
2. Serial Peripheral Interface (SPI).

3. Two Wire Interface (TWI)/ Inter-Integrated Circuit (I2C).



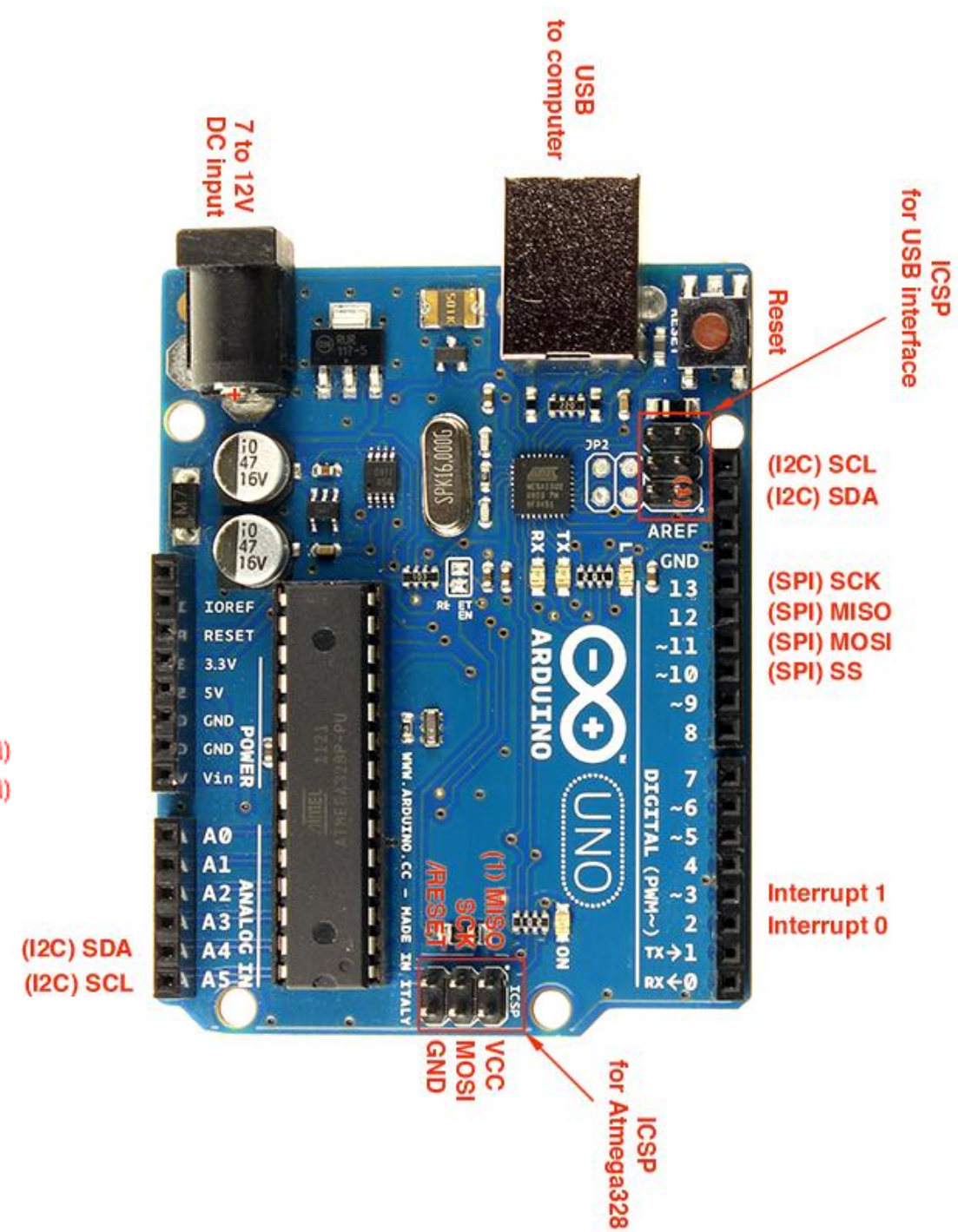
Arduino Pin Mapping

www.arduino.cc



digital pin 0 (RX)
digital pin 1 (TX)
digital pin 2
digital pin 3
digital pin 4
digital pin 5
digital pin 6
digital pin 7
digital pin 8

analog input 5
analog input 4
analog input 3
analog input 2
analog input 1
analog input 0
digital pin 13 (LED)
digital pin 12
digital pin 11 (PWM)
digital pin 10 (PWM)
digital pin 9 (PWM)



USART vs UART

Character	UART	USART
Full Name	Universal Asynchronous Receiver/Transmitter	Universal Synchronous/Asynchronous Receiver/Transmitter
Data type and rate	It generates asynchronous data, hence has **low data rate.**	It generates clocked/synchronous data, hence has **higher data rate**.
Data and Clock	UART requires only **data** signal	In USART, synchronous mode requires **both data and a clock**
Baud rate	Receiver **need to know baud rate** of the transmitter before communication to be established so that UART can generate clock internally and synchronize it with data stream with the help of transition of start bit.	Receiver **need not be required** to know the baud rate of the transmitter. This is derived from the clock signal and data line.
Data Structure	It uses **start bit (before data word), stop bits (one or two, after data word), parity bit (even or odd)** in its base format for data formatting.	USART can also generate data similar to UART. Hence **USART can be used as UART but reverse is not possible.**
Protocol	UART is **simple protocol** to generate data.	USART is **complex and uses many different protocols** to generate the data for transmissions.

Synchronous vs. Asynchronous Communication

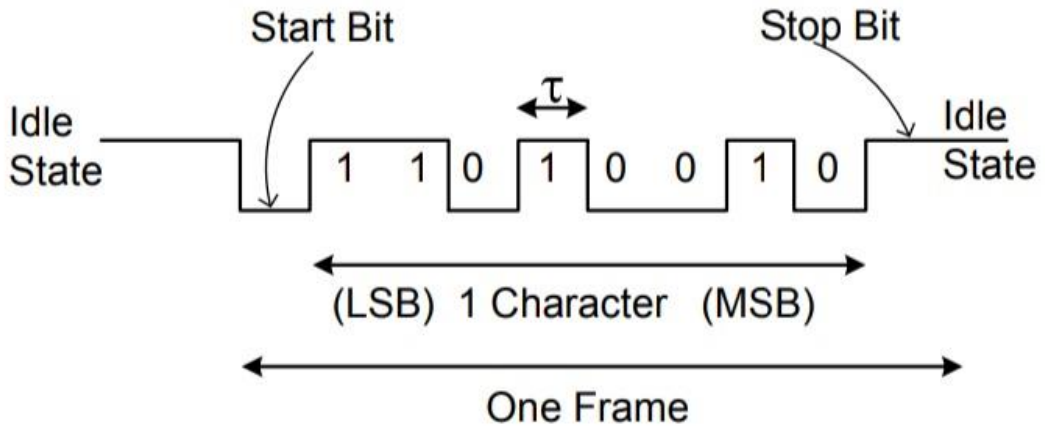
- **Synchronous data transfer: sender and receiver use the same clock signal**
  - supports high data transfer rate
  - needs clock signal between the sender and the receiver
  - requires master/slave configuration
- **Asynchronous data transfer: sender provides a synchronization signal to the receiver before starting the transfer of each message**
  - does not need clock signal between the sender and the receiver
  - slower data transfer rate

Why asynchronous data has low data rate?

- There are many serial data transfer protocols. The protocols for serial data transfer can be grouped into two types: synchronous and asynchronous. For synchronous data transfer, both the sender and receiver access the data according to the same clock. Therefore, a special line for the clock signal is required. A master (or one of the senders) should provide the clock signal to all the receivers in the synchronous data transfer.
- For asynchronous data transfer, there is no common clock signal between the sender and receivers. Therefore, the sender and the receiver ***first need to agree on a data transfer speed***. This speed usually does not change after the data transfer starts. Both the sender and receiver set up their own internal circuits to make sure that the data accessing is follows that agreement. However, just like some watches run faster than others, computer clocks also differ in accuracy. Although the difference is very small, it can accumulate fast and eventually cause errors in data transfer. This problem is solved by adding synchronization bits at the front, middle or end of the data. Since the synchronization is done periodically, the receiver can correct the clock accumulation error. ***The synchronization information may be added to every byte of data or to every frame of data. Sending these extra synchronization bits may account for up to 50% data transfer overhead and hence slows down the actual data transfer rate***.

