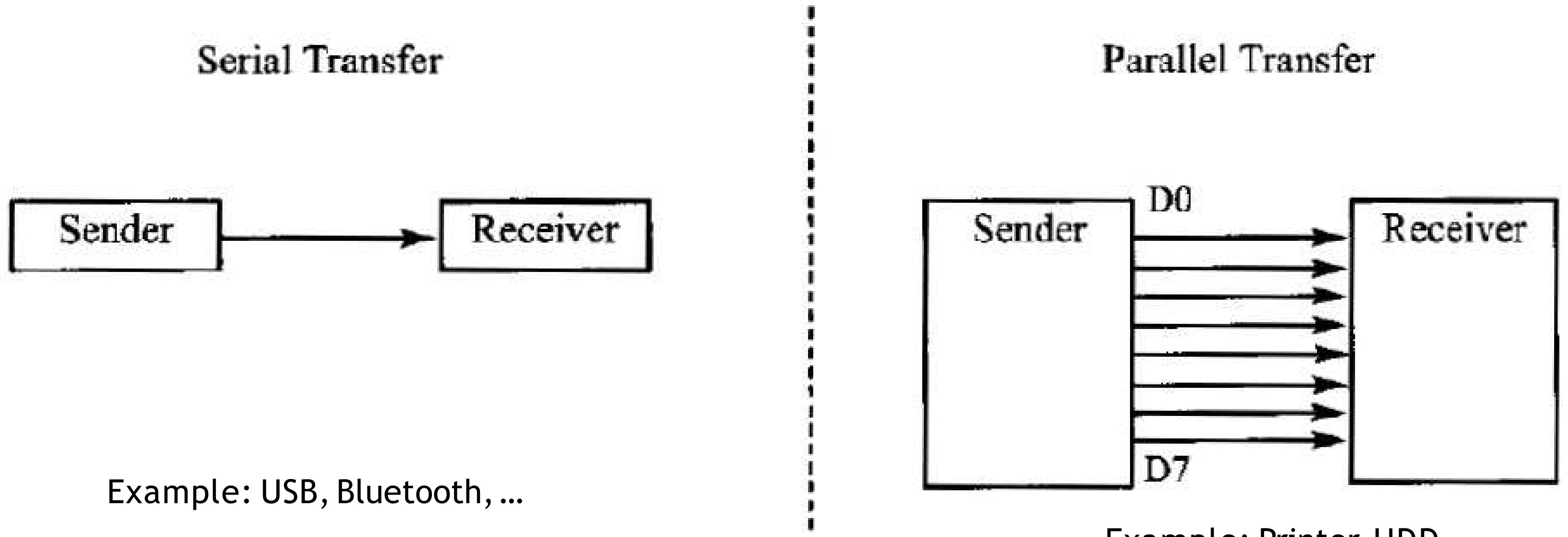
SERIAL COMMUNICATION USING ATMEGA

(Basic)

DIGITAL COMMUNICATION



Example: USB, Bluetooth, …

Example: Printer, HDD, …

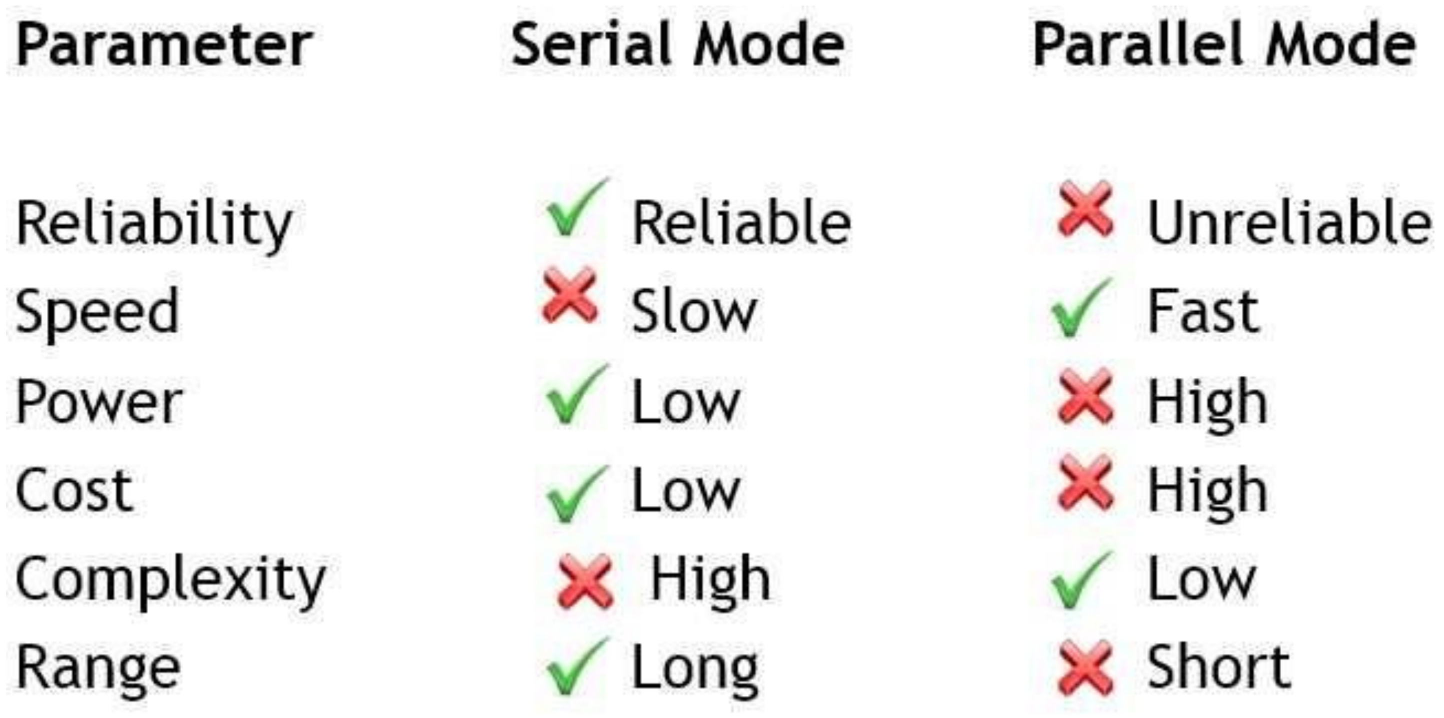
Computers transfer data in two ways: **parallel** and **serial**.

- **Parallel**: Several data bits are transferred simultaneously, e.g. printers and hard disks.
- **Serial**: A single data bit is transferred at one time, e.g. bluetooth and USB.

WHY SERIAL COMMUNICATION ???

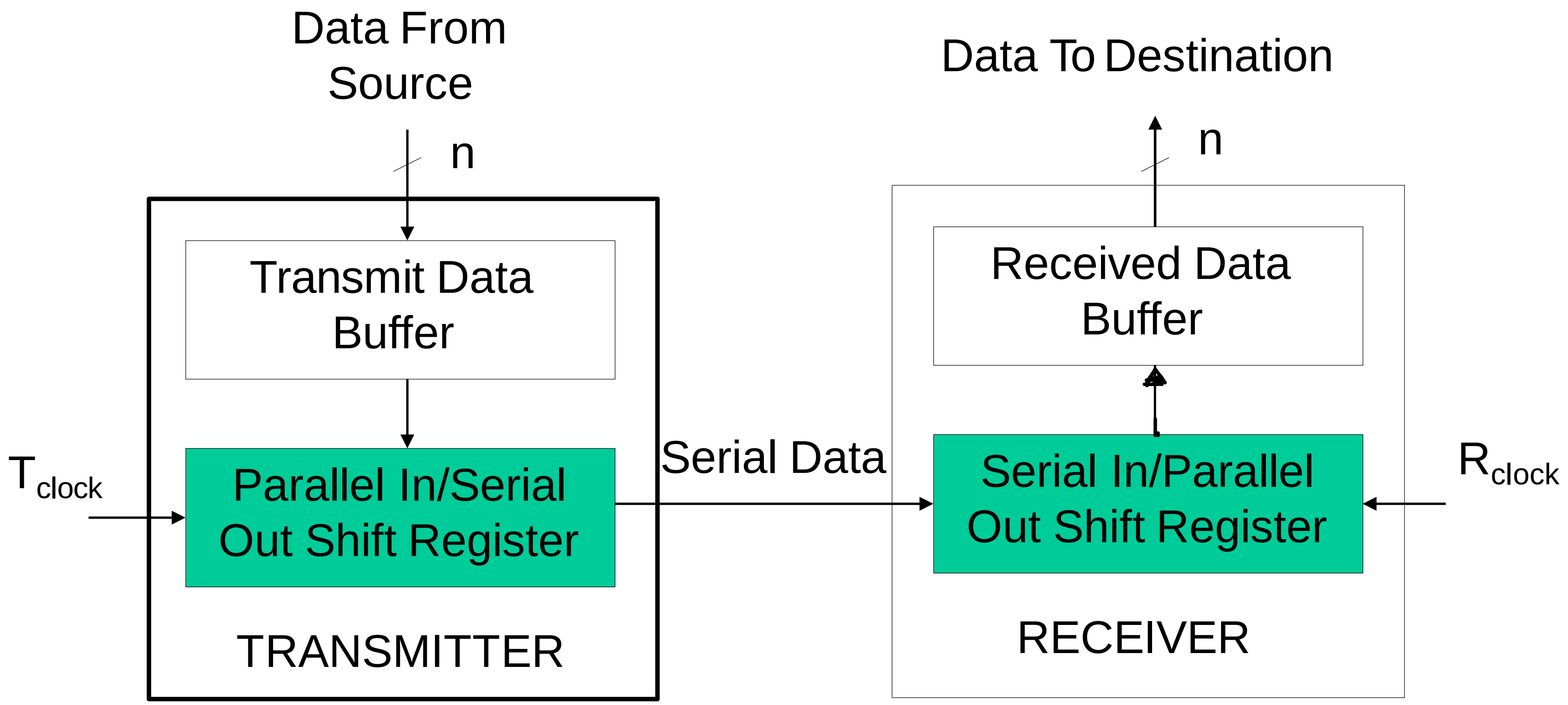
- longer distances
- easier to synchronize
- fewer IO pins
- lower cost

