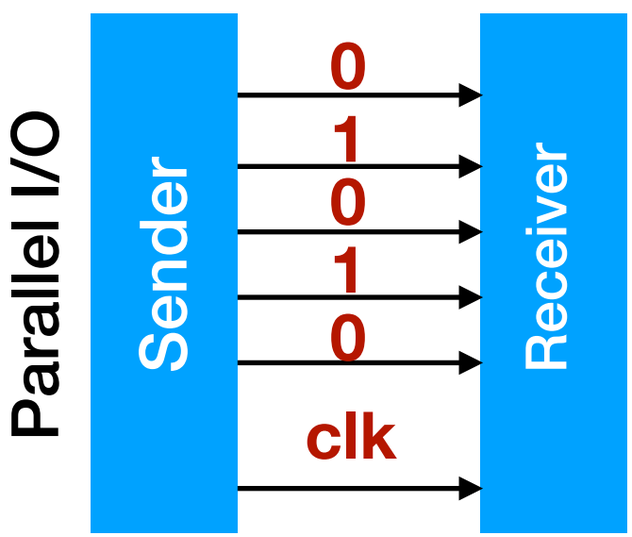
Chapter 9

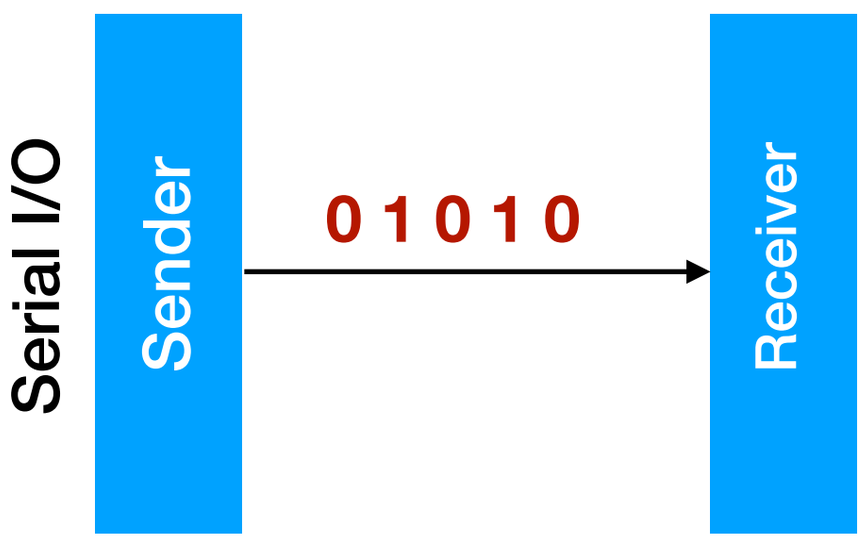
Asynchronous Communications

Yu Luo
Assistant Professor
Department of Electrical and Computer Engineering
Office: 239 Simrall Engineering Building
Email: yu.luo@ece.msstate.edu