USART

- Universal Synchronous/Asynchronous Receiver/Transmitter
- It uses 2 pins in Port D:
    1. TXD/PD1 – The serial data transmission line.
    2. RXD/PD0 – The serial data reception line.
- In Arduino, it communicates on **digital pins 0 (Rx) and 1 (Tx) as well as with the computer via USB**. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.
- Data is transmit/receive in a serial frame as follow:

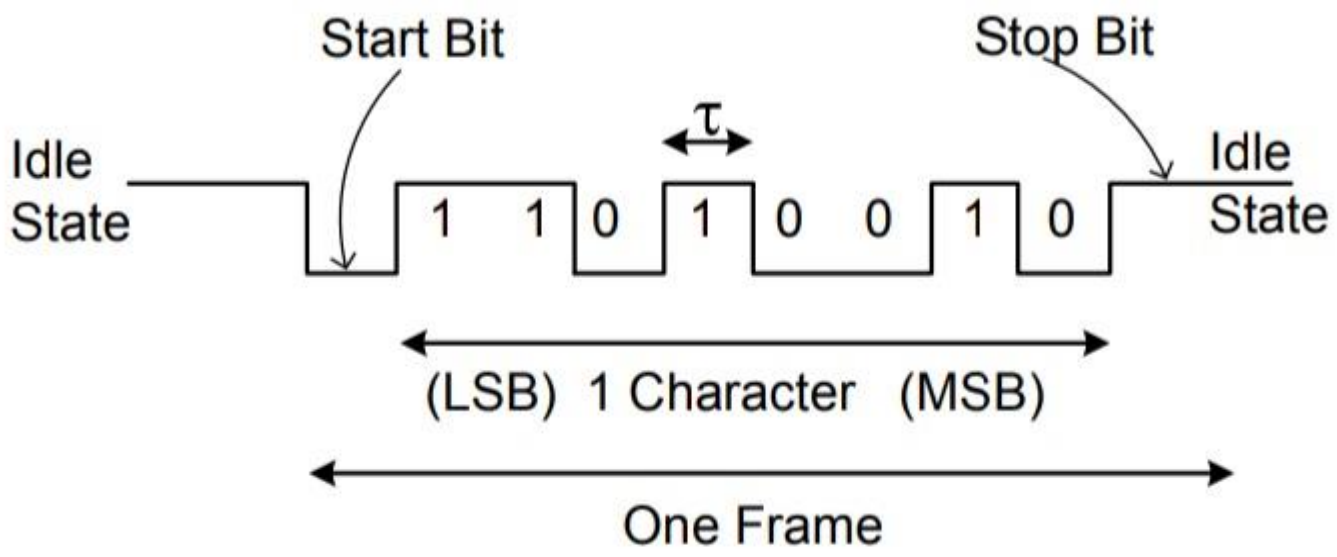


USART

- Each bit is sent with a specific time duration τ, called **bit-time** i.e., required time to transmit each bit. The smaller is τ, the faster is data transmission. The rate of data transmission/reception is called the **Baud rate**.
- **Standard Baud Rate: 2400, 4800, 9600, 14400,19200,. .**
- In the ATmega328, the Baud rate is generated from internal clock. The Baud rates at the transmitter & receptor **must be the same** to avoid communication error.
- **The baud error should be < ± 2% to avoid communication error.**
- **Baud error** = $\frac{\text{Standard baud rate} - \text{Calculated baud rate}}{\text{Standard baud rate}} \times 100\ \%$

Example

- **Draw a serial frame for data (11010010). Calculate baud rate and baud error for bit time 0.104 ms. Also find out whether this will face communication error or not.**
- Soln:



Cont...

- **Here, bit time, τ = 0.104 ms.**
- **Baud rate = $1/\tau$ = 9615 bit per second**
- **Baud Error =** $\frac{9600-9615}{9600} \times 100\%$ = -0.156%
- We know, to avoid communication error, the baud error should be <±2%
- Here, -0.156% < ±2%
- So, this data will not face any communication error for this given bit rate.

USART: Internal Clock Generation – The Baud Rate Generator

- Internal clock generation is used for the asynchronous and the synchronous master modes of operation.
- The **USART Baud Rate Register (UBRRn)** and the down-counter connected to it functions as a baud rate generator.
- The down-counter, running at system oscillator clock frequency (f_{osc}), is loaded with the **UBRRn value**, **each time the counter has counted down to zero**, thus a clock is generated.
- The Transmitter divides the baud rate generator clock output by 2, 8, or 16 depending on mode.
- The baud rate generator output is used directly by the **Receiver's clock** and **data recovery units**.

Calculation of the Baud Rate

Table 17.1 Equations to calculate Baud Rate Register Setting

Operating Mode	Baud Rate Equations	Equations for UBRRn Values
Asynchronous Normal Mode (U2Xn = 0)	$Baud\ rate = \frac{f_{osc}}{16(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{16 \times Baud\ rate} - 1$
Asynchronous Double Speed Mode (U2Xn = 1)	$Baud\ rate = \frac{f_{osc}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{8 \times Baud\ rate} - 1$
Synchronous Master Mode	$Baud\ rate = \frac{f_{osc}}{2(UBRRn + 1)}$	$UBRRn = \frac{f_{osc}}{2 \times Baud\ rate} - 1$

Note:

1. The *Baud rate* is defined as the data transfer rate in bits per second (bps)
2. System oscillator clock frequency (f_{osc}) should be set in Hz
3. UBRRn means contents of the UBRRnH and UBRRnL registers, and their values may vary from 0 to 4095; there are 12 bits data, so the total values can be $2^{12} = 4096$.

Example to Practice

Find the baud rate for the three operating modes when f_{OSC} = 1 MHz and UBRRn = 25. Calculate the baud error and comment whether there will be any communication error or not.

Solution:

For asynchronous normal mode:

$$Baud\ rate = \frac{f_{osc}}{16(UBRRn+1)} = \frac{1\times10^6}{16(25+1)} = 2404 \text{ bps}$$

$$Baud\ error\ rate = \frac{Standard\ baud\ rate - Calculated\ baud\ rate}{Standard\ baud\ rate} \times 100\%$$

$$= \frac{2400 - 2404}{2400} \times 100\% = -0.167\% < \pm 2\%$$

So, there will be no communication error for the given information.

Continuation…

Solution:

For asynchronous double speed mode:

$$Baud\ rate = \frac{f_{osc}}{8(UBRRn+1)} = \frac{1\times10^6}{8(25+1)} = 4808 \text{ bps}$$

$$Baud\ error\ rate = \frac{Standard\ baud\ rate - Calculated\ baud\ rate}{Standard\ baud\ rate} \times 100\%$$

$$= \frac{4800 - 4808}{4800} \times 100\% = -0.167\% < \pm 2\%$$

So, there will be no communication error for the given information.

Continuation...

Solution:

For synchronous master mode:

$$Baud\ rate = \frac{f_{osc}}{2(UBRRn+1)} = \frac{1\times10^6}{2(25+1)} = 19231 \text{ bps}$$

$$Baud\ error\ rate = \frac{Standard\ baud\ rate - Calculated\ baud\ rate}{Standard\ baud\ rate} \times 100\%$$

$$= \frac{19200 - 19231}{19200} \times 100\% = -0.161\% < \pm 2\%$$

So, there will be no communication error for the given information.