SERIAL VS. PARALLEL



Parameter	Serial Mode	Parallel Mode
Reliability	✓ Reliable	✗ Unreliable
Speed	✗ Slow	✓ Fast
Power	✓ Low	✗ High
Cost	✓ Low	✗ High
Complexity	✗ High	✓ Low
Range	✓ Long	✗ Short

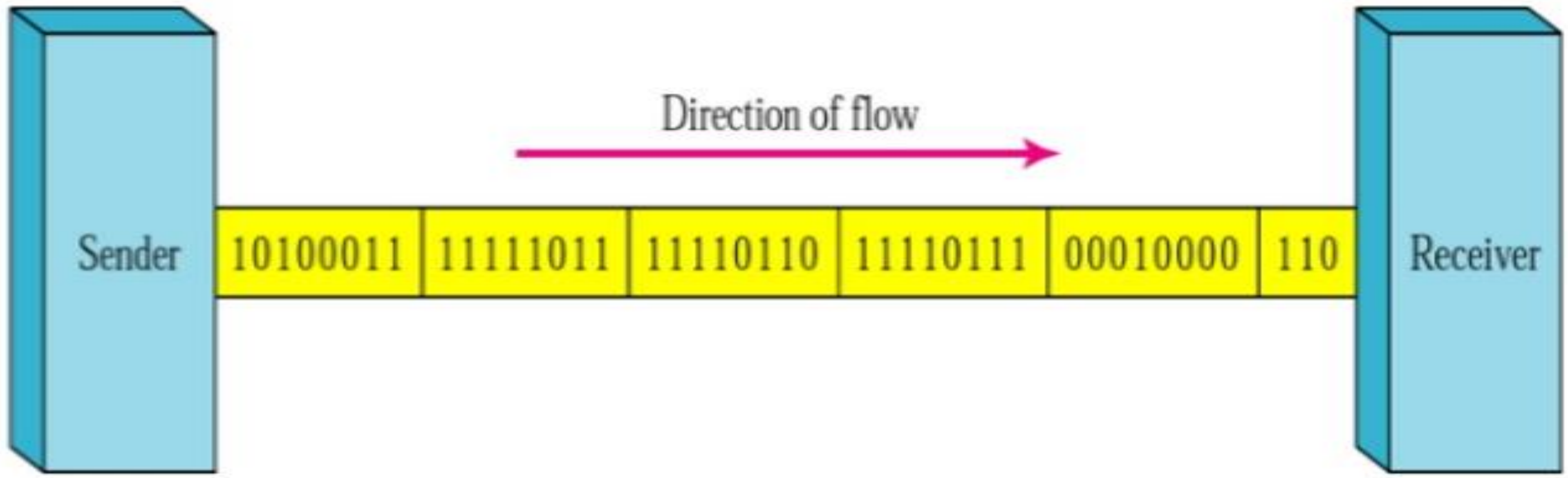
Serial Communication (System Structure)



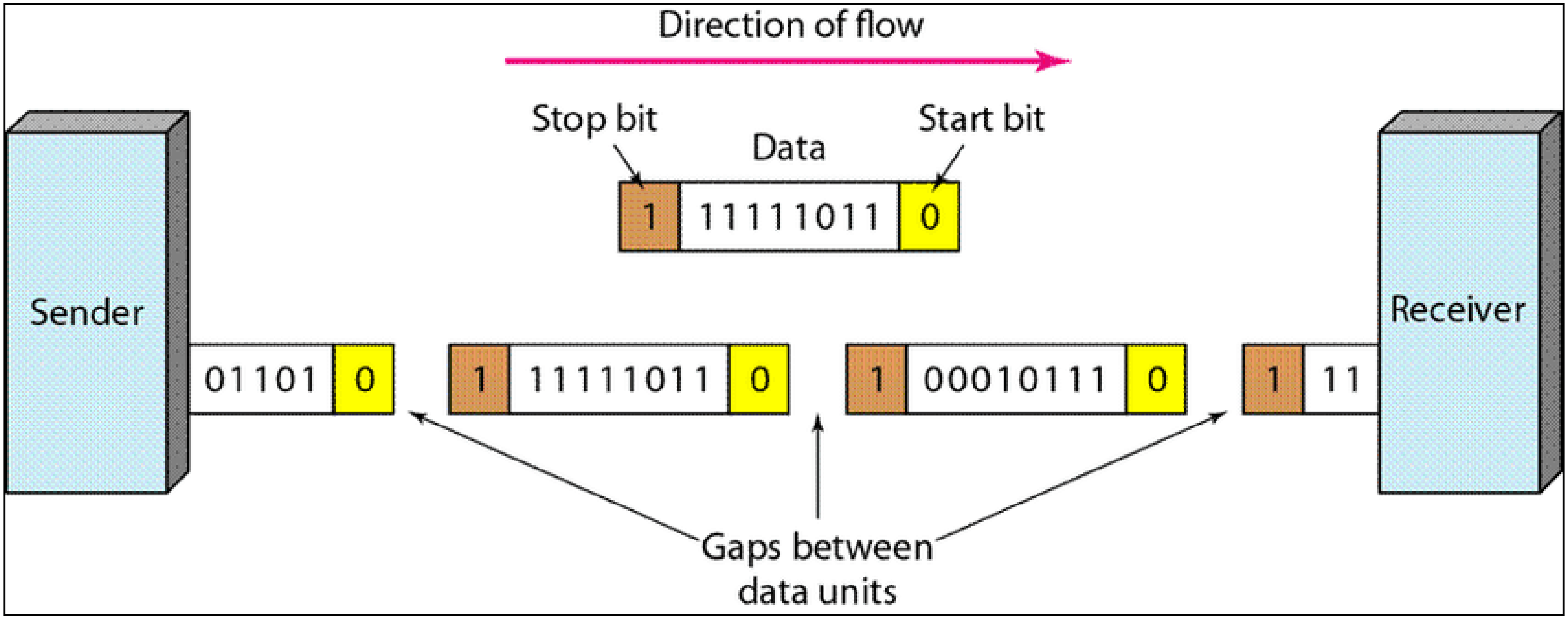
Serial Communication (Types)

- There are two basic types of serial communications
  - Synchronous
    - Sender and receiver share a common clock
    - Transfers a block of data at a time
    - Example: SPI
  - Asynchronous
    - Sender and receiver agree on a common data transmission speed
    - Transfers a single byte at a time
    - Much slower than synchronous
    - Example: UART

