


# **CSSE2010/CSSE7201**

## **Lecture 18**

### **Serial Input/Output**

School of Information Technology and Electrical Engineering  
The University of Queensland

# Today

- Serial I/O
  - Serial communications basics
  - Serial Communication on the AVR 
- Analog to Digital Conversion (ADC)
  - ADC basics
  - ADC on AVR

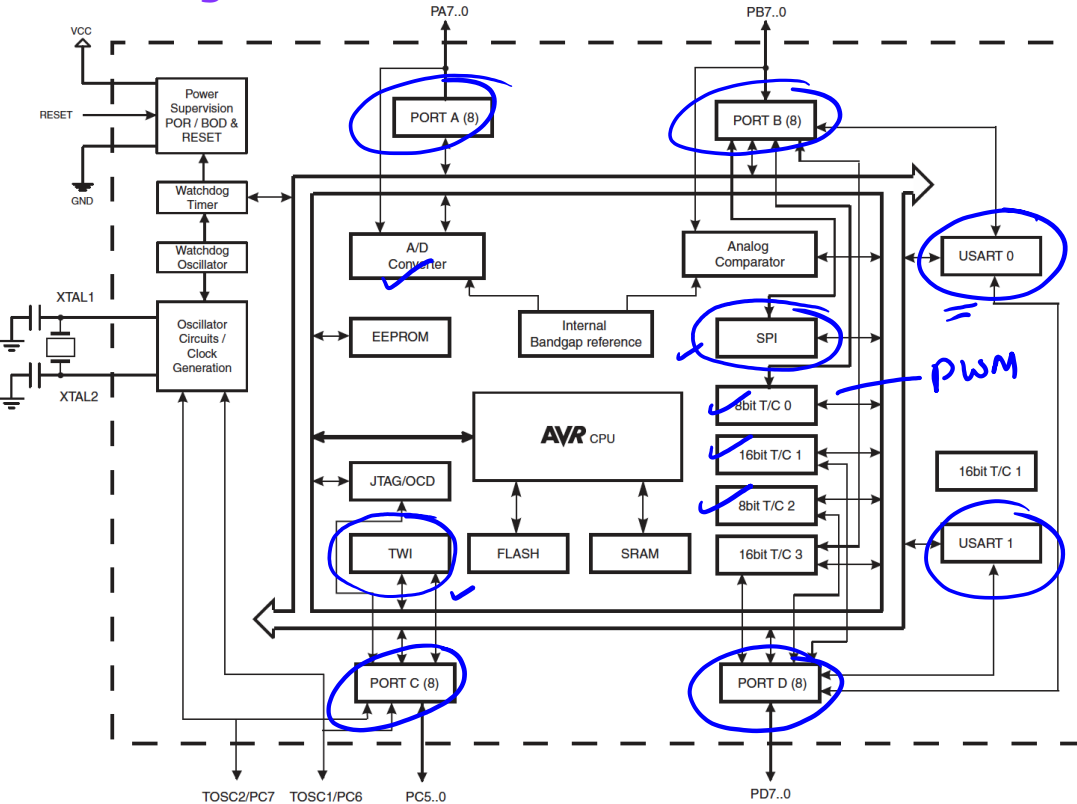
# Admin

- Quiz 8 is due this week Friday (08/10/21)  
4PM AEST
- Lab 15 (AVR interrupts) has a  
preparation task