Example to practice:

A signal generator has a system oscillator frequency, f_{OSC} = 9MHz. The USART baud rate generator, UBRRn has a content of 50. Utilize the information for calculating the baud rate of the three operating modes. Also, identify the baud error and comment whether there will be any communication error or not.

Table 17-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRRn Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{OSC}}{2(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps)

BAUD	Baud rate (in bits per second, bps)
f_{osc}	System Oscillator clock frequency
UBRRn	Contents of the UBRRnH and UBRRnL Registers, (0-4095)

USART- Arduino Libraries

- USART functions can be used with **Serial Monitor** of Arduino.

1. ***Serial.begin(baud)*** – to enable input/output to serial monitor with baud speed. Must be written in *setup()*

2. ***Serial.available()*** – Get the number of bytes (characters) available for reading from the serial port.

3. ***Serial.println(val)*** – to display *val* value to serial monitor with *newline* added.

4. ***Serial.print(val)*** – as above but without *newline*.

5. ***Serial.print("Error")*** – display message "*Error*" without newline.

6. ***Serial.read()*** – Reads incoming serial data.

- others functions – refer to arduino.cc.

Advantages and Disadvantages of USART

❑ Advantages

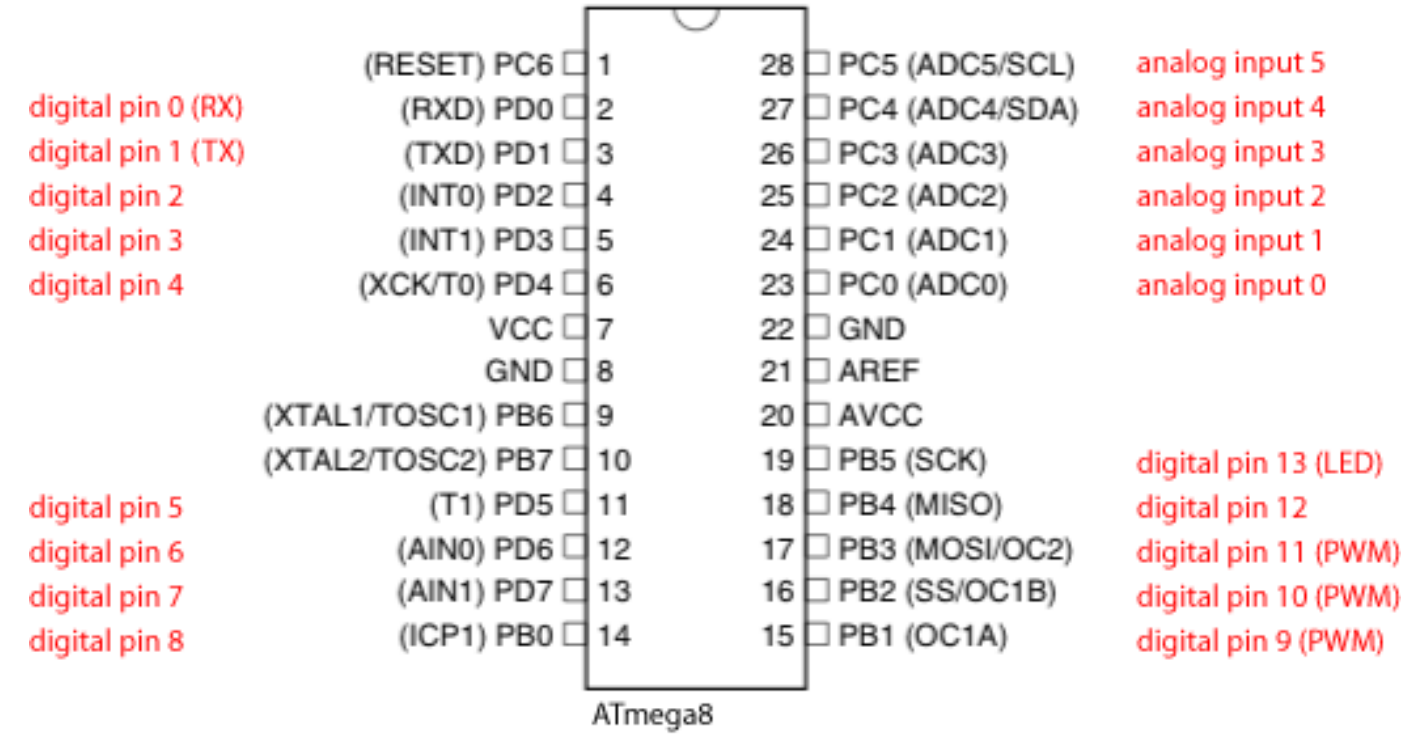
- Hardware complexity is low.
- As this is one to one connection between two devices, software addressing is not required.
- Due to its simplicity, it is widely used in the **devices having 9 pin connectors**.

❑ Disadvantages

- It is suitable for **communication between only two devices**.
- It supports fixed data rate between devices wanting to communicate otherwise data will be **garbled** (distorted ,unclear).