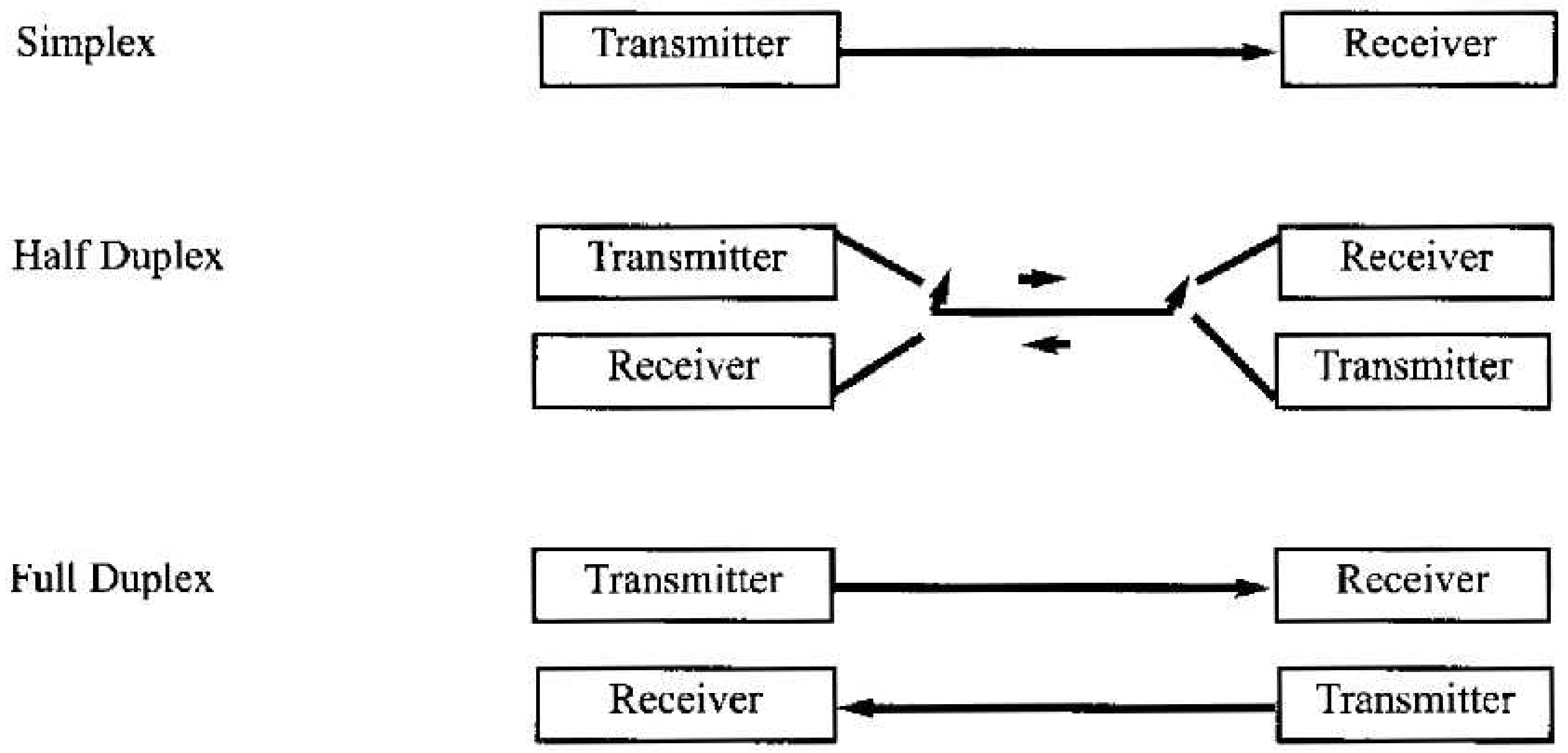
Synchronous Transmission



Asynchronous Transmission



Communication System Types



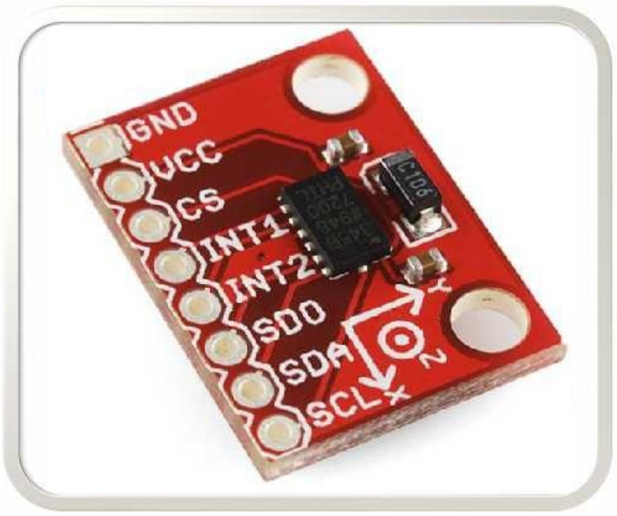
Terminologies (for Serial Communication)

- Parity bit
  - A single bit that is sent together with data bits to make the total number of 1's even (for even parity) or odd (for odd parity)
  - used for error checking.
- Start bit
  - to indicate the start of a character. Its typical value is 0.
- Stop bit
  - to indicate the end of a character. Its typical value is 1
- Baud rate
  - the number of bits sent per second (bps)

SERIAL COMMUNICATIONS IN ATMEGA32

- ATmega32 provides three subsystems for serial communications
  - Universal Synchronous & Asynchronous Serial Receiver & Transmitter (USART)
  - Serial Peripheral Interface (SPI)
  - Two-wire Serial Interface (TWI)

JPG Color Camera: Uses UART to communicate



Accelerometers: Communication through SPI or TWI

Universal Synchronous & Asynchronous Serial Receiver & Transmitter (USART)

Characteristics:

- Supports full-duplex mode between a receiver and transmitter
- Typically used in asynchronous communication.
  - Called **UART**
  - We focus on UART only!!
- Start bit and stop bit are used for each byte of data

Serial Peripheral Interface (SPI)

Characteristics:

- The receiver and transmitter share a common clock line.
- Supports higher data rates.
- The transmitter is designated as the master, the receiver as the slave.

Two-wire Serial Interface (TWI)

Characteristics:

- Network several devices such as microcontrollers and display boards, using a two-wire bus.
- Up to 128 devices are supported.
- Each device has a unique address and can exchange data with other devices in a small network.

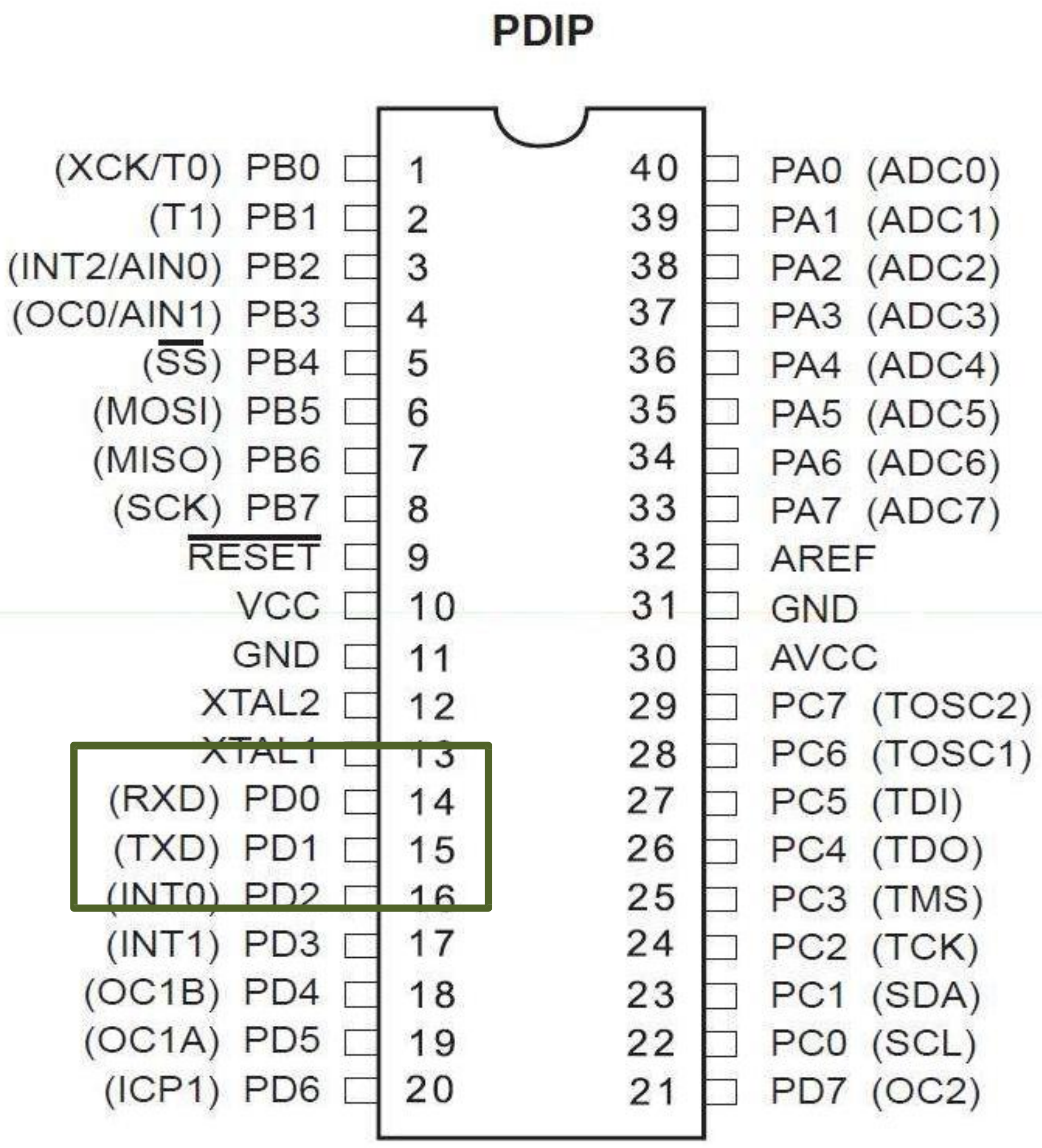
Universal Synchronous & Asynchronous Serial Receiver & Transmitter (USART)

(Intermediate)