# AVR – What have we looked at so far

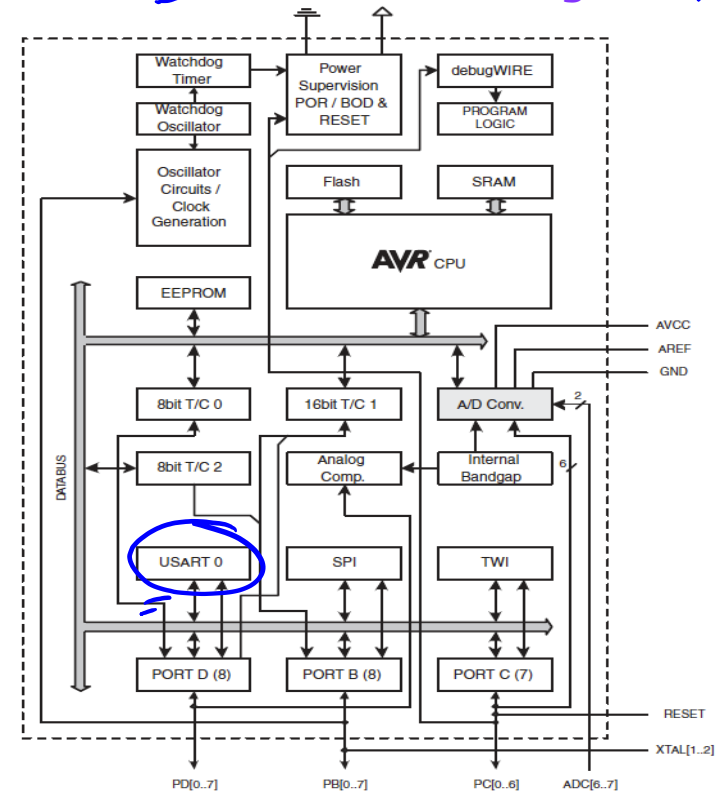
## ATmega324A

IN



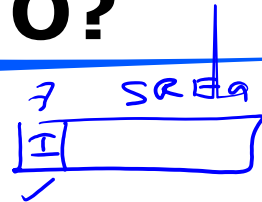
EX

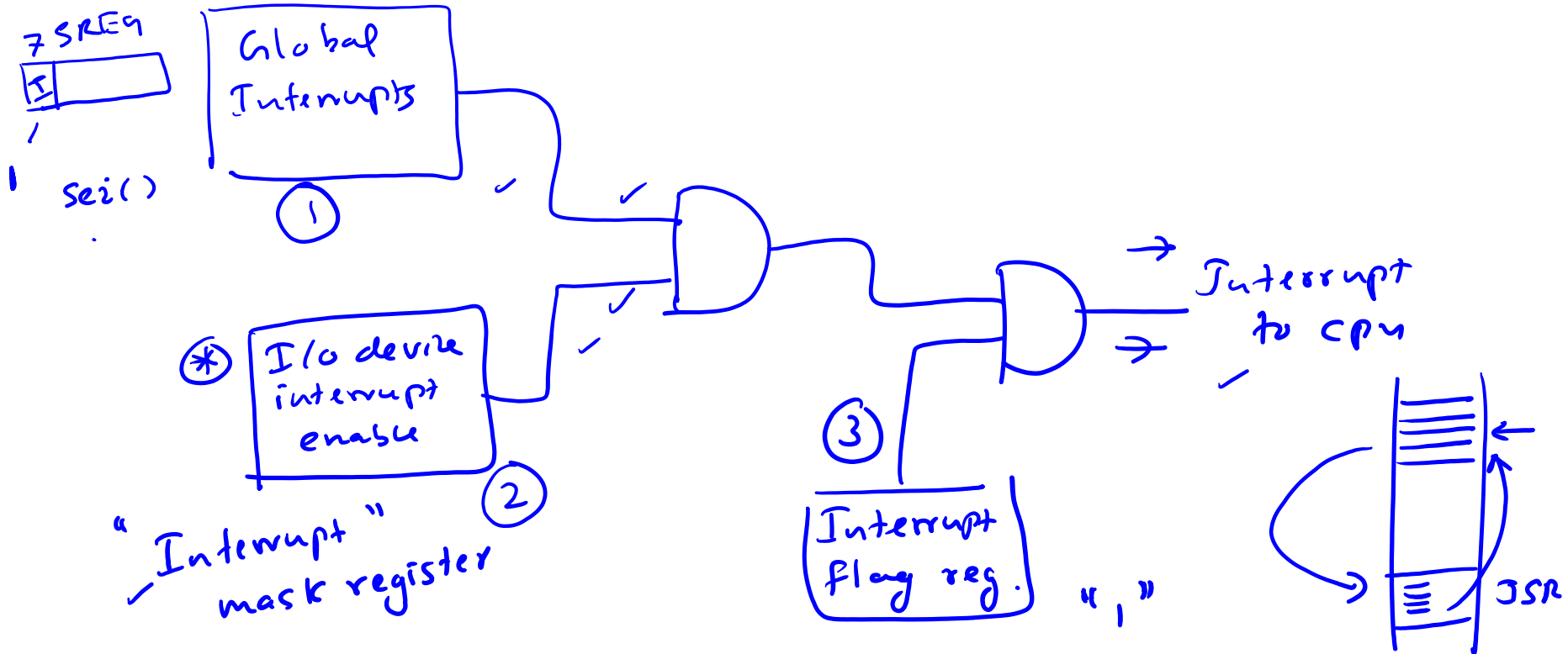
## ATmega328P



# Which of the following is FALSE when configuring interrupt driven I/O?

- A. ✓ Global interrupts should be enabled - T
- B. ✓ SREG bit 7 should be set to 1 - T
- C. ✓ Interrupt should be enabled in the I/O device - T
- D. ✓ '1' should be written to the particular interrupt flag register bit - T
- E. ✓ '0' should be written to the particular interrupt flag register bit

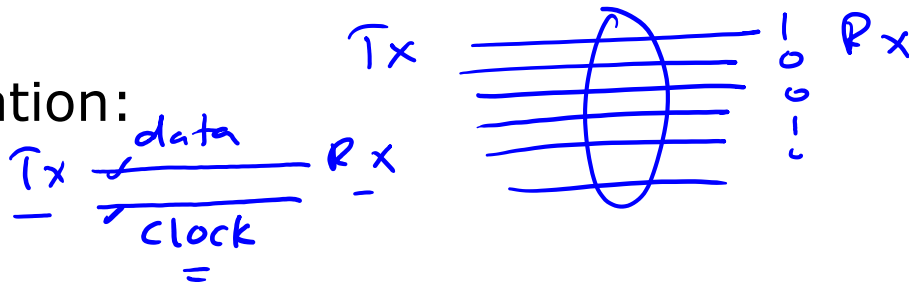




# Serial Communications

- Communications can be serial or parallel
- AVRs and many other devices support serial communication
  - Communicate one bit at a time
- Two types of serial communication:
  - Synchronous Transfer, e.g.
    - SPI - Serial Peripheral Interface
  - Asynchronous Transfer, e.g.
    - UART - Universal Asynchronous Receiver and Transmitter
    - AVR ATmega324A/328P has USART - Universal Synchronous/Asynchronous Receiver and Transmitter
      - ✓ ATmega324A (IN students) - USART0 and USART1
      - ✓ ATmega328P (EX students) - USART0
    - USB is another example

Tx 1010 → Rx



Tx data Rx

# Synchronous vs Asynchronous

## ● Synchronous

- Clock signal is transmitted with data

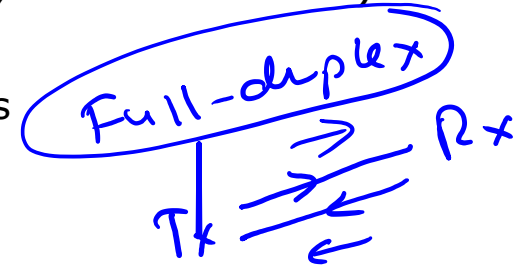
## ● Asynchronous

- Clock signal must be recovered from data stream
  - Data stream must be encoded to allow this to happen, e.g.
    - ✓ Start and stop bits ←
    - ✓ 4B/5B
    - ✓ 8B/10B
    - Bit stuffing (used in USB low speed and full-speed)
    - etc