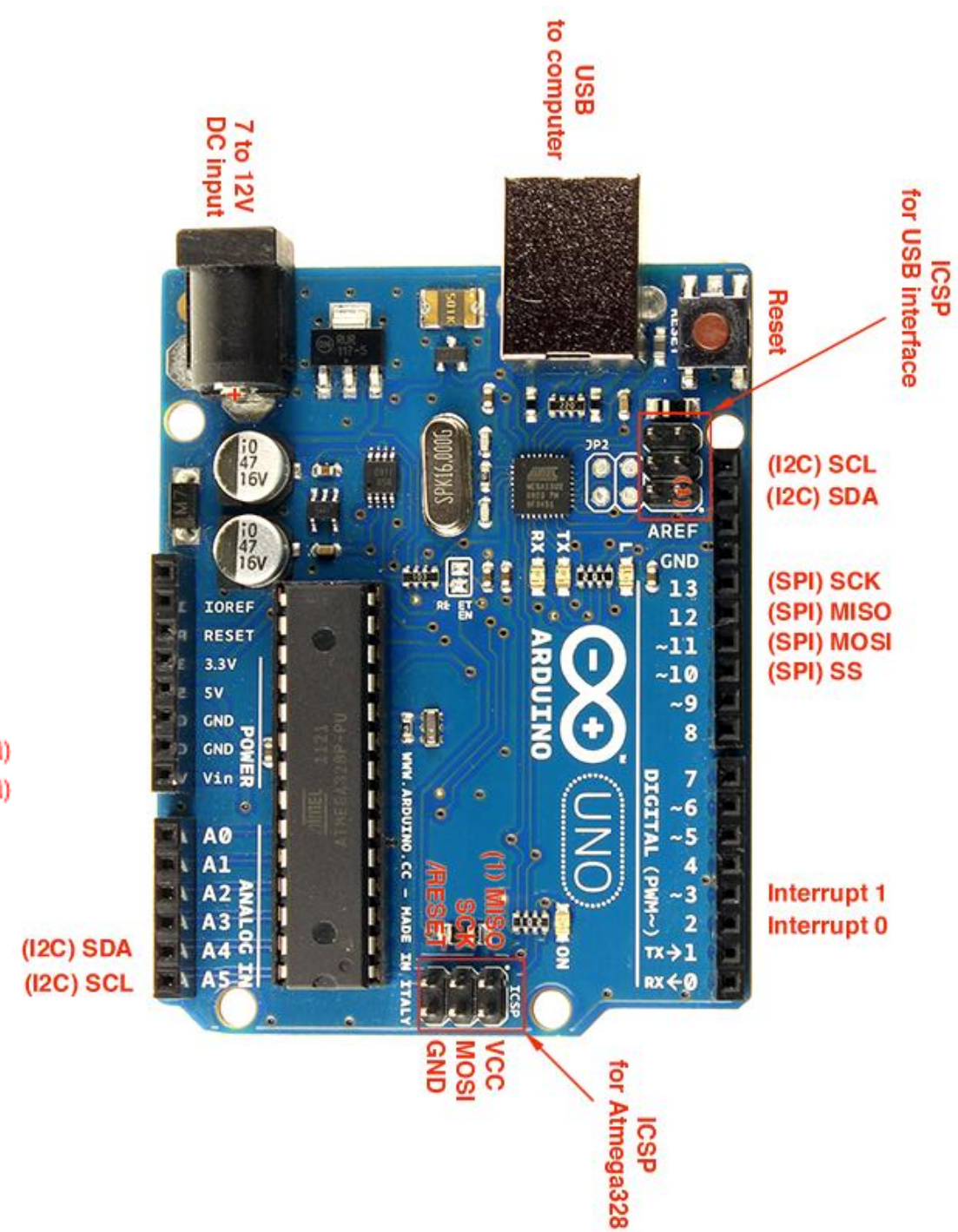
Arduino Pin Mapping

www.arduino.cc

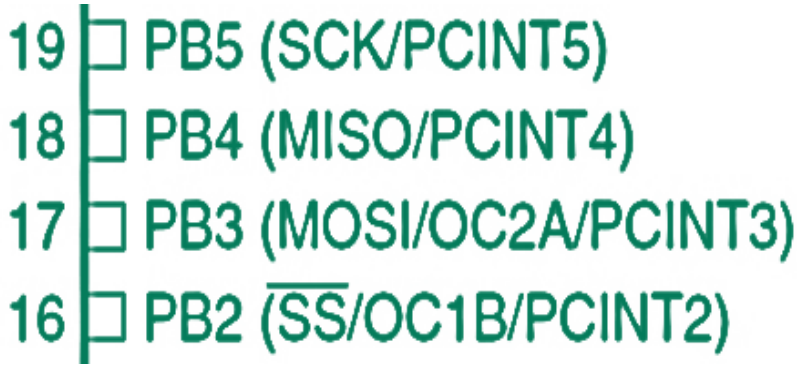


digital pin 0 (RX)
digital pin 1 (TX)
digital pin 2
digital pin 3
digital pin 4
digital pin 5
digital pin 6
digital pin 7
digital pin 8

analog input 5
analog input 4
analog input 3
analog input 2
analog input 1
analog input 0
digital pin 13 (LED)
digital pin 12
digital pin 11 (PWM)
digital pin 10 (PWM)
digital pin 9 (PWM)



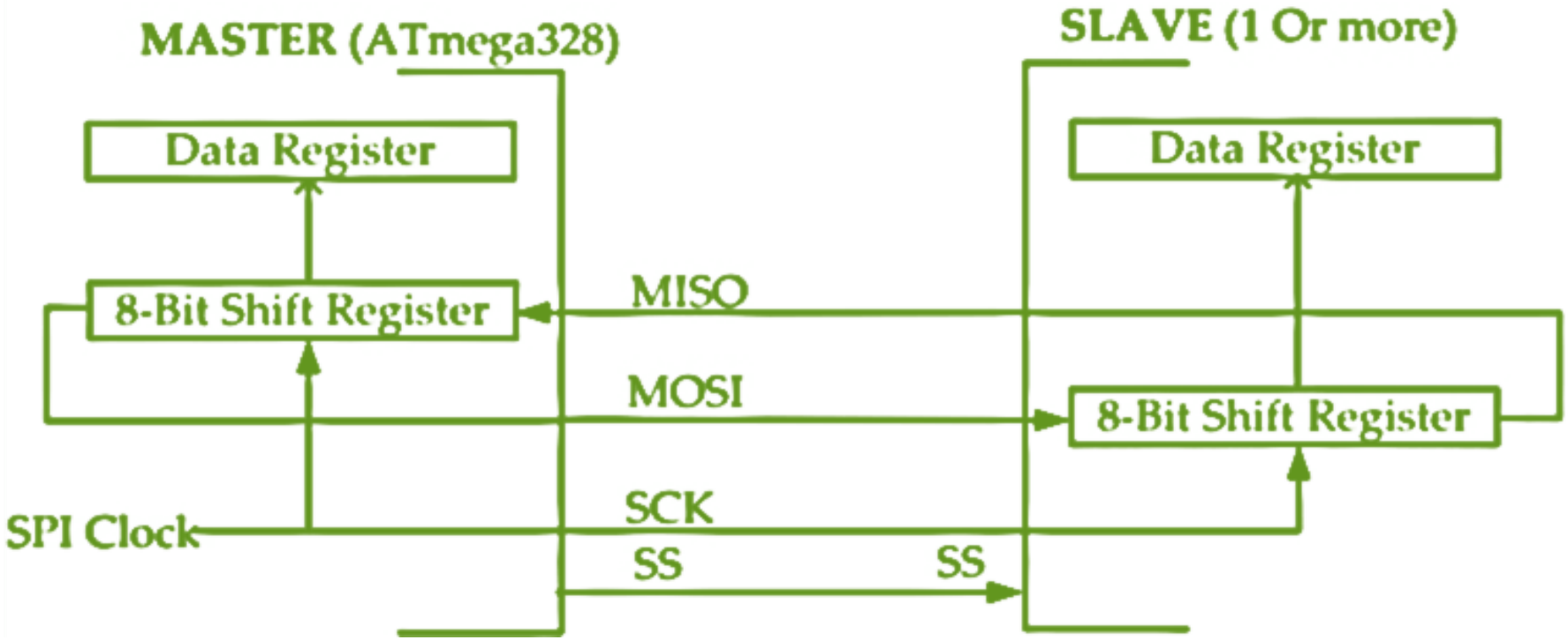
Serial Peripheral Interfaces (SPI)



- SPI is a synchronous data communication.
- SPI uses 4 pins in Port B:
- **SS/PB2** – Slave Selection pin, this pin on each peripheral enables the Master to enable and disable a slave or peripheral device.
- **MOSI/PB3** – Master Out Slave In, the Master line for sending data to the peripherals (Slaves), this pin enables to Master drive a slave.
- **MISO/PB4** – Master In Slave Out, the Slave line for sending data to the master, this pin enables the Master to receive any slave data.
- **SCK/PB5** – The clock pulses which synchronize data transmission generated by the Master.