USART IN ATMEGA32

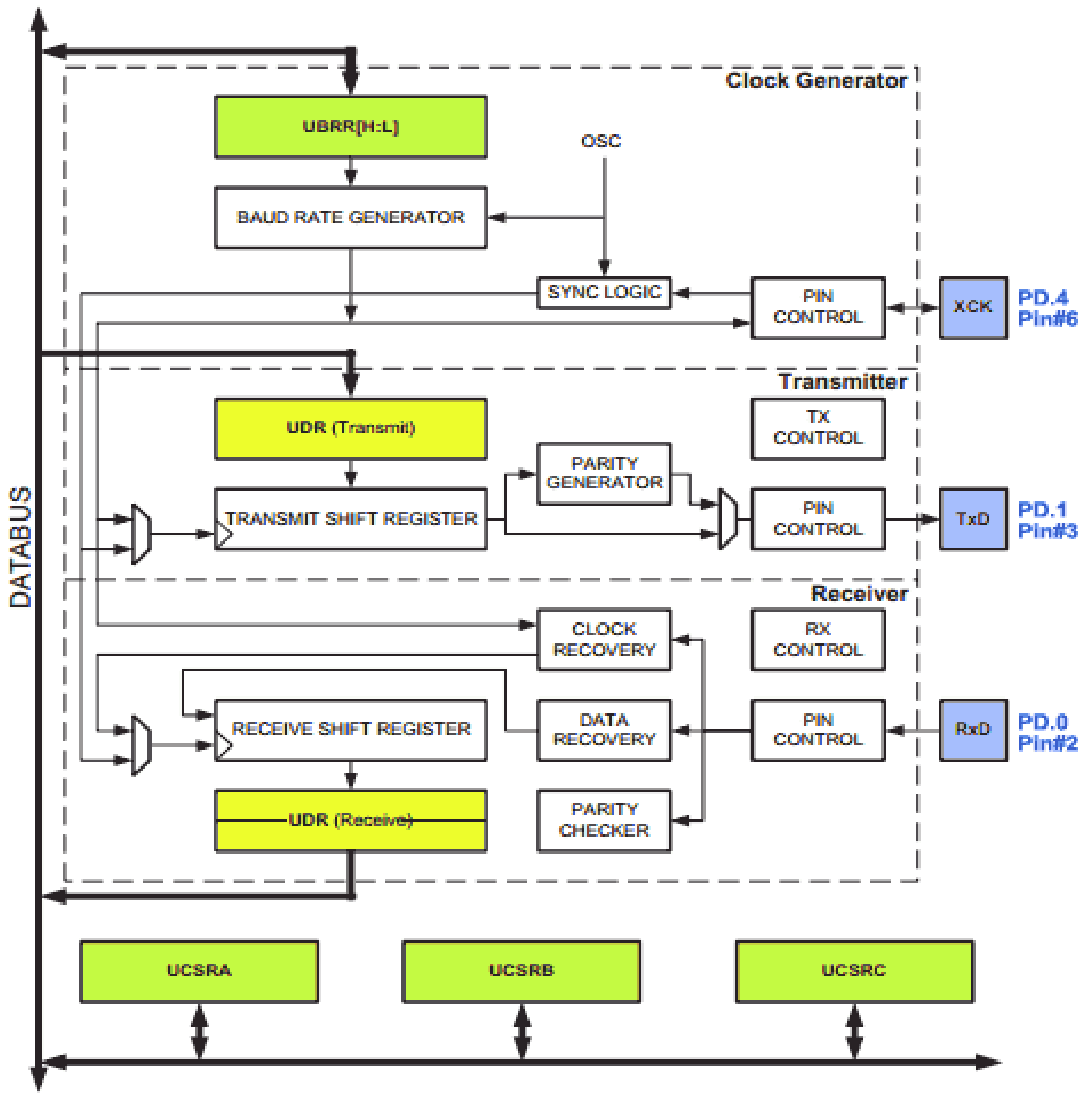
- baud rates from 960bps up to 57.6kbps
- character size: 5 to 9 bits
- 1 start bit
- 1 or 2 stop bits
- parity bit (optional: even or odd parity)

RELEVANT PINS



- RXD
  - To recieve
- TXD
  - To transmit

USART – Architecture



Clock Generator

Transmitter

Receiver

Registers
(Colored cells)

HARDWARE ELEMENTS

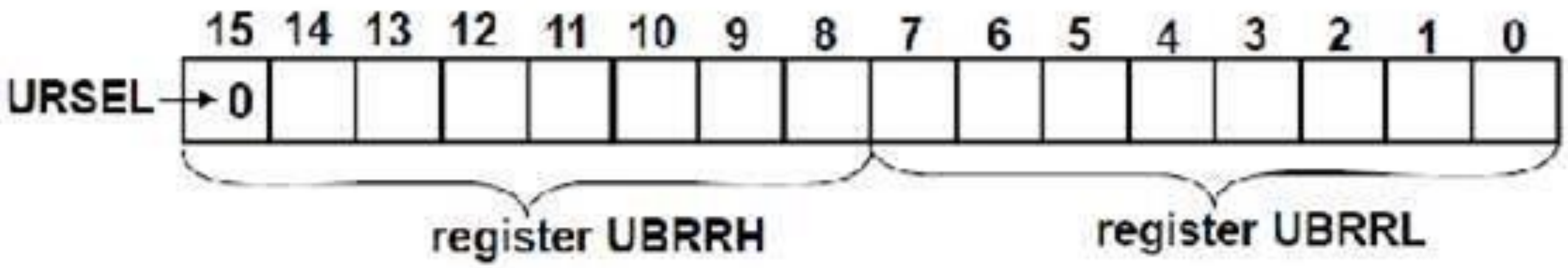
- USART Clock Generator
  - to provide clock source
  - to set baud rate using UBRR register
- USART Transmitter:
  - to send a character through TxD pin
  - to handle start/stop bit framing, parity bit, shift register.
- USART Receiver:
  - to receive a character through RxD pin
  - to perform the reverse operation of the transmitter
- USART Registers:
  - to configure, control and monitor the serial USART

USART – Resistors

- USART Baud Rate Registers
  - UBRRH and UBRRL
- USART Control and Status Registers
  - UCSRA
  - UCSRB
  - UCSRC
- USART Data Registers
  - UDR
- Understanding these registers is essential in using the serial port. Therefore, we'll study these registers in depth.

USART BAUD RATE REGISTERS

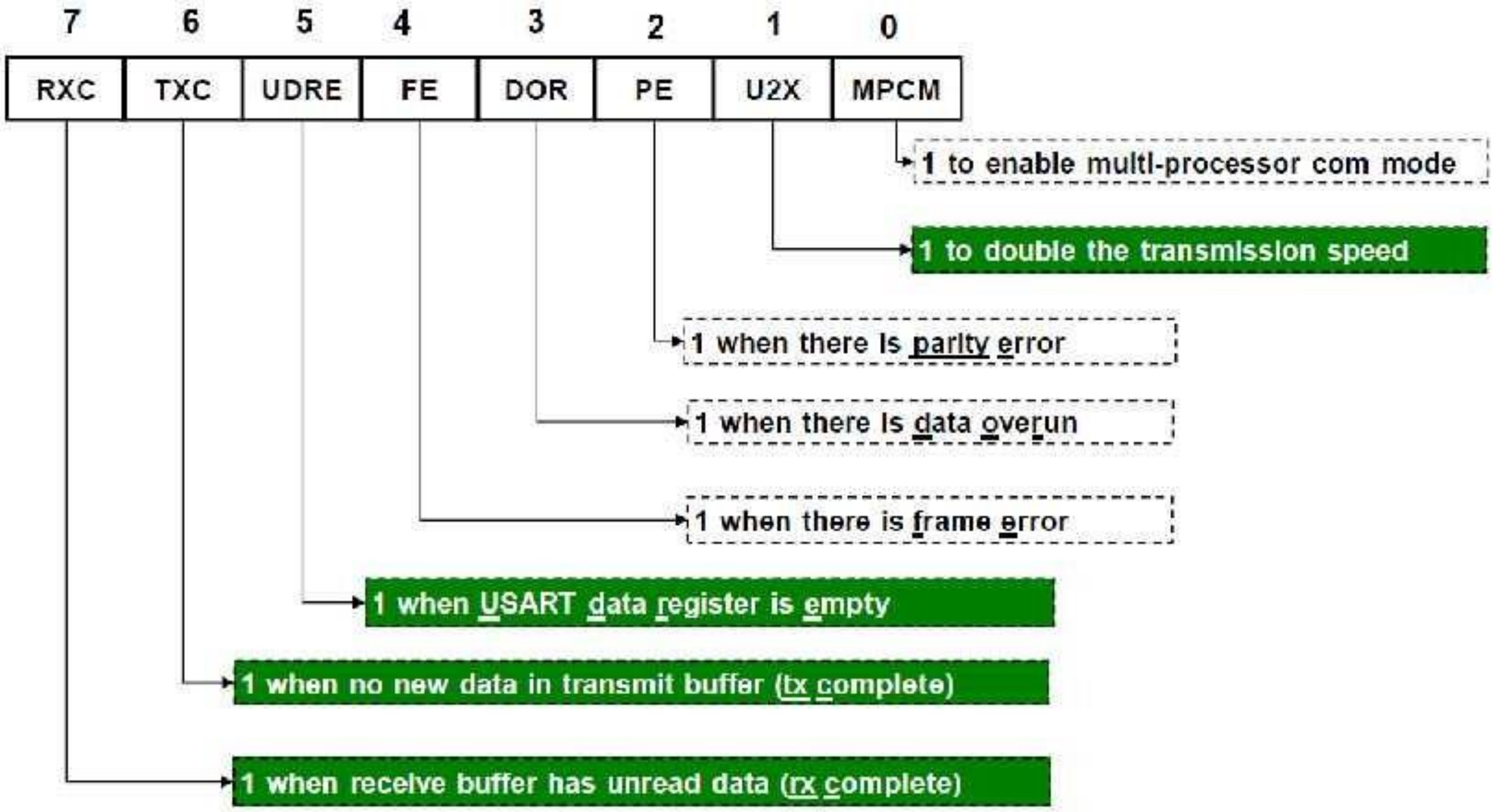
- Two 8-bit registers together define the baud rate.