Parallel I/O versus Serial I/O

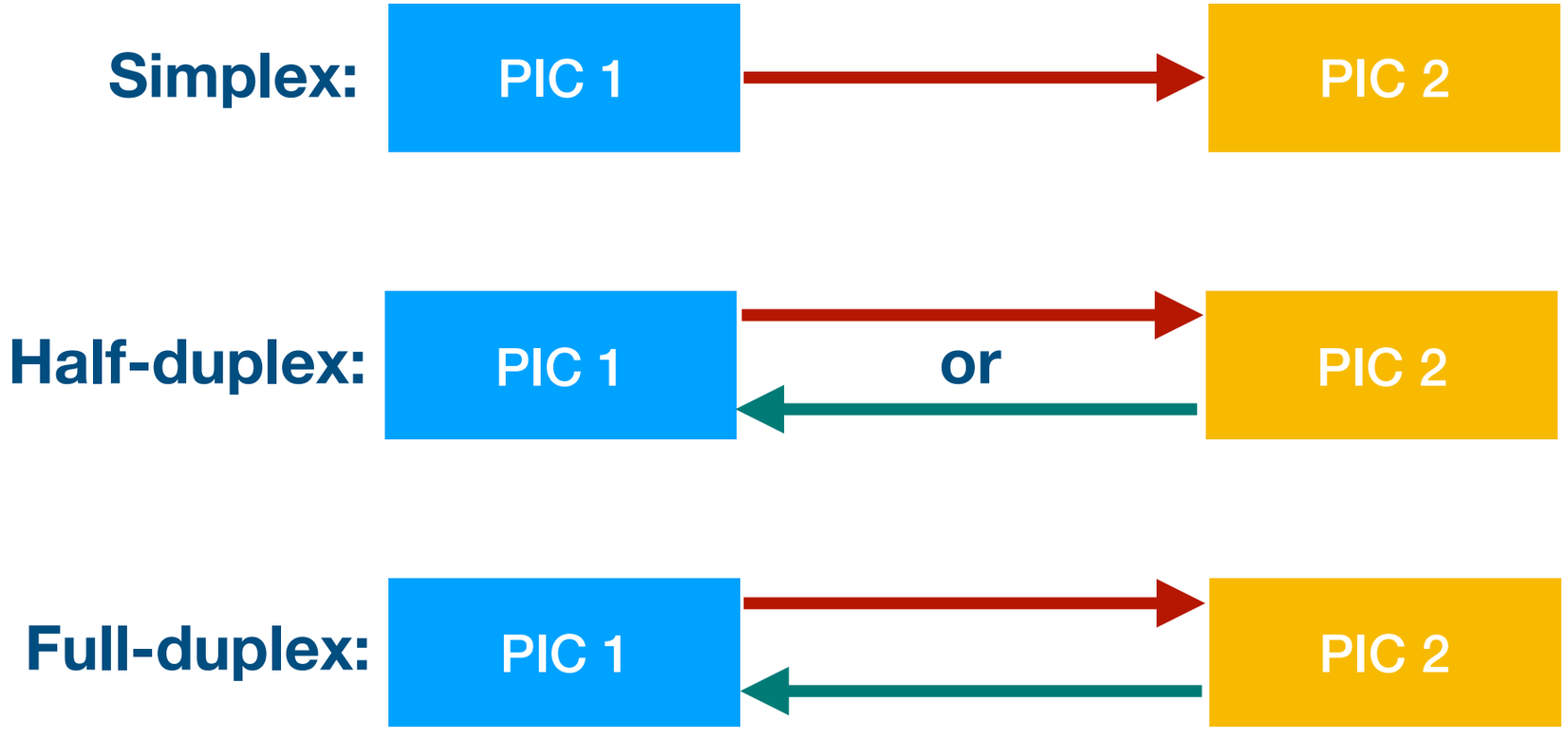
- Parallel I/O: Data sent over a group of parallel wires.
  A. Fast: Multiple bits can be transmitted simultaneously
  B. Need clock signal for synchronization
  C. High cost because cable is expensive
- Serial I/O: Data sent one bit at a time, over a single wire.
  A. Slow: Data bits need to be transmitted one-by-one
  B. A clock may or may not be used for synchronization
  C. Low cost, need only single cable





Different Communication Modes

- There are three common communication modes
  A. Simplex: Communication in one direction only
  B. Half-duplex: Communication in either direction, but only one way at a time
  C. Full-duplex: Communication in both directions at same time