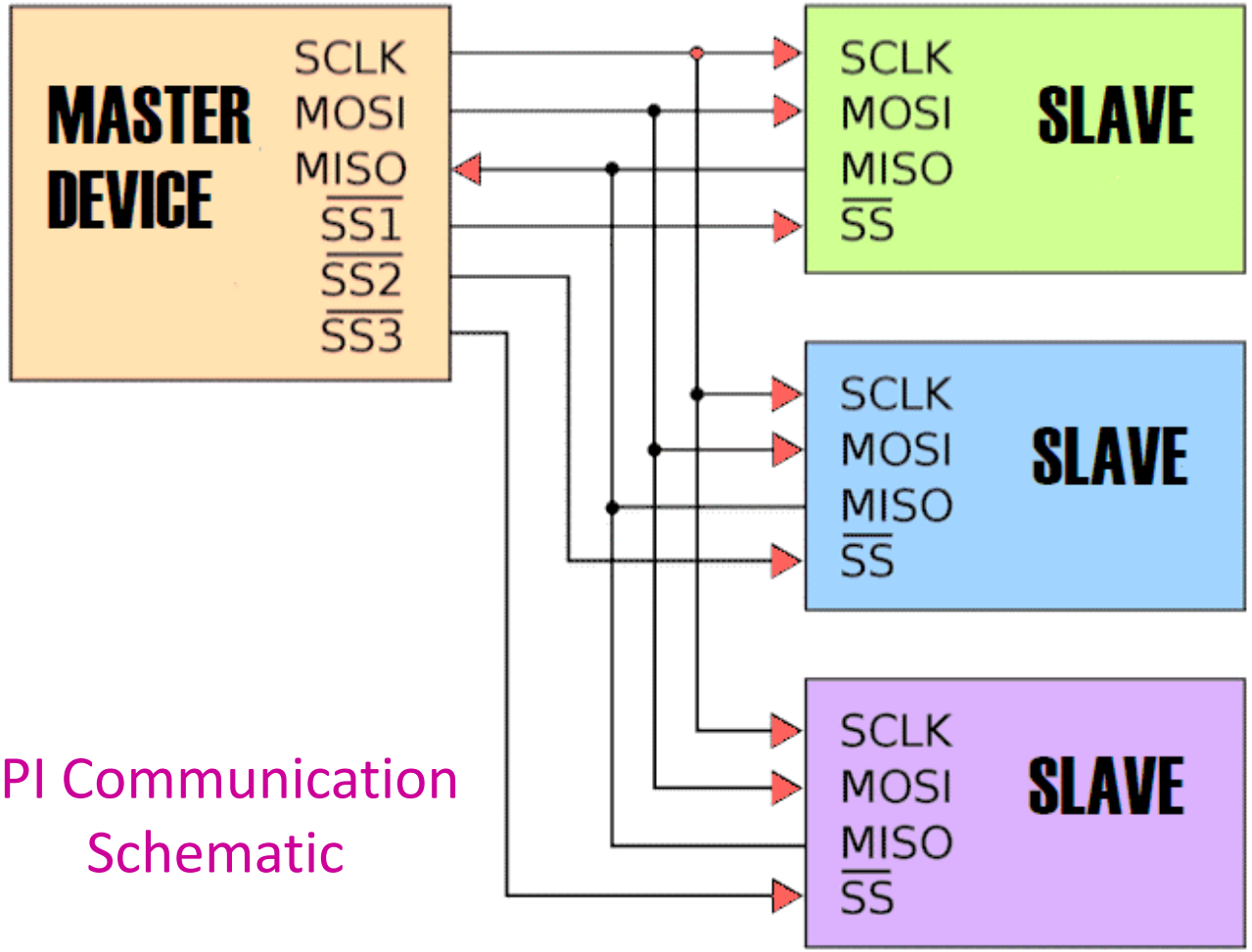
Serial Peripheral Interfaces (SPI)

- Connection using SPI is in the Master-Slave configuration.
- **Master** – Normally, is the ATmega328. Master **initiates** the data transfer. SPI clock is also generated by master.
- **Slave** – Consists of 1 or more SPI I/O peripherals. The slave transfers data as a **reaction** to master.



Serial Peripheral Interfaces (SPI)

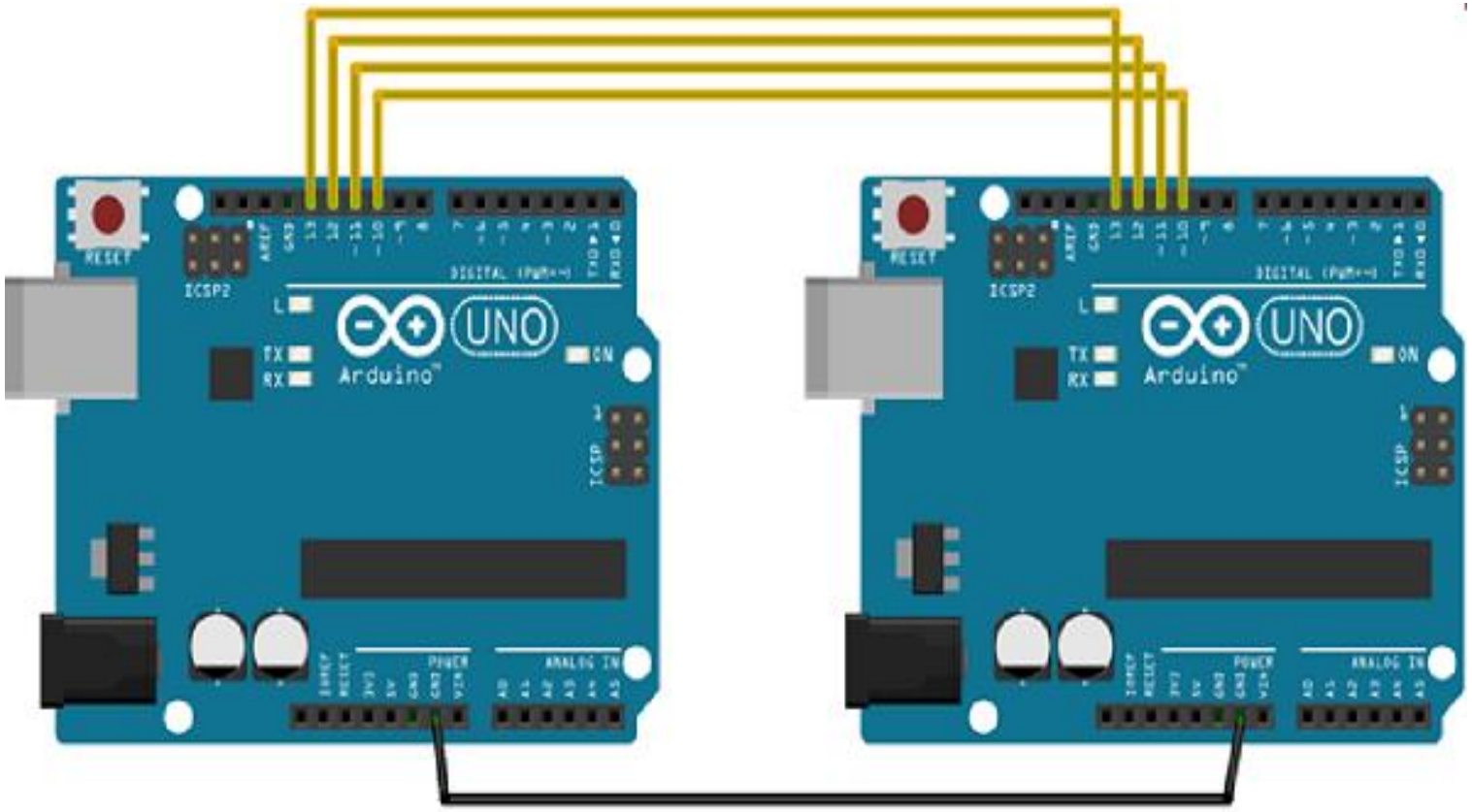
- Connection using SPI is in the Master-Slave configuration.



SPI Communication Schematic

Serial Peripheral Interfaces (SPI)

- Let's make an example with Arduino. In this example, we are going to let the two Arduinos to communicate with each other.



Pin connections of these two Arduinos

We will connect two Arduino UNO boards together; one as a master and the other as a slave.

- (SS): pin 10; Slave Selection
- (MOSI): pin 11
- (MISO): pin 12
- (SCK): pin 13

Communication

SPI Arduino Libraries

- **SPI.begin()** – Initializes the SPI bus by setting SCK, MOSI, and SS to outputs, set SCK & MOSI low, & SS high. Must be written in **setup()**
- **SPI.end()** – Disables the SPI bus.
- **SPI.setBitOrder(order)** – Sets the order of the bits shifted out of and into the SPI bus, either LSBFIRST or MSBFIRST.
- **SPI.setClockDivider(divider)** – Sets the SPI clock divider (SPI_CLOCK_DIVn, n = 2, 4, 8, 16, 32, 64, or 128). The **default setting** is **SPI_CLOCK_DIV4**, which sets the SPI clock to **4 MHz** for Uno
- **SPI.setDataMode(mode)** – Sets the SPI data mode: clock polarity and phase. **Available modes: SPI_MODE0 – SPI_MODE3**. refer to arduino.cc
- **SPI.transfer(val)** – Transfers **one byte** over the SPI bus, both sending and receiving. **val: the byte to send out**.
- **Returns:** the byte read from the bus.
- **SPI.beginTransaction(SPISettings(speedMaximum, dataOrder, dataMode))** – speedMaximum is the clock, dataOrder(MSBFIRST or LSBFIRST), dataMode(SPI_MODE0, SPI_MODE1, SPI_MODE2, or SPI_MODE3).