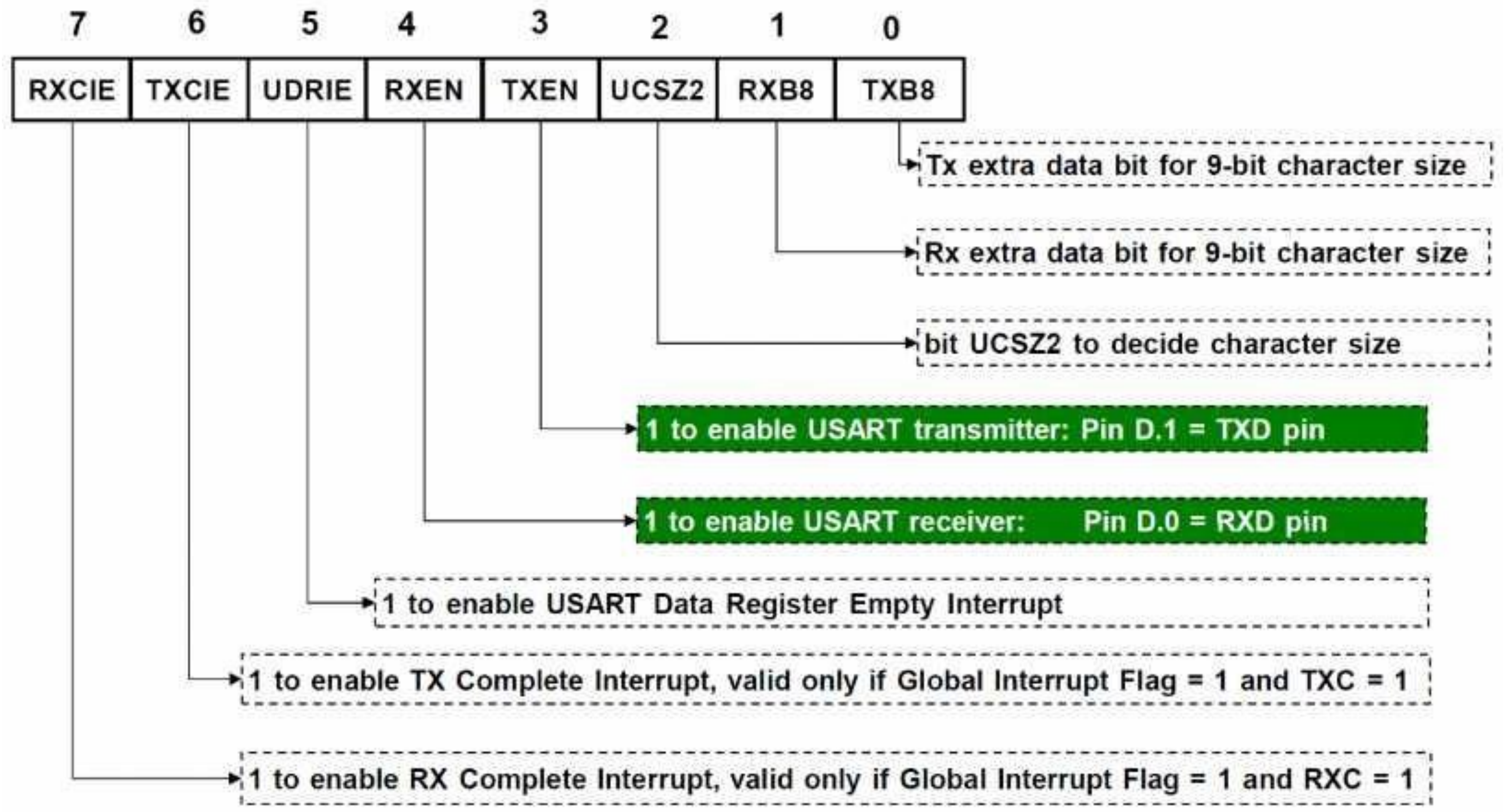
$$\text{UBRR} = \frac{\text{system clock frequency (Hz)}}{16 \times \text{baud rate}} - 1$$

- Example: Find UBRR registers for baud rate of 1200bps, assuming system clock is 1MHz.
  - UBRR = 1000000/(16 × 1200) — 1 = 51_d = 0033_H.
  - Therefore, UBRRH = 00_H and UBRRL = 33_H.
  - C code: `UBRRH = 0x00; UBRRL = 0x33;`

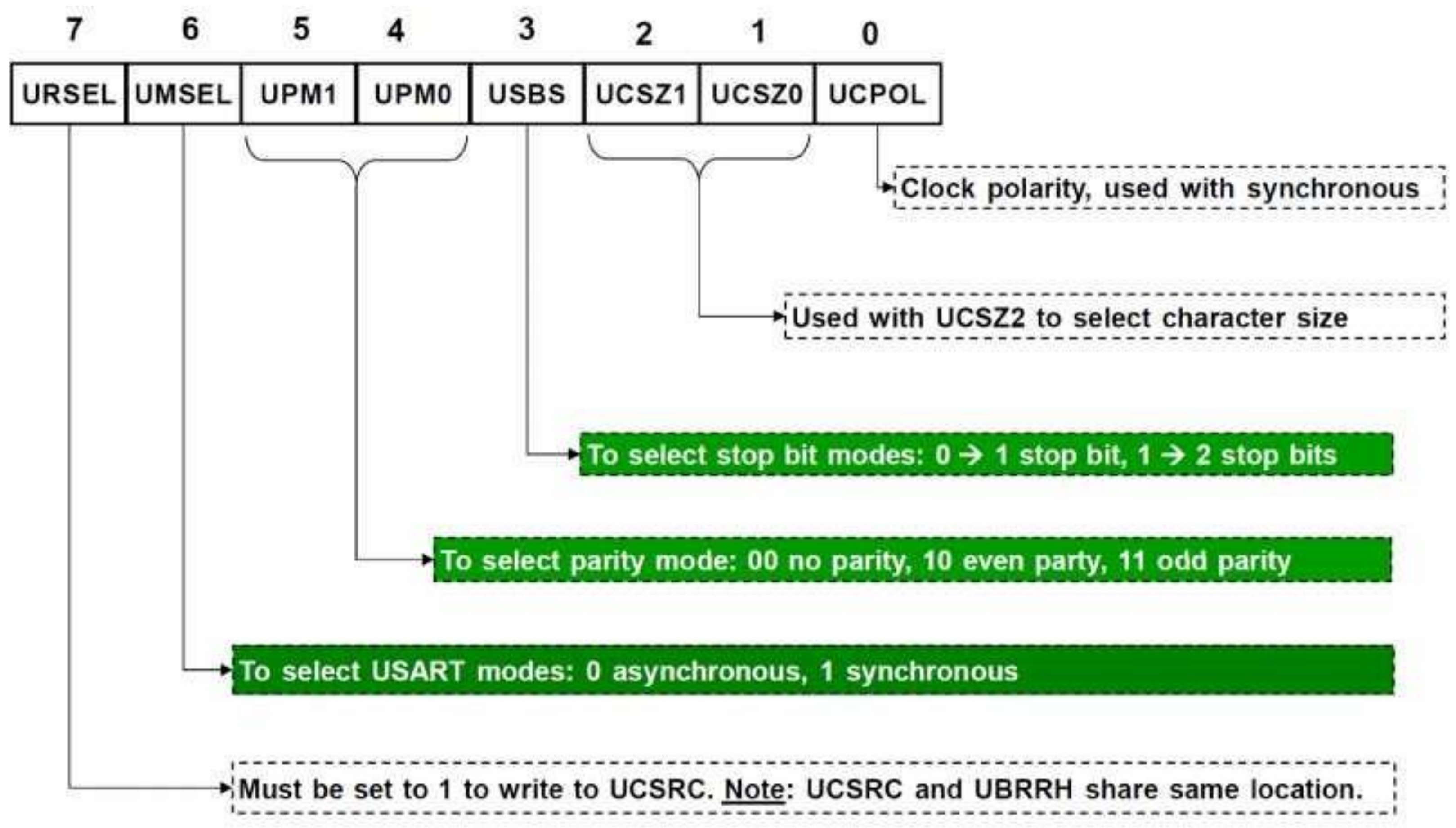
USART CONTROL AND STATUS REGISTER A: UCSRA


7
6
5
4
3
2
1
0
RXC
TXC
UDRE
FE
DOR
PE
U2X
MPCM
1 to enable multi-processor com mode
1 to double the transmission speed
1 when there is parity error
1 when there is data overun
1 when there is frame error
1 when USART data register is empty
1 when no new data in transmit buffer (tx complete)
1 when receive buffer has unread data (rx complete)

USART Control and Status Register B: UCSRB


7
6
5
4
3
2
1
0
RXCIE
TXCIE
UDRIE
RXEN
TXEN
UCSZ2
RXB8
TXB8
Tx extra data bit for 9-bit character size
Rx extra data bit for 9-bit character size
bit UCSZ2 to decide character size
1 to enable USART transmitter: Pin D.1 = TXD pin
1 to enable USART receiver: Pin D.0 = RXD pin
1 to enable USART Data Register Empty Interrupt
1 to enable TX Complete Interrupt, valid only if Global Interrupt Flag = 1 and TXC = 1
1 to enable RX Complete Interrupt, valid only if Global Interrupt Flag = 1 and RXC = 1

USART CONTROL AND STATUS REGISTER C: UCSRC


7
6
5
4
3
2
1
0
URSEL
UMSEL
UPM1
UPM0
USBS
UCSZ1
UCSZ0
UCPOL
Clock polarity, used with synchronous
Used with UCSZ2 to select character size
To select stop bit modes: 0 → 1 stop bit, 1 → 2 stop bits
To select parity mode: 00 no parity, 10 even party, 11 odd parity
To select USART modes: 0 asynchronous, 1 synchronous
Must be set to 1 to write to UCSRC. Note: UCSRC and UBRRH share same location.