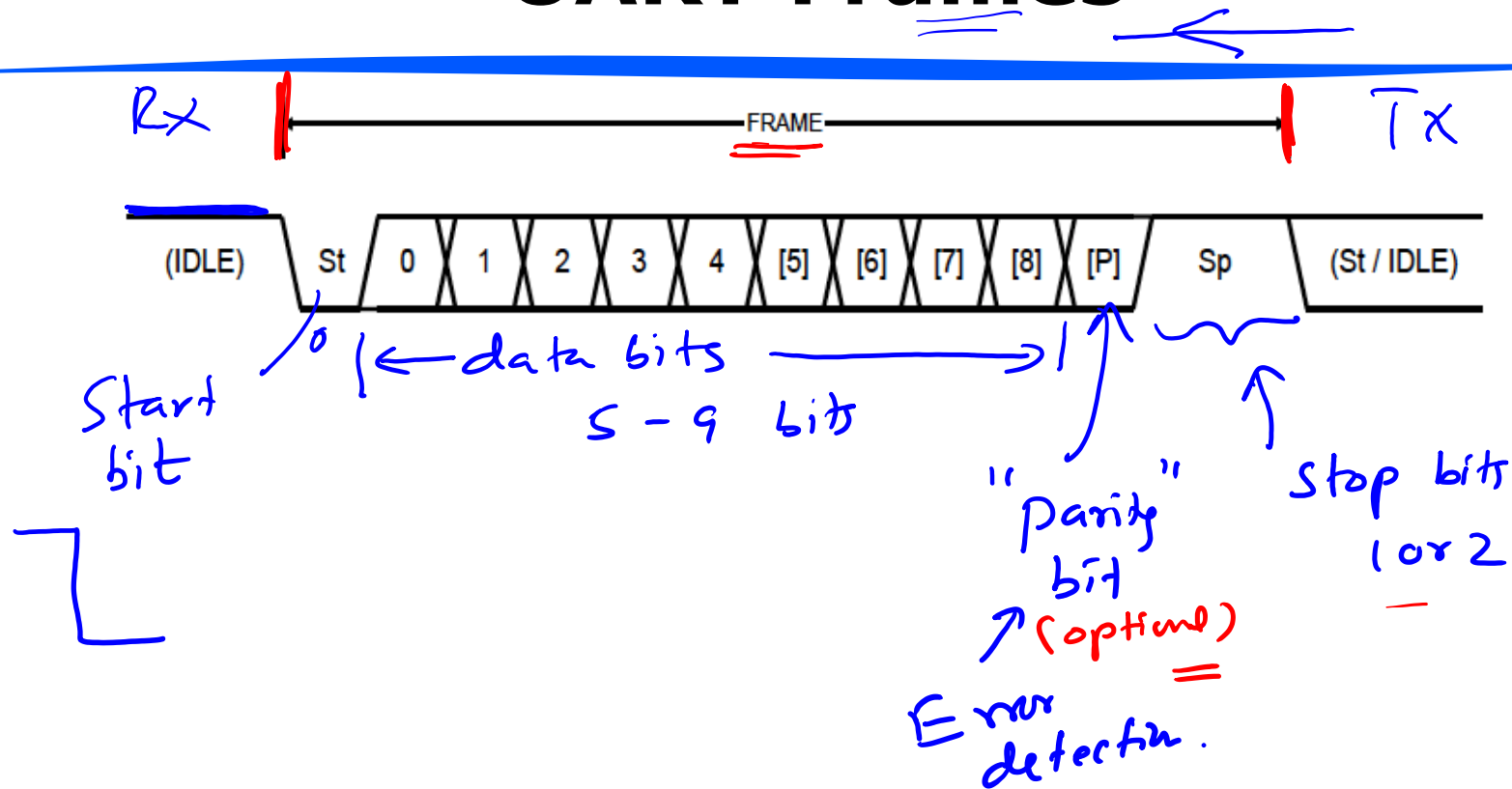


# AVR USART

- **USART** = **U**niversal **S**ynchronous and **A**synchronous **R**eceiver/**T**ransmitter
- Key points for us
  - **Serial** communication (we communicate *frames*, 1 bit at a time)
  - **Asynchronous**
    - Receiver and transmitter can have different clock rates
    - Clock not transmitted with data
    - Uses start and stop bits
- Often see term **UART** used
- Datasheet pages for UART
  - ATmega324A (IN students) – pages 175 to 201
  - ATmega328P (EX students) – pages 179 to 204