Advantages and Disadvantages of SPI

❑ Advantages

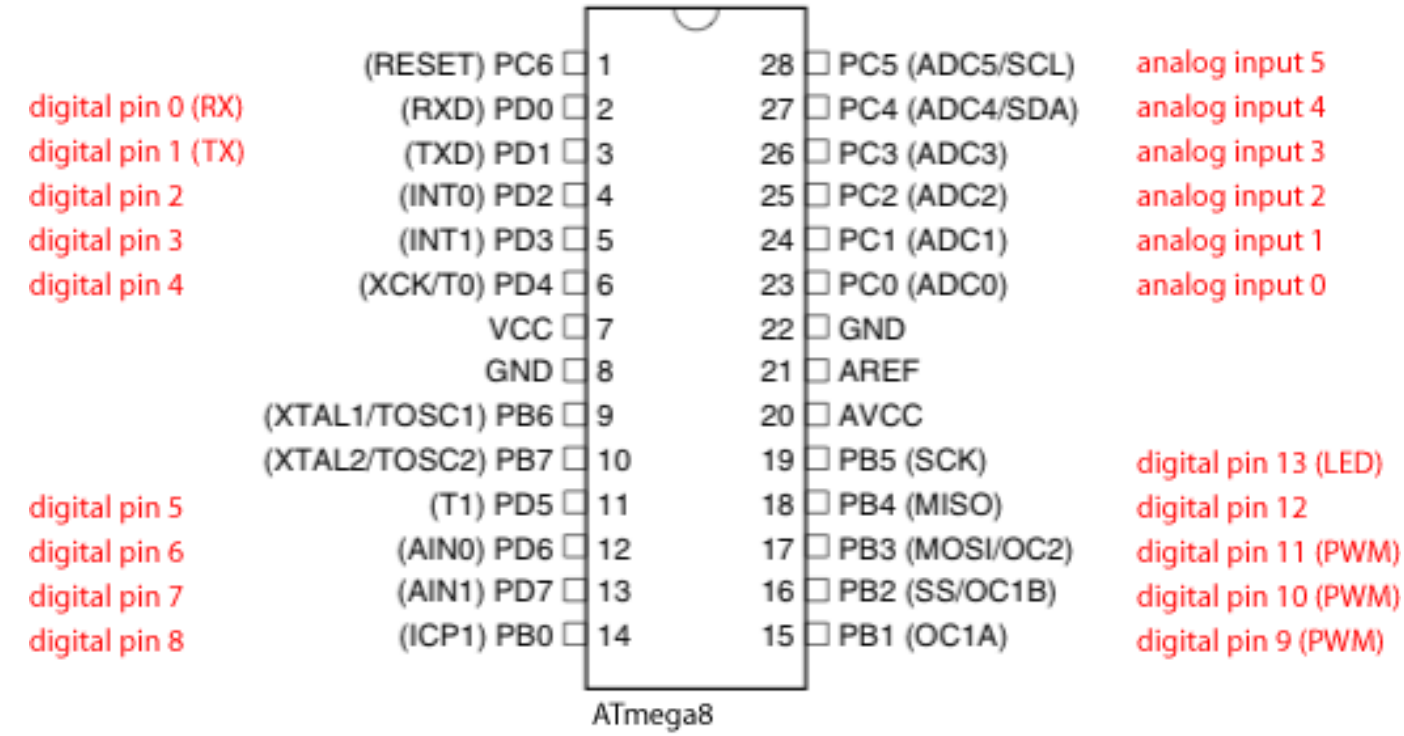
- It is a simple protocol and hence does not require processing overheads.
- Supports **full duplex** communication.
- Due to separate use of CS lines, same kind of multiple chips can be used in the circuit design.
- SPI uses push-pull and hence higher data rates and longer ranges are possible.
- SPI uses less power compared to I2C

❑ Disadvantages

- As number of slave increases, number of CS lines increases, this results in hardware complexity as number of pins required will increase.
- To add a device in SPI requires one to add extra CS line and changes in software for particular device addressing is concerned.
- Master and slave relationship can not be changed as usually done in I2C interface.
- No flow control available in SPI.

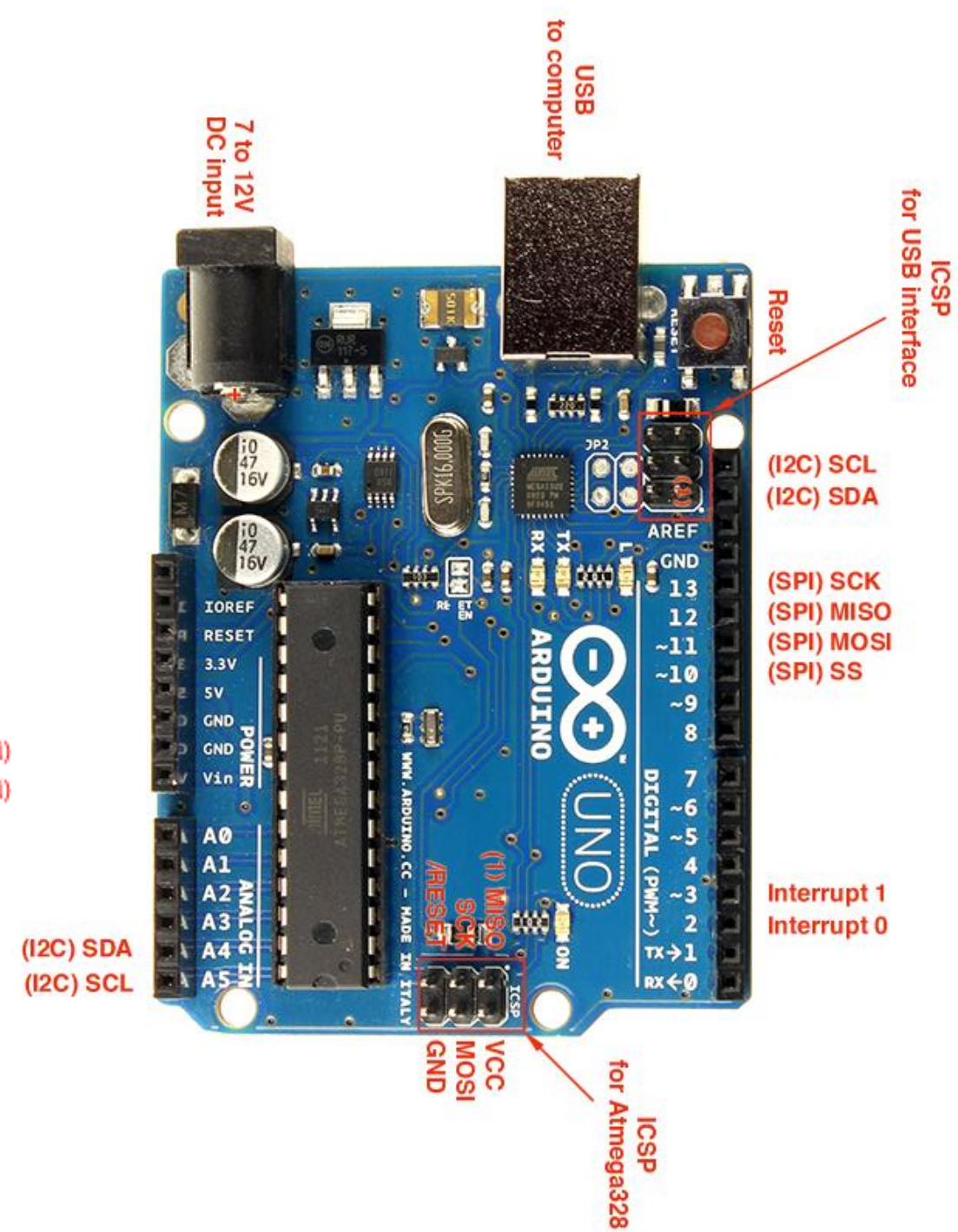
Arduino Pin Mapping

www.arduino.cc



digital pin 0 (RX)
digital pin 1 (TX)
digital pin 2
digital pin 3
digital pin 4
digital pin 5
digital pin 6
digital pin 7
digital pin 8

analog input 5
analog input 4
analog input 3
analog input 2
analog input 1
analog input 0
digital pin 13 (LED)
digital pin 12
digital pin 11 (PWM)
digital pin 10 (PWM)
digital pin 9 (PWM)