SETTING CHARACTER SIZE

- Character size (5, 6, 7, 8, 9) is determined by three bits
  - bit UCSZ2 (in register UCSRB),
  - bit UCSZ1 and bit UCSZ0 (in register UCSRC).

- Example: For a character size of 8 bits, we set

  UCSZ2 = 0, UCSZ1 = 1, and UCSZ0 = 1.

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

USART DATA REGISTER

- Register UDR is the buffer for characters sent or received through the serial port.

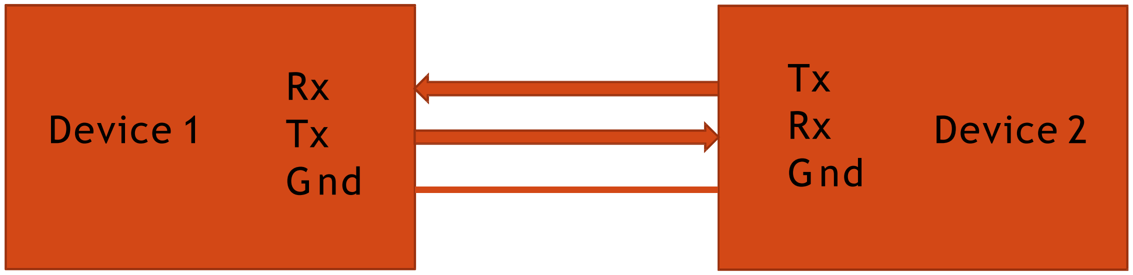
- To start sending a character, we write it to UDR:

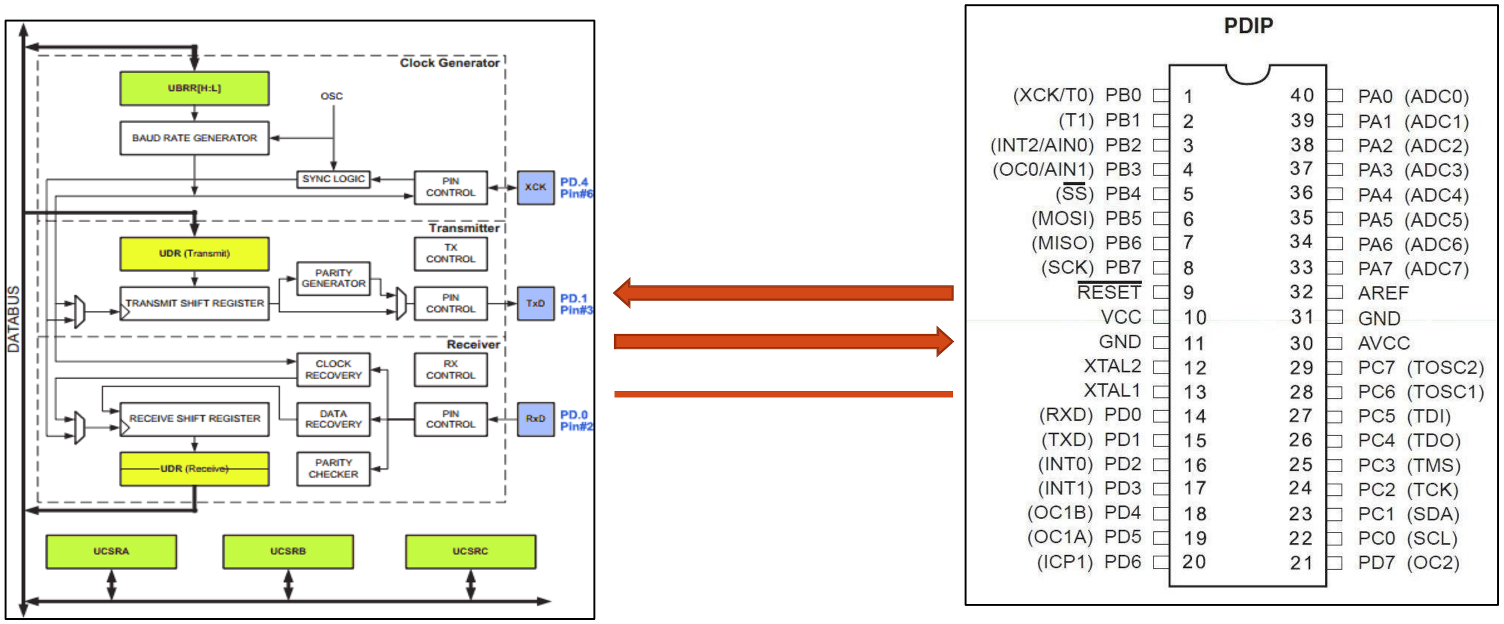
```
unsigned char data;
data = 'a';
UDR = data;         // start sending character
```

- To process a received character, we read it from UDR:

```
unsigned char data;
data = UDR;        // this will clear UDR
```

CONNECTIONS FOR UART





SERIAL PERIPHERAL INTERFACE (SPI)

SPI: SERIAL PERIPHERAL INTERFACE

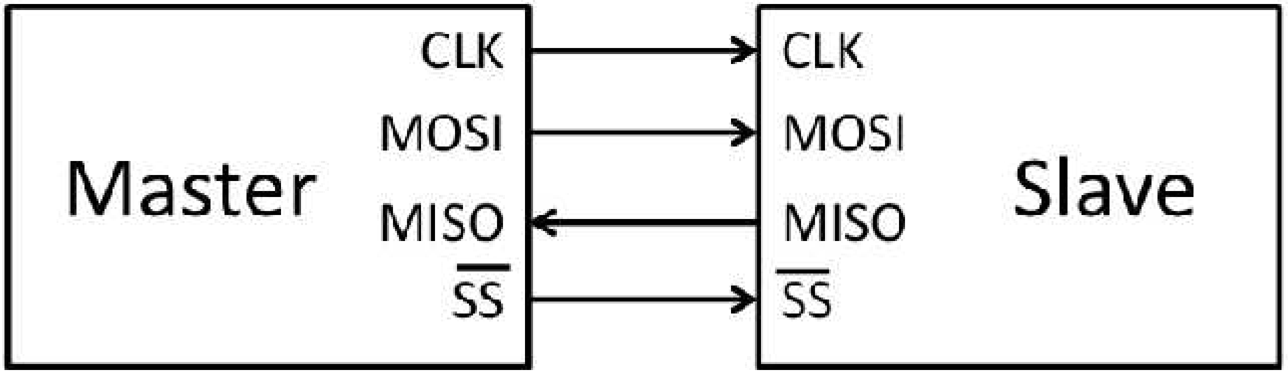
- Serial
  - Because it works on serial mode of transfer
  - It is also synchronous and full duplex
- Peripheral Interface
  - Bus interface
  - It has the capability of communicate with many nodes
- Advantages
  - Less power consumption
  - Higher speed
  - Easy tounderstand

SPI

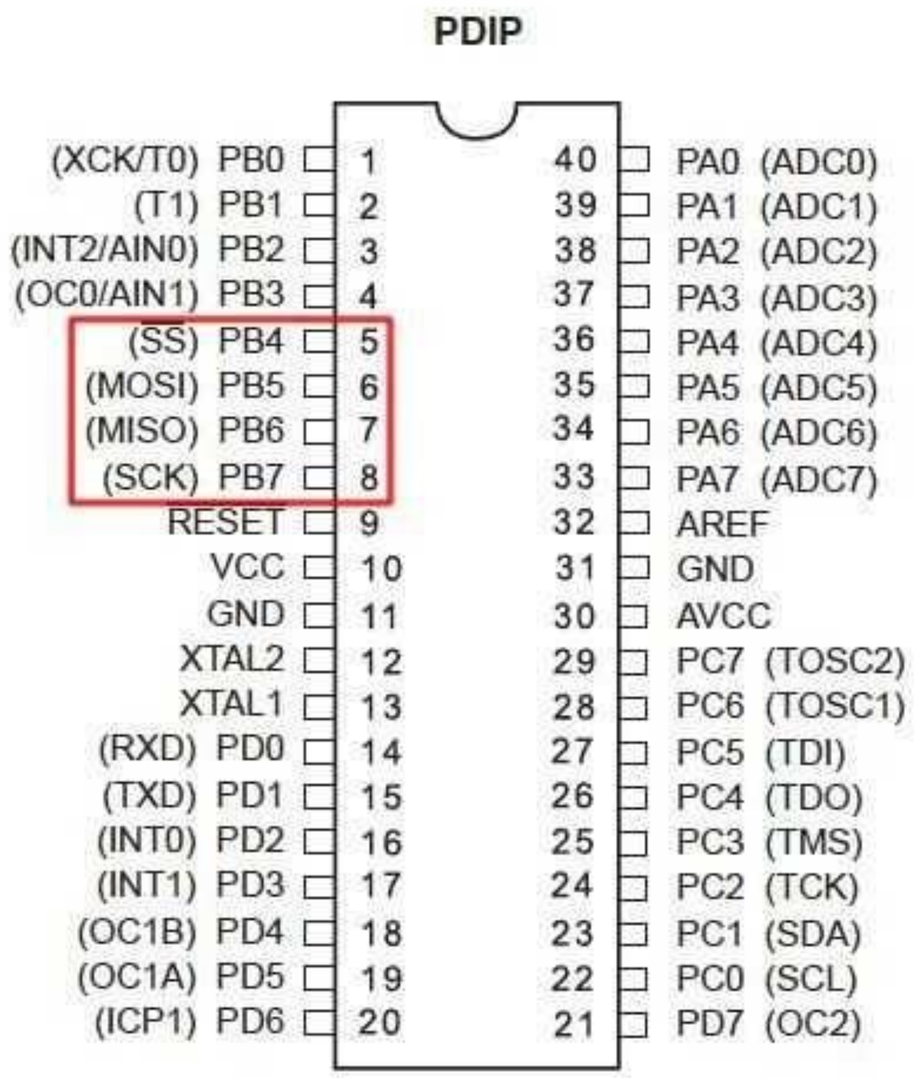
- The sender and receiver follows a master-slave relationship
- There may be multiple nodes in the network
- One node is master, the rest are slaves
- The transmission is always initiated by the master
  - Every transmission is both a send and a receive on both sides
- The slaves can communicate only with the master
- How do master selects the slave??
  - Using Chip select pin