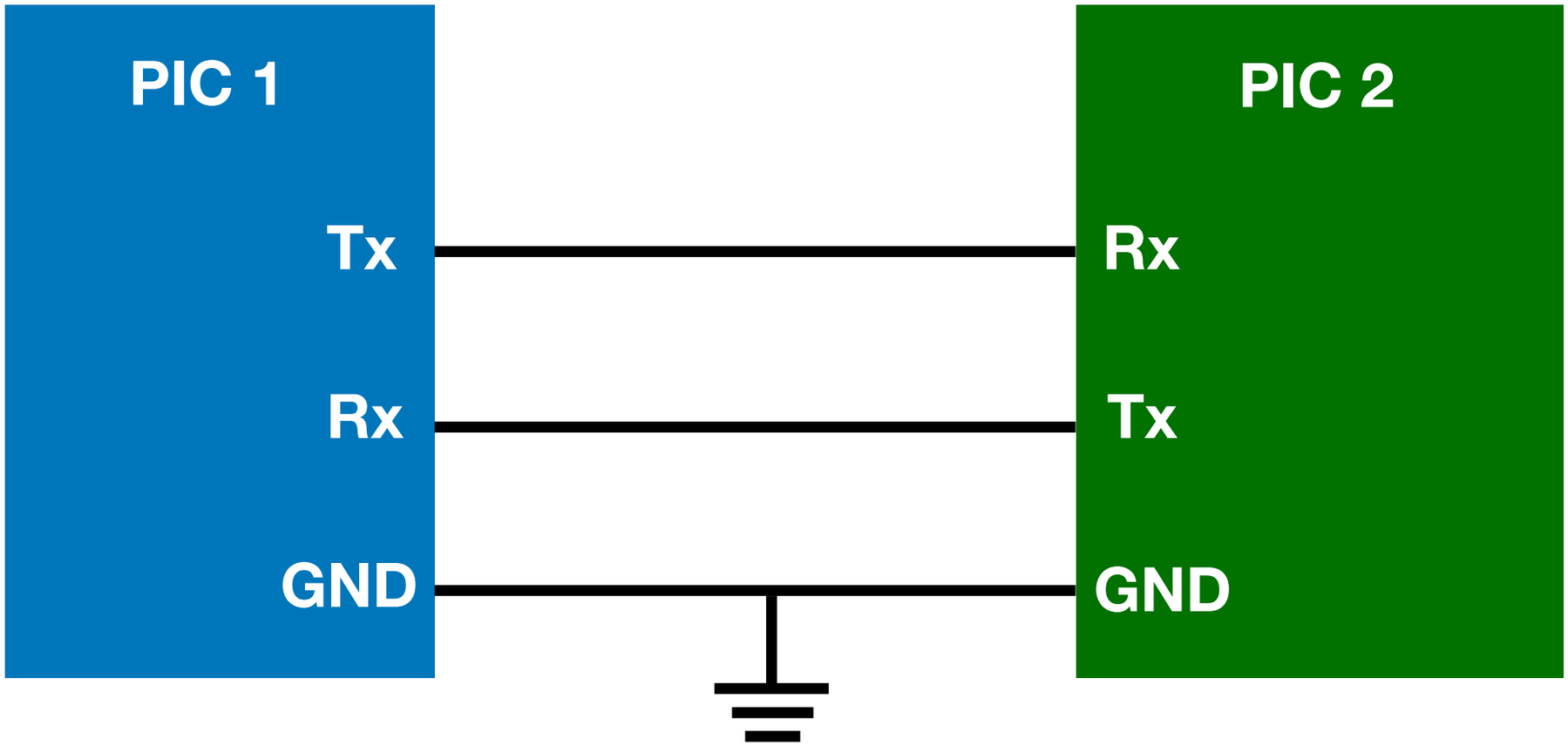


Synchronous vs Asynchronous Serial I/O

- Synchronous Serial I/O
  A. Clock signal is required to synchronize the time between the sender and the receiver
  B. Clock signal can be sent through a separate wire
  C. Receiver can also extract clock signal from data stream through a Phase- Locked-Loop (PLL)
  D. Synchronous serial IO can achieve high speeds. All new high speed serial standards are synchronous

- Asynchronous Serial I/O
  A. Does not transmit the clock signal on a separate wire
  B. It is easy to implement, but is slower than synchronous serial standards
  C. Asynchronous Serial I/O is used in older standards, but it is still very popular today