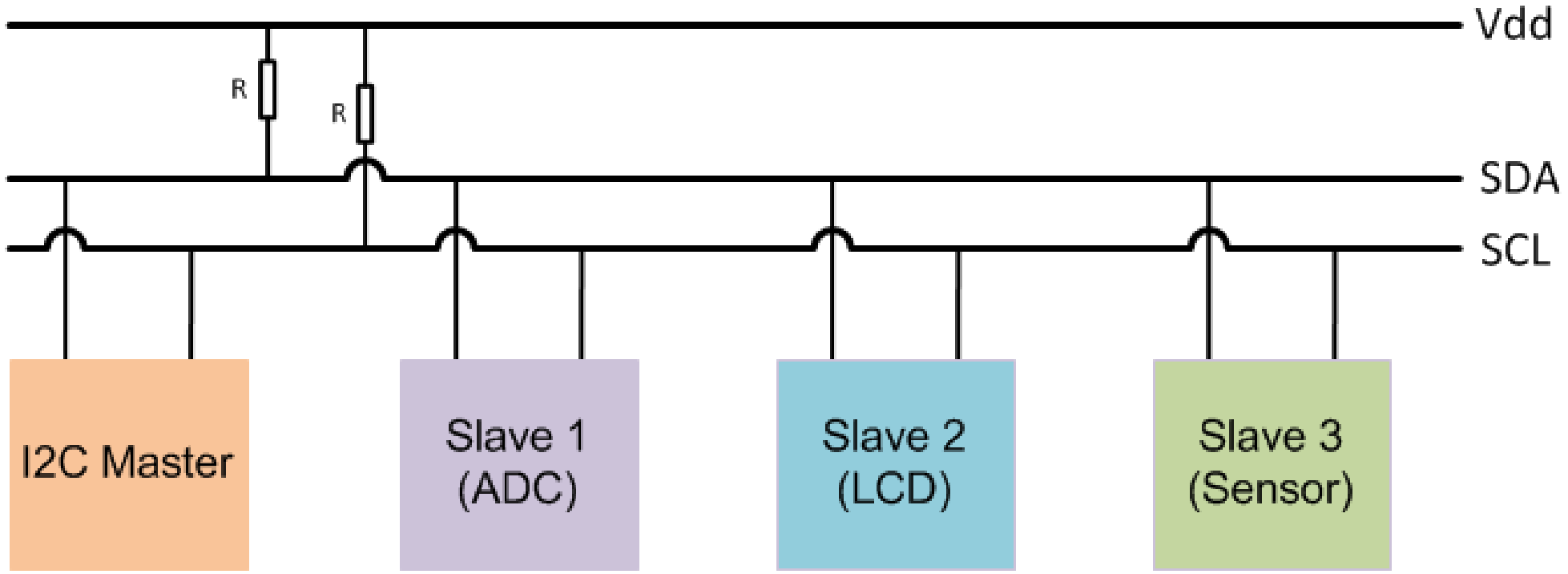


I2C (Inter-Integrated Circuit): What is it?

- An inter-integrated circuit (I2C) or two-wire interface (TWI) is a synchronous serial protocol originally developed by Philips Semiconductors (now NXP).
- It's a multi-master, multi-slave serial bus for low-speed devices that only requires two wires among multiple devices. It can easily be implemented with two digital input/output channels on a device.
- An I2C bus has just two wires over which hundreds of devices communicate serially.
- As a master-slave type communication standard, **at least one device** connected to the bus should be the **master** that generates a clock signal for synchronous serial data communication.
- The slave devices can transfer data to and from the master device(s), which access slave devices by their I2C addresses. The address of each slave device on an I2C bus must be unique. The I2C slave devices still must obtain their addresses from NXP.

I2C (Inter-Integrated Circuit)

- A chip-to-chip protocol for communicating with low-speed peripherals
- The I2C bus drivers are **open drain**, which means the devices can pull the I2C signal line low but cannot drive it high. **By default, both the lines are pulled high** by pull-up resistors until the bus is accessed by a master device to avoid **bus contention**.



I2C (Inter-Integrated Circuit)

- I2C is another serial protocol for **two-wire interface** to connect to **low-speed devices** like Micro-controller, EEPROMs, I/O Interfaces, and other similar devices used in embedded systems.
- **I2C is a bus** for communication between **a master (or can be multiple masters) and a single or multiple slave devices.**
- I2C uses only **two wires- SCL (Serial Clock) and SDA (Serial Data)**.
- **SCL (Serial Clock):** The clock line used to **synchronize all data transfers** over the I2C bus, the line over which master device(s) generate the clock signal.
- **SDA (Serial Data):** The data line used to **transmit the data** between devices, the line over which the master and slave devices communicate serial data
- Each I2C Slave devices have a 7-bit/10-bit addressing.
- The data transfer rate depends on the clock frequency**.** In the **standard mode**, the clock frequency is 100-400 kHz with **7 bit addressing** and data transfer of 100 kbps.

Working of I2C in Arduino

- The I2C is a **half-duplex type of communication**. A master device can only read or write data to the slave at a time. All operations are controlled by master device(s).
- In I2C data transfer occurs in Message Frames which are then divided into Frames of Data. A message contains the various number of Frames in which one frame contain the address of the slave, and remaining frames for data to be transmitted.
- The message includes START/STOP Conditions, READ/WRITE Bits and ACK/NACK (Acknowledgement/No-acknowledgement) Bits between each Data Frame. Working shown below:
- **Start Condition:** The SDA line switches from **high to low** voltage level before SCL switches from **high to low**.
- **Stop Condition:** The SDA line switches from **low to high** voltage level after SCL switches from **low to high**.