CONNECTIONS FOR SPI

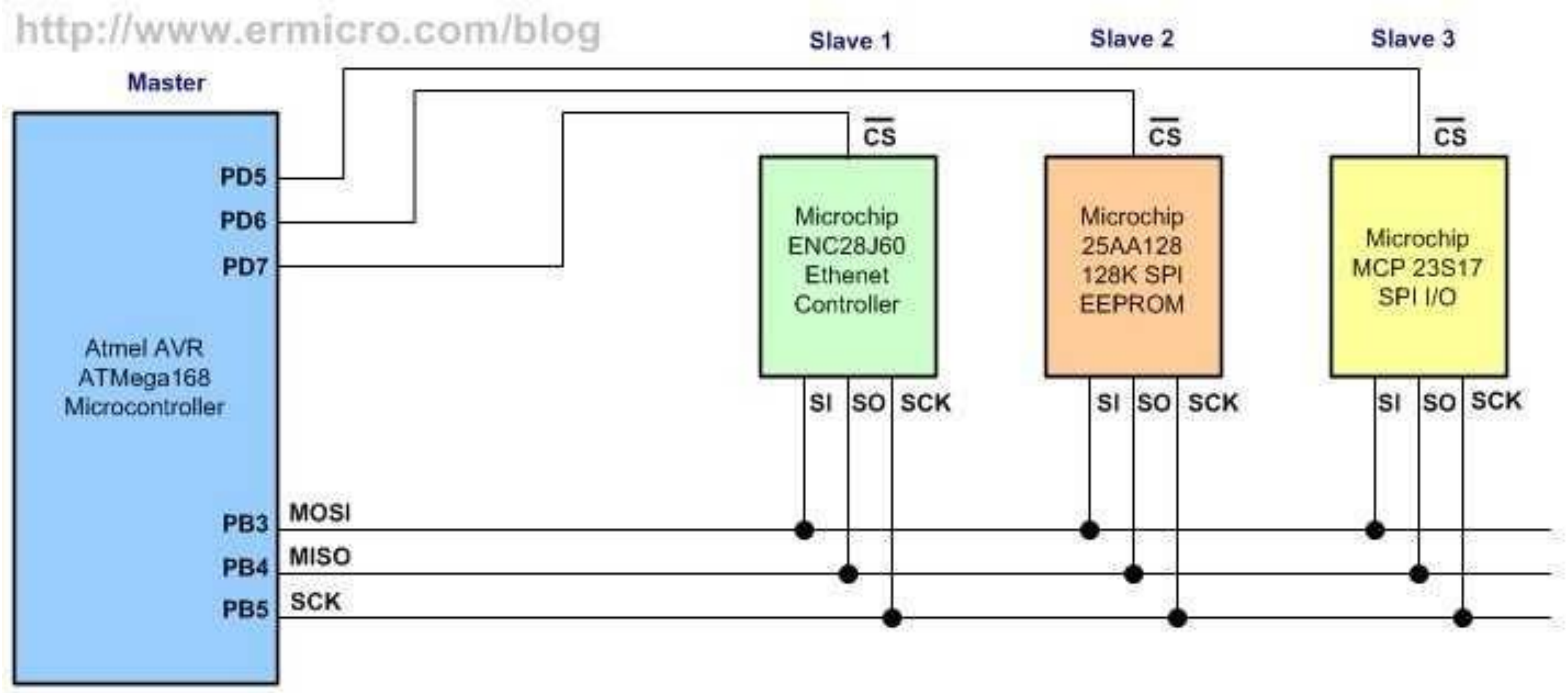
- CLK
  - Generated by Master, used by both
- MOSI is Master Out Slave In:
  - Data sent by Master to Slave
- MISO is Master In Slave Out:
  - Data sent by Slave to Master
- $\overline{SS}$ is slave select:
  - Slave communicates with Master only if this pin's value is set as LOW



RELEVANT PINS



SPI: MULTIPLE SLAVES


http://www.ermicro.com/blog
Master
Slave 1
Slave 2
Slave 3
CS
CS
CS
PD5
PD6
PD7
Atmel AVR ATMega168 Microcontroller
Microchip ENC28J60 Ethenet Controller
Microchip 25AA128 128K SPI EEPROM
Microchip MCP 23S17 SPI I/O
SI
SO
SCK
SI
SO
SCK
SI
SO
SCK
PB3
MOSI
PB4
MISO
PB5
SCK

Typical SPI Master with Multiple SPI Slave Device Connection

TWO-WIRE SERIAL INTERFACE (TWI)

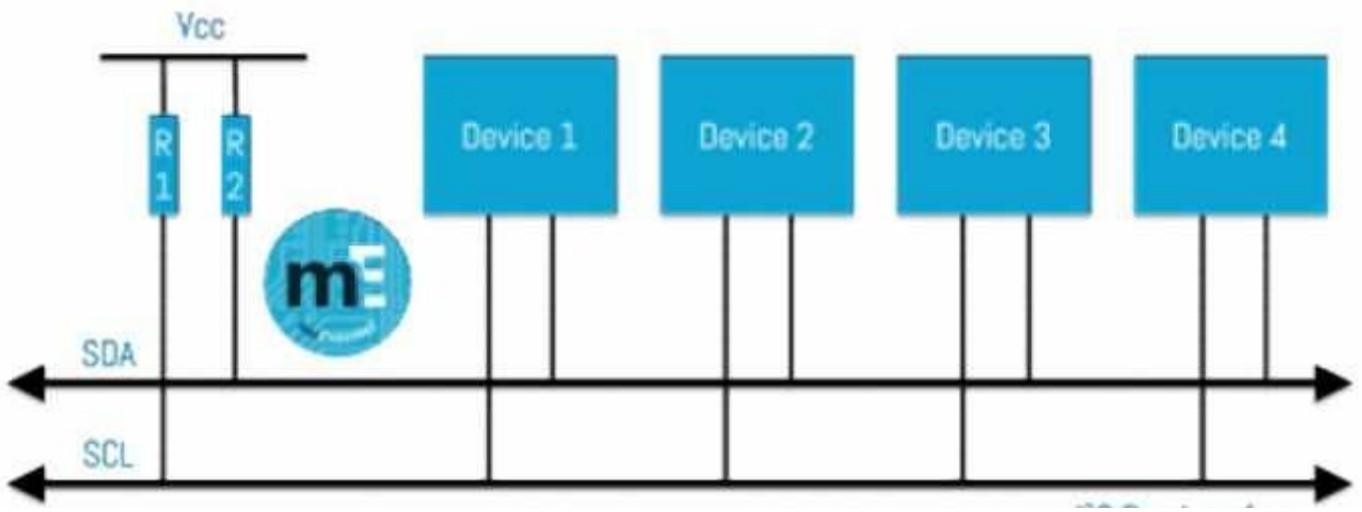
TWI: TWO-WIRE SERIAL INTERFACE

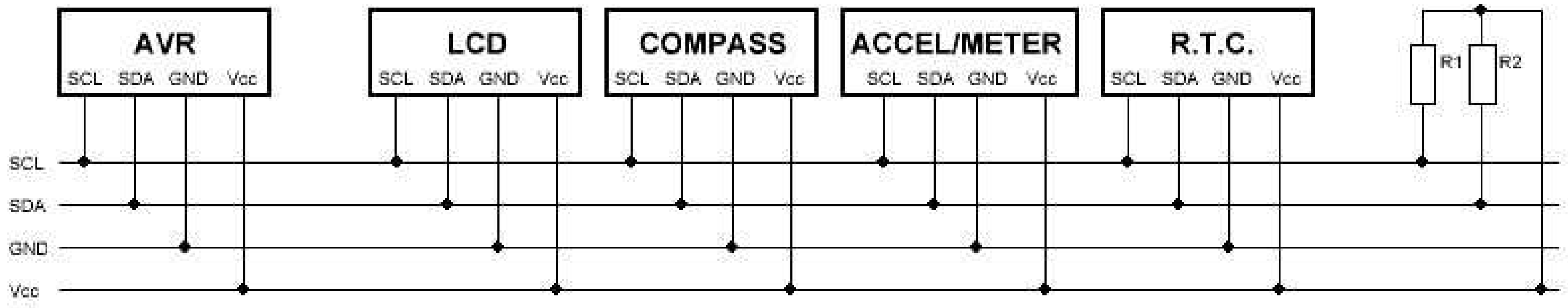
- Another bus protocol for serial communication
  - Synchronous
  - Half duplex
  - Connection-oriented?
  - Complex
- Also known as Inter-Integrated Circuit (I^2C or I2C)
- Advantages
  - Less error
  - Connect up to 128 different devices using just 2 wires