Three-Wire Async Serial Interface



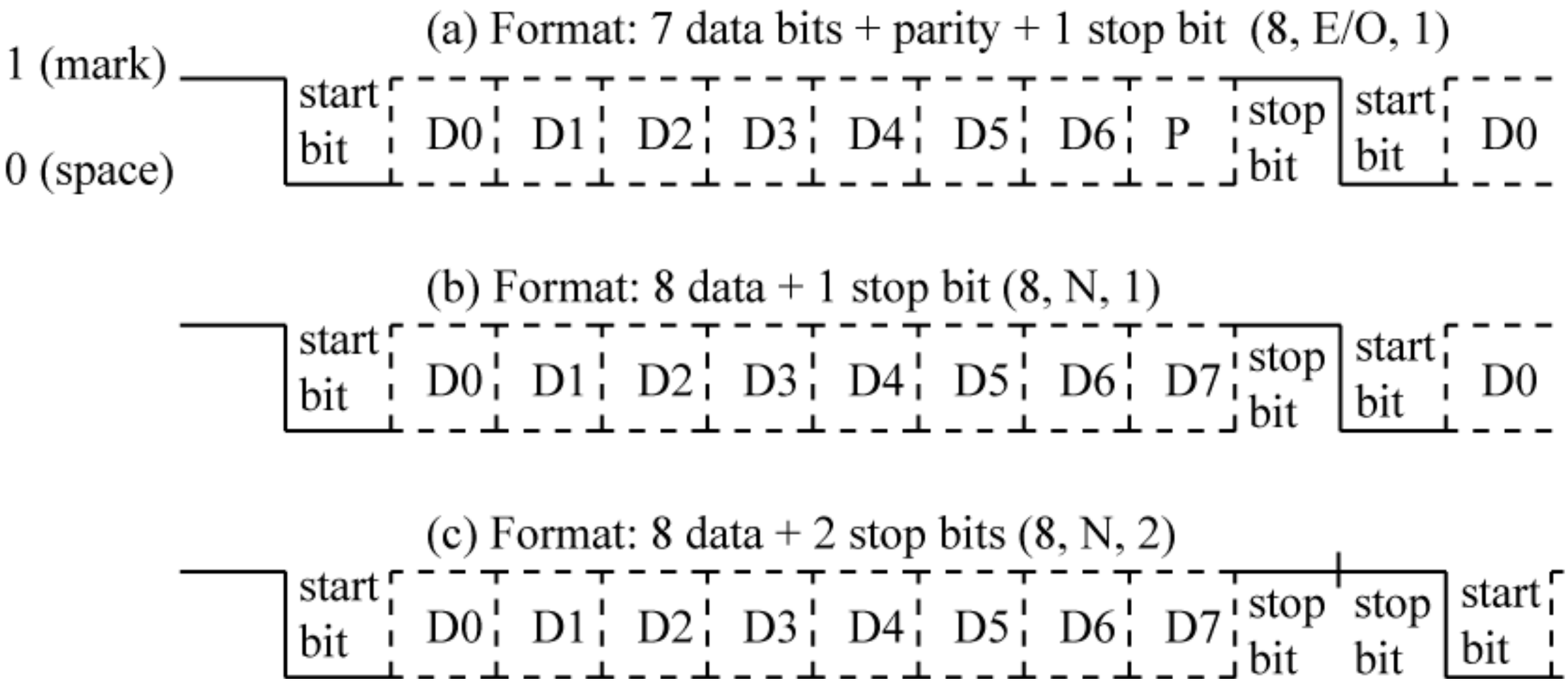
- A version of async serial interface standard is known as RS-232
- We will use a three-wire asynchronous serial interface to connect the PIC to an external devices (e.g. PC and PIC)

Asynchronous Serial Data Formats

- Data sent LSb to MSb

  Example: Assuming data is 0b1000, then sending order is 0 –> 0 –> 0 –> 1

- There are different data formats:



Parity

- A parity bit is an extra bit added to a data frame to detect a single bit error

  A. Odd parity – parity bit value makes the total number of ‘1’ bits in the frame odd

  B. Even parity – parity bit value makes the total number of ‘1’ bits in the frame even

Example

For 7-bit data value 0x56 (0b 1010110)