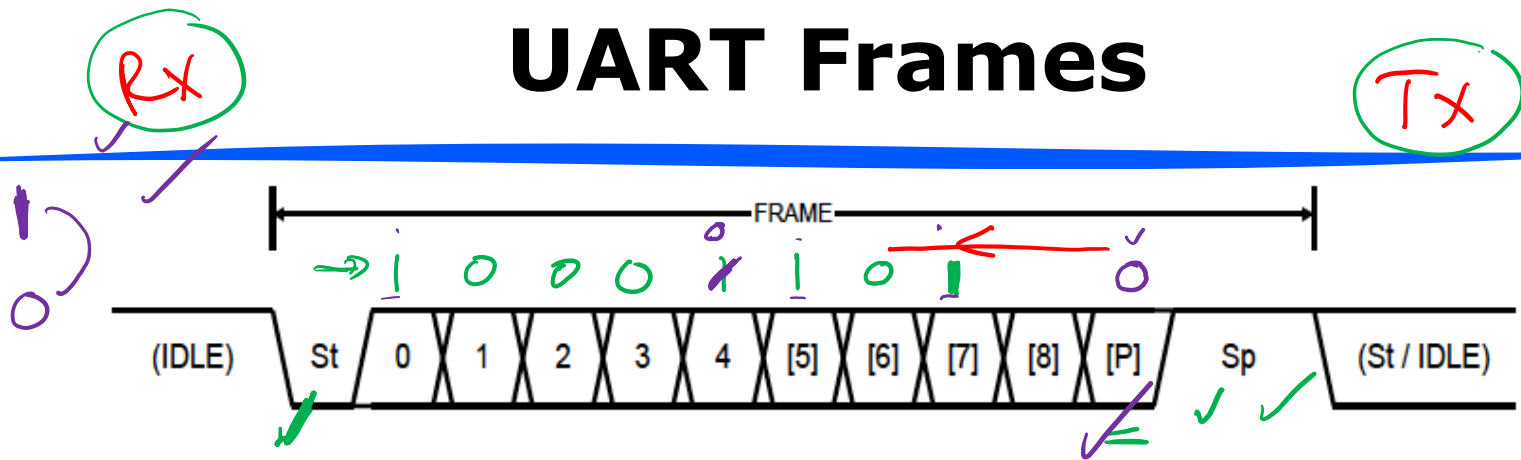
# UART Frames



[From page 180 of ATmega324A datasheet]

[From page 184 ATmega328P datasheet]

# UART Frames



❑ Parity bit is used for error detection

❑ Even parity – the number of 1's (in the data bits) including the parity bit must be even

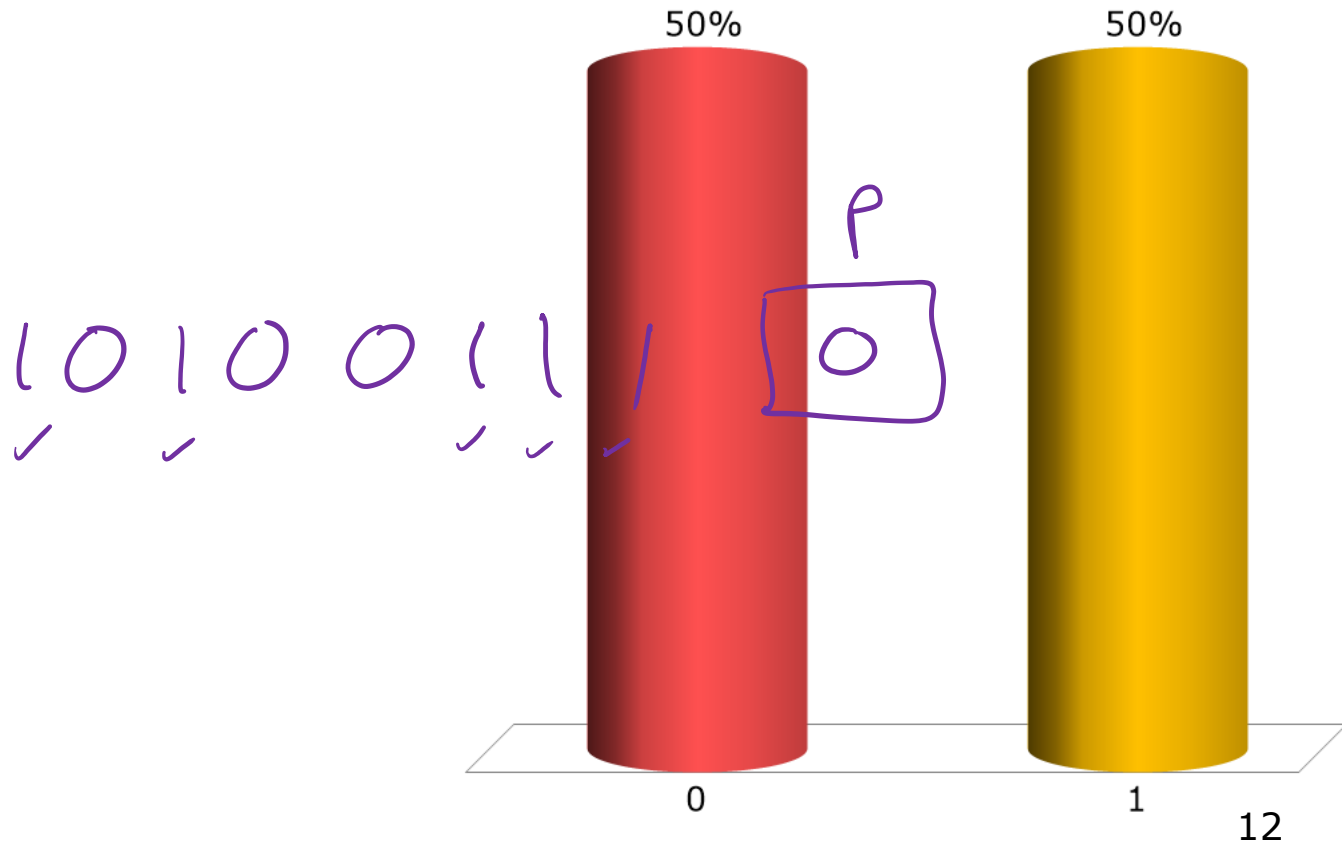
❑ Odd parity – the number of 1's (in the data bits) including the parity bit must be odd