Advantages and Disadvantages of I2C

❑ Advantages

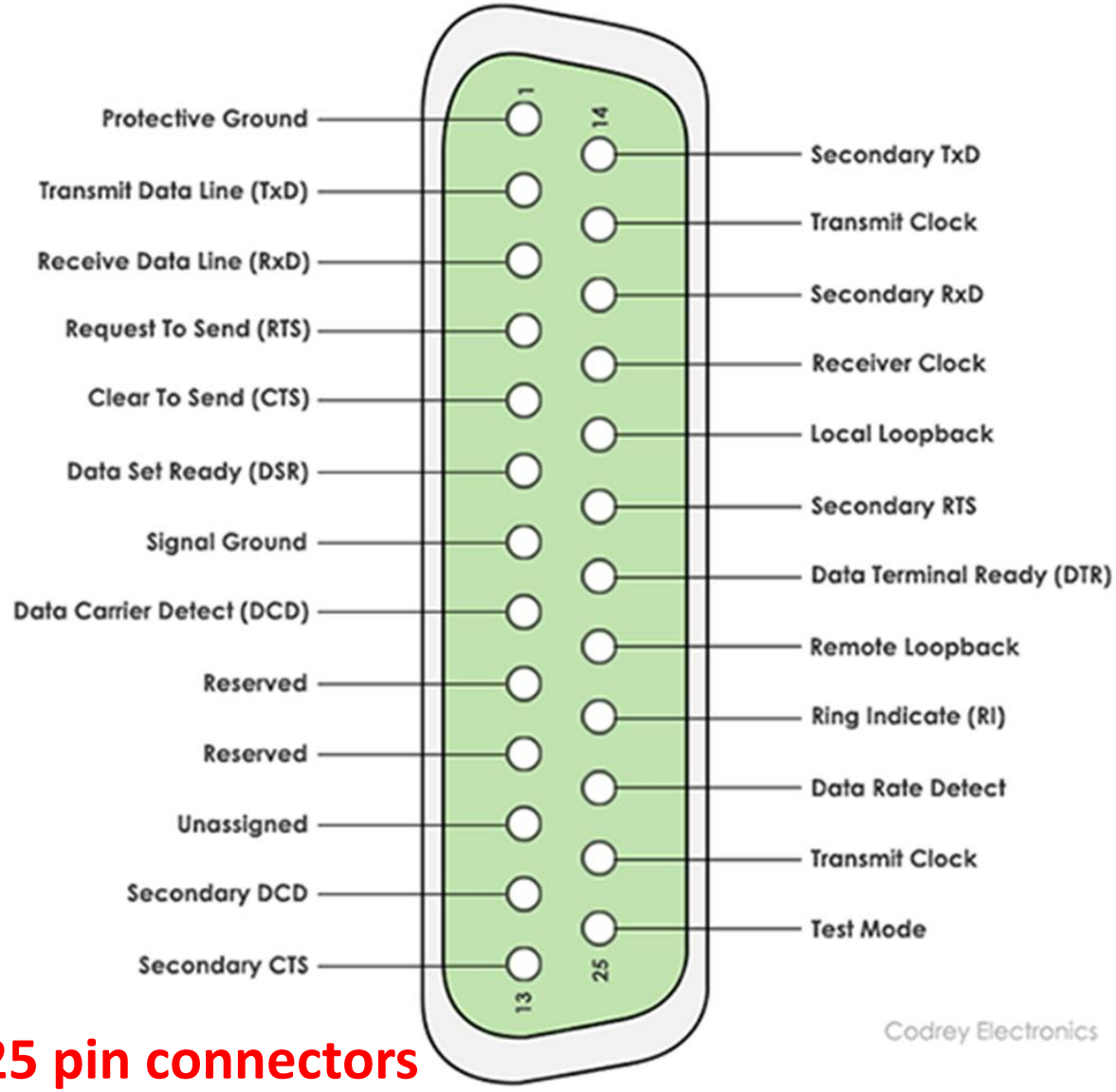
- Due to open collector design, limited slew rates can be achieved.
- More than one masters can be used in the electronic circuit design.
- Needs fewer i.e., only 2 wires for communication.
- I2C addressing is simple which does not require any CS lines used in SPI and it is easy to add extra devices on the bus.
- It uses open collector bus concept. Hence there is bus voltage flexibility on the interface bus.
- Uses flow control.

• Disadvantages

- Increases complexity of the circuit when number of slaves and masters increases.
- I2C interface is **half duplex**.
- Requires software stack to control the protocol and hence it needs some processing overheads on microcontroller/ microprocessor

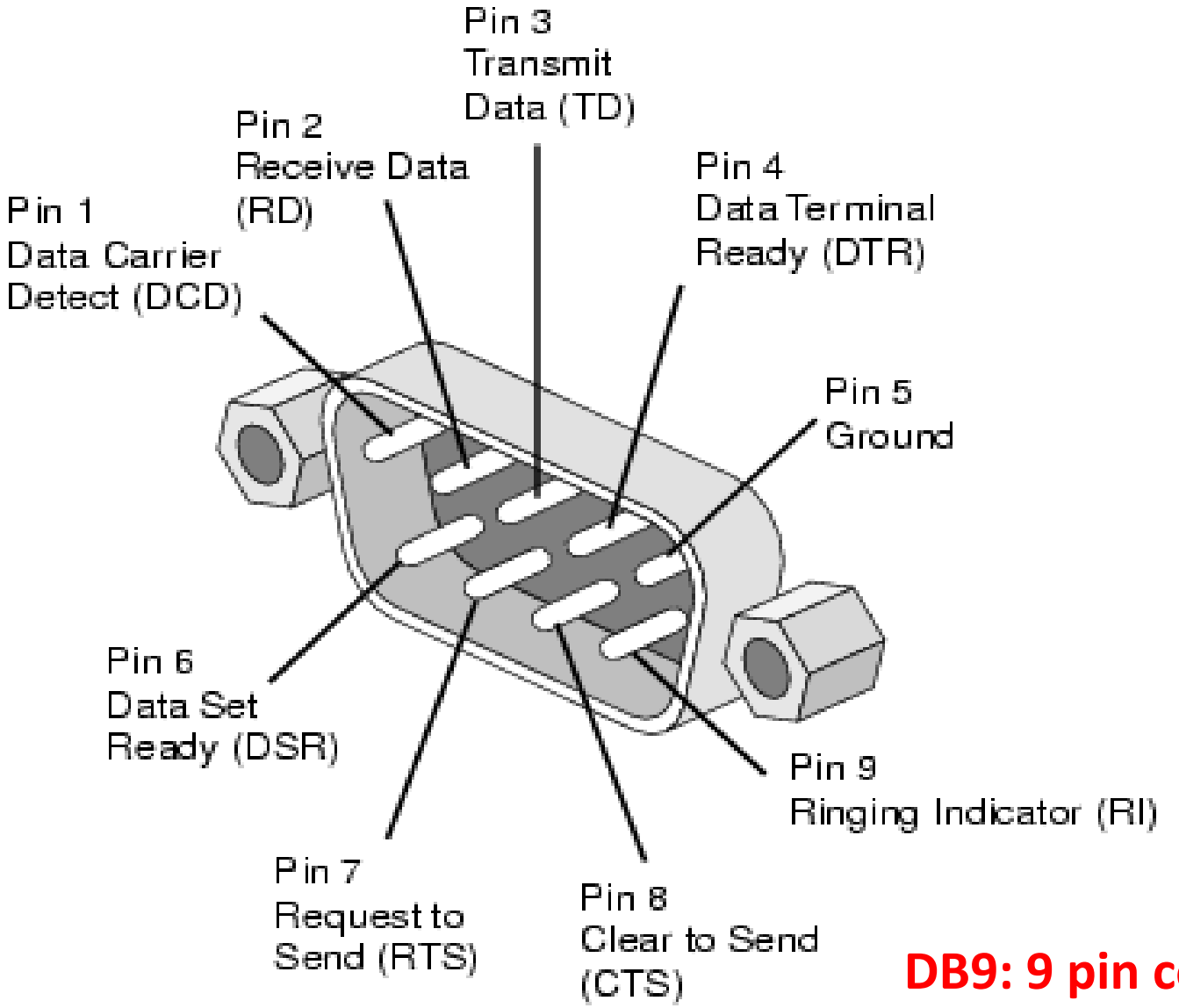
RS232

- **RS232** is the **interface** mainly used for **serial data communication**.
- It supports **data transfer rate from about 110 bps to about 115200 bps**.
- Hyper terminal is the application mainly used to check serial communication port of the computer, often referred as **COM port**.
- The interface is of **two types-**
  - **DB9 pin connector and**
  - **DB25 pin connector**.
- The interface is **mainly used for one to one serial communication**.



DB25: 25 pin connectors

RS232



DB9: 9 pin connectors

Summary

- The USART is good for basic, **full-duplex** data communication between two devices with a similar clock.
- An SPI is good for **full-duplex**, high-speed data communication with two or more peripherals.
- An I2C is **half-duplex** type of communication good for slow-speed data communication with multiple devices, among multiple masters over a 2-wire bus.

Question

Q. What is Baud rate? Draw a serial frame for data (10010110). Calculate baud rate and baud error for bit time 0.25 ms. Also find out whether this will face communication error or not?

Q. A signal generator has a system oscillator frequency, f_{OSC} = 9MHz. The USART baud rate generator, UBRRn has a content of 50. Utilize the information for calculating the baud rate of the three operating modes. Also, identify the baud error and comment whether there will be any communication error or not.

Q. Explain the working principle of SPI interface using master slave configuration.

Q. Write differences between SPI and I2C?

Q. Compare between USART and UART?

Thanks for attending….