A. If it is odd parity, then odd parity bit = ‘1’ —> 0b10101101

Parity bit

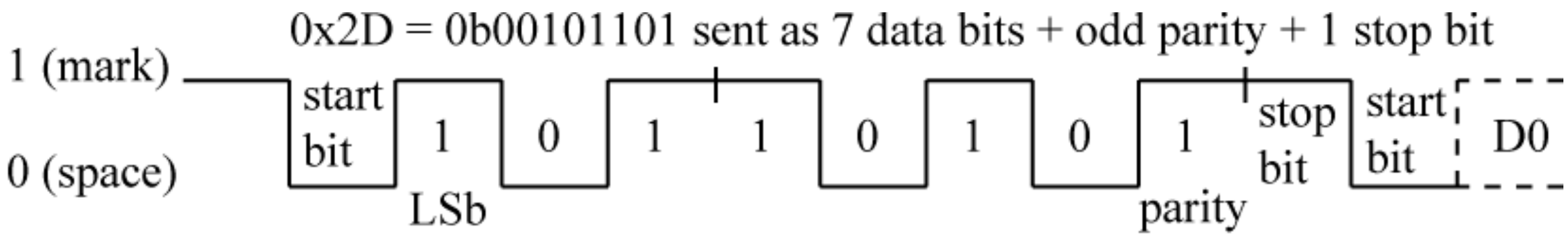
Total number of 5 (odd) ones

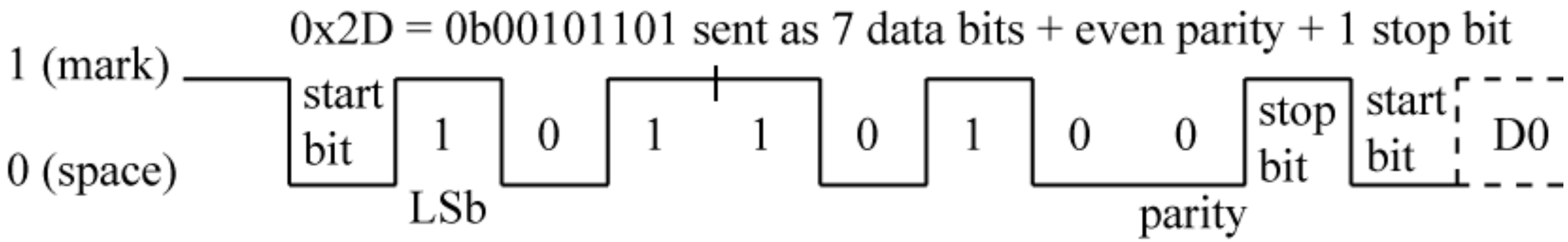
B. If it is even parity, then odd parity bit = ‘0’ —> 0b10101100

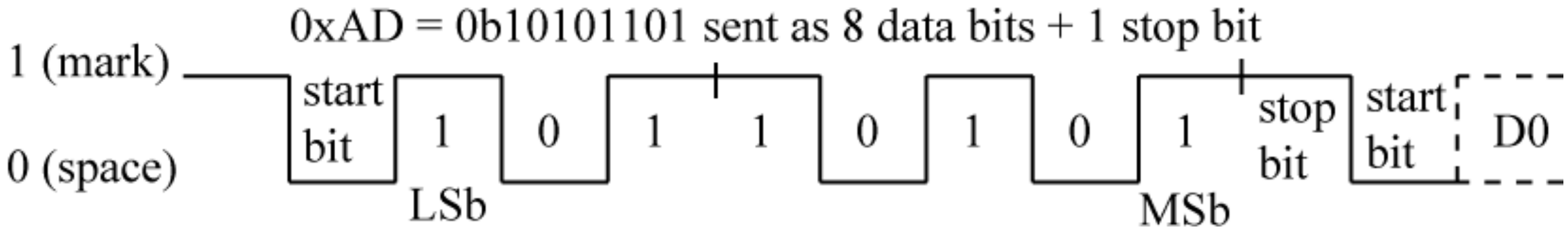
Parity bit

Total number of 4 (even) ones

Examples


0x2D = 0b00101101 sent as 7 data bits + odd parity + 1 stop bit
1 (mark)
0 (space)
start bit
1
0
1
1
0
1
0
1
stop bit
start bit
D0
LSb
parity