TWI

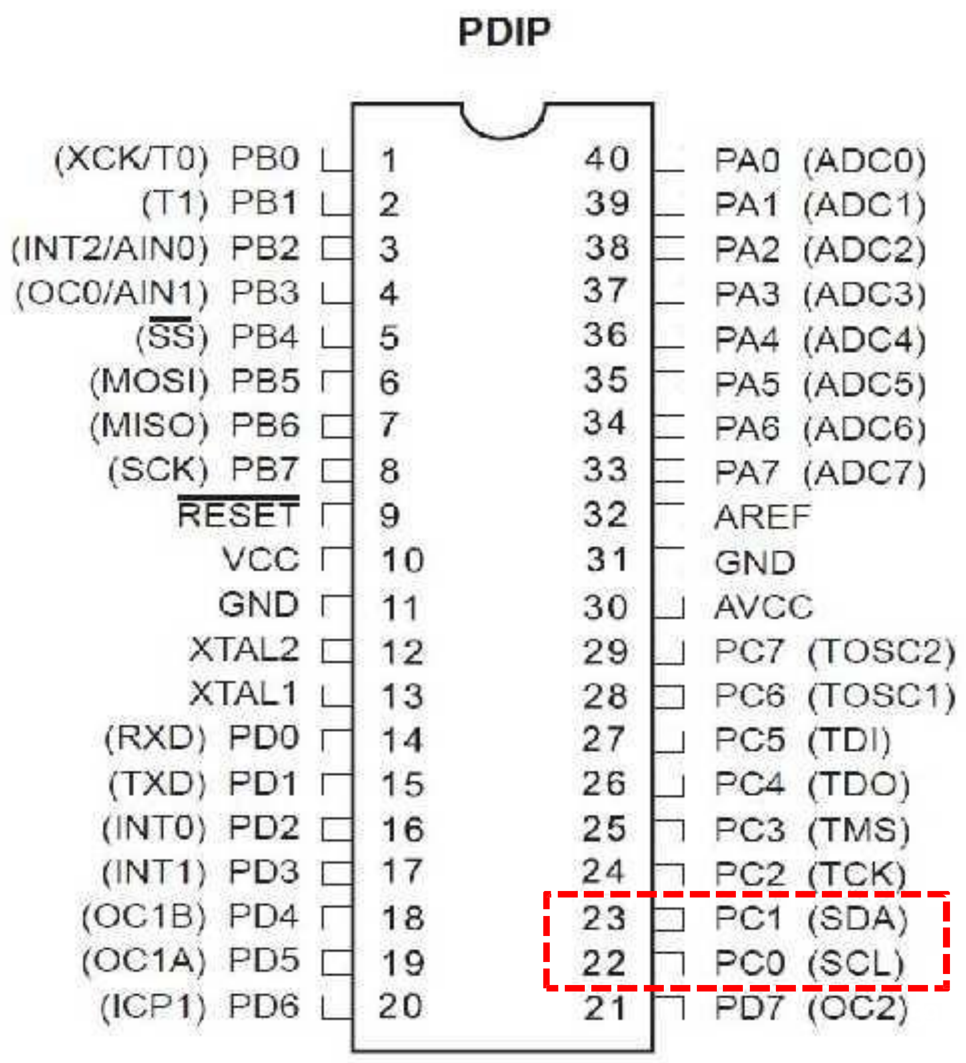
- Maximum 128 nodes in the network
  - Ideal for attaching low-speed peripherals
- Each nodes has an unique 7-bit address
- Each node can operate as either master or slave
- Master generates the clock for the system
  - initiates and terminates a transmission
- Slave receives theclock
  - addressed by the master
- The slaves can communicate only with the master
- How do master selects the slave??
  - Using address

CONNECTIONS FOR TWI

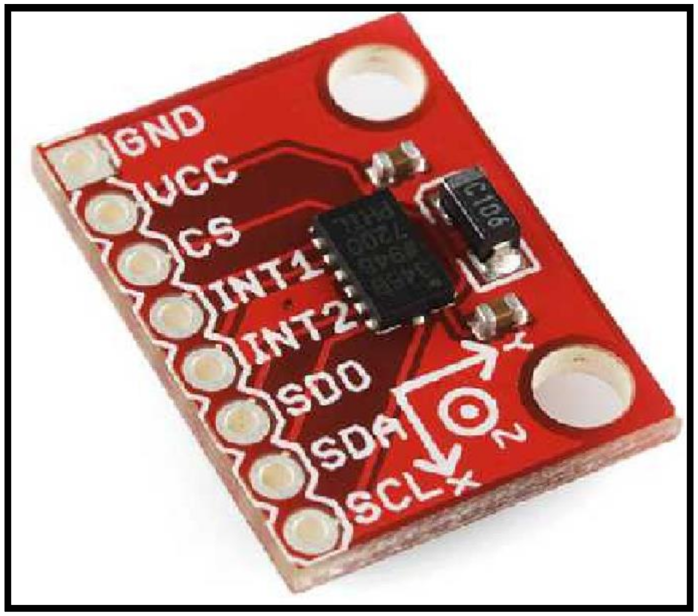


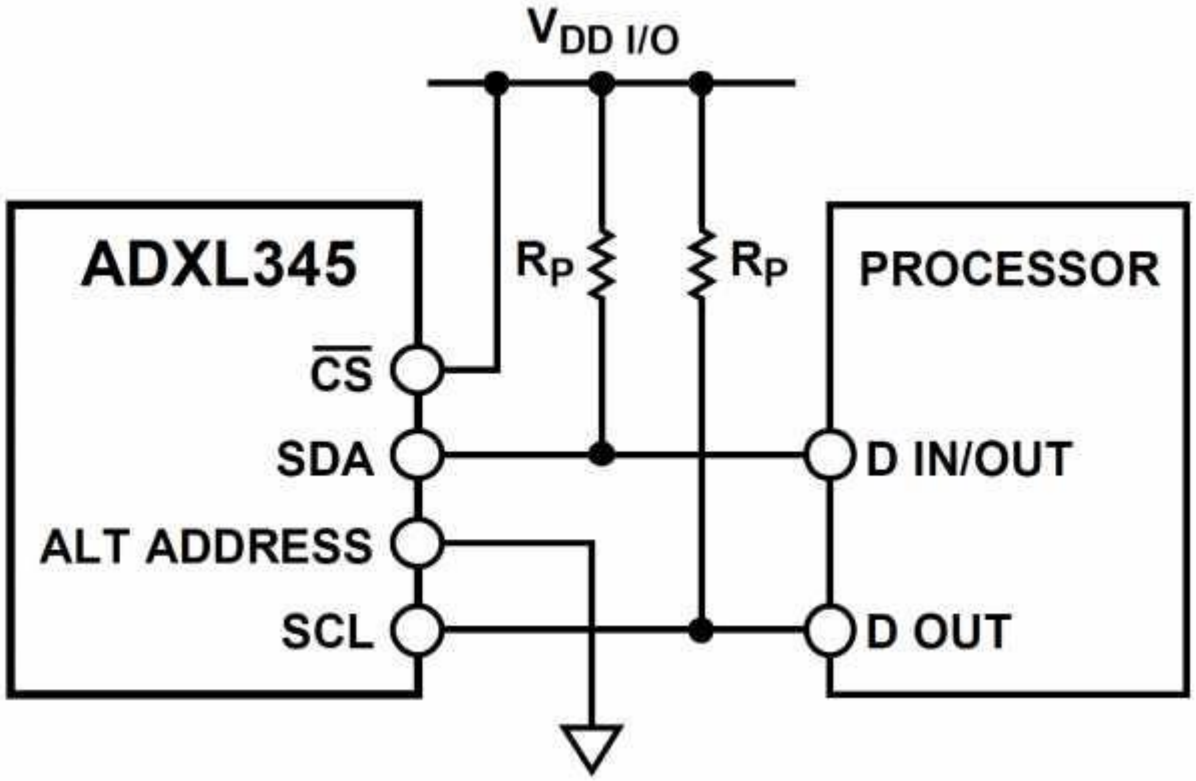


RELEVANT PINS



EXAMPLE APPLICATION




$V_{DD\ I/O}$
ADXL345
$\overline{CS}$
SDA
ALT ADDRESS
SCL
R_P
R_P
PROCESSOR
D IN/OUT
D OUT