0x2D = 0b00101101 sent as 7 data bits + even parity + 1 stop bit
1 (mark)
0 (space)
start bit
1
0
1
1
0
1
0
0
stop bit
start bit
D0
LSb
parity


0xAD = 0b10101101 sent as 8 data bits + 1 stop bit
1 (mark)
0 (space)
start bit
1
0
1
1
0
1
0
1
stop bit
start bit
D0
LSb
MSb

Baud Rate vs Bits Rate

- Baud rate is the rate at which symbols (signaling event) are sent
- Bits rate (bps) is the number of bits sent per second

Example

Use 4 different voltage levels, send two bits of data per symbol:

-15 V = 00 -5 V = 01 +5 = 10 +15 = 11

If 1000 symbols are sent, then baud rate = 1000 and data rate = 2000 bps

- If only a ‘1’ or ‘0’ is sent for each signaling event, then baud rate = bit rate
- The effective data rate is the rate at which total bits are transferred, minus the overhead bits (i.e., start, stop, and parity bits)

Examples

- Assuming each symbol represents 1 bit. What is a bit time (in μs) for a baud rate of 57600?

Bit time = 1/baud rate

=1/57600

=1.736 x 10^{-5} seconds

=17.36 μs

- Assuming a data format of 1 start + 8 data + 1 stop. How long does it take to send 20 bytes effective data at a baud rate of 19200?

Total bits = 20 x (1 + 8 + 1) = 20 x 10 = 200 bits

Total time = Total bits x 1 bit time = 200 x 1/19200 = 10.42 ms

PIC µC UARTx

- UART = Universal Asynchronous Receiver Transmitter
- UART is hardware module that implements asynchronous serial IO
- PIC has multiple UART modules: e.g., UART1, UART2…
- PIC can transmit and receive data through UART while processor can do other tasks

Frees the processor from data transmission and reception

UART Registers

- U1RXREG:
  A. 8-bit register for UART1
  B. It holds a received data (1 byte)
  C. Read this register to get the received data, e.g., u8_a = U1RXREG
- U1TXREG:
  A. 8-bit register for UART1
  B. Write to this register to send a data (1 byte), e.g., U1TXREG = u8_b
- U1BRG: It sets the baud rate for UART1
- U1MODE: It contains configuration bits for the UART1 module
- U1STA: It contains status bits for the UART1 module

Set Baud Rate

- **UxBRG**: It is a **16-bit** register, which sets the baud rate for UARTx

$$\text{Baud rate} = \frac{\text{Clock frequency}}{S \times (\text{UxBRG} + 1)}$$

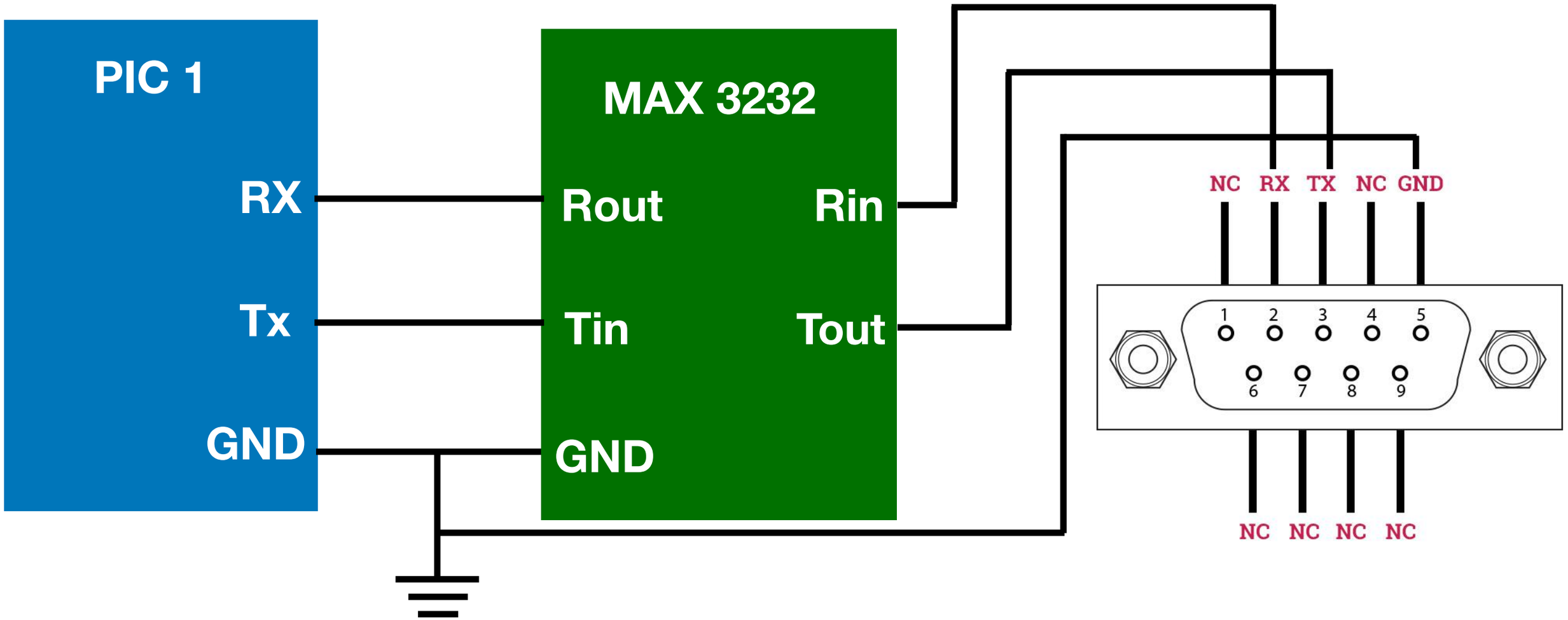
$$\text{UxBRG} = \frac{\text{Clock frequency}}{S \times \text{Baud rate}} - 1$$

In above equations, $S = 16$ (low speed mode) or $S = 4$ (high speed mode)

Example: Let UART work at low speed mode, then set the baud rate of UART1 based on the variable *buadRate*

```
static inline void CONFIG_BAUDRATE_UART1 (uint32 baudRate)
{
    U1MODEbits.BRGH = 0;      // U1MODEbits.BRGH = 0 —> low speed mode
    uint32 brg = (FCY/baudRate/16) - 1;
    U1BRG = brg;   // U1BRG is 16-bit register. So,make sure that brg  < 65536
}
```

RS-232: PIC µC to PC Serial I/O Connection

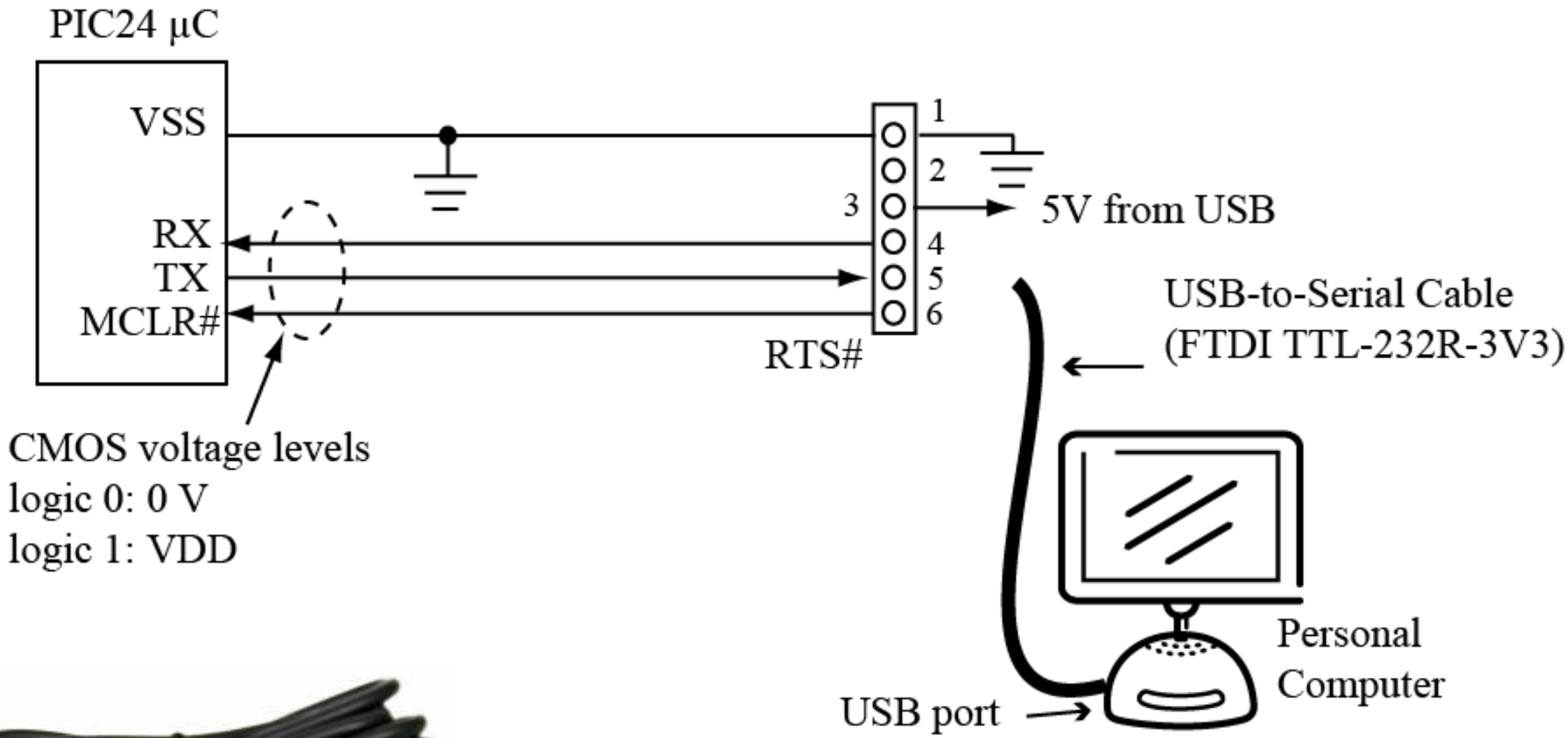


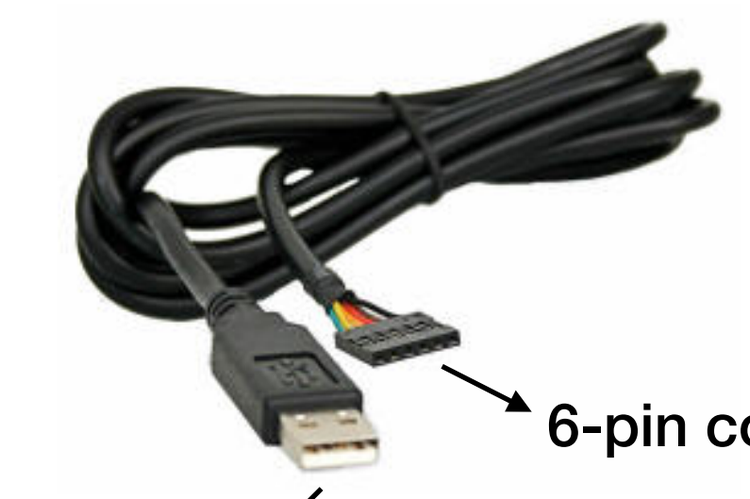
PIC uses CMOS voltage levels, which are different from RS-232 voltage levels

CMOS voltage levels:	RS232 voltage levels:
Logic 0: 0 V	Logic 0: 3 V to 25 V
Logic 1: VDD (1.8 V - 3.8 V)	Logic 1: -3 V to -25 V

We need to use MAX 3232 to convert between CMOS voltage and RS 232 voltage

USB to Serial: PIC µC to PC Serial I/O Connection





- RS232 connectors on PCs have been replaced by USB
- There is a processor built into the connector that converts from USB to asynchronous serial.