❑ Parity bit is created at the Tx (transmitter) and parity check is performed at the Rx (receiver)

If 8 data bits 0xA7 to be transmitted with odd parity, what is the parity bit?

A. 0 ✓✓

B. 1

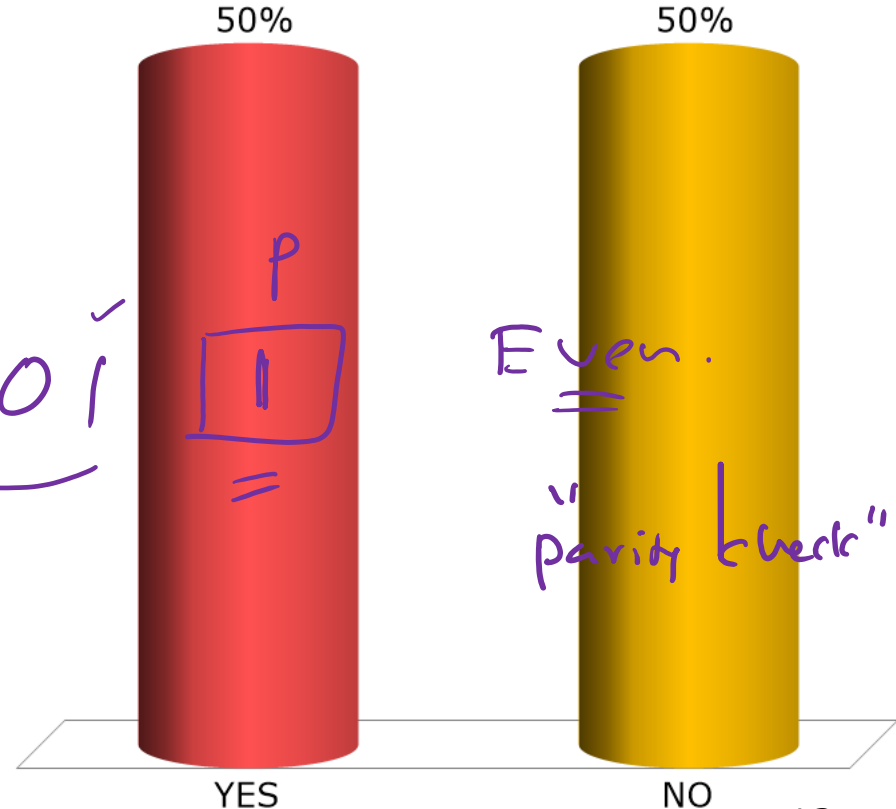


If 8 data bits 0xA9 have been received with parity bit 1 under even parity, is there any bit errors?

A. YES

B. NO

1010 1001  
data  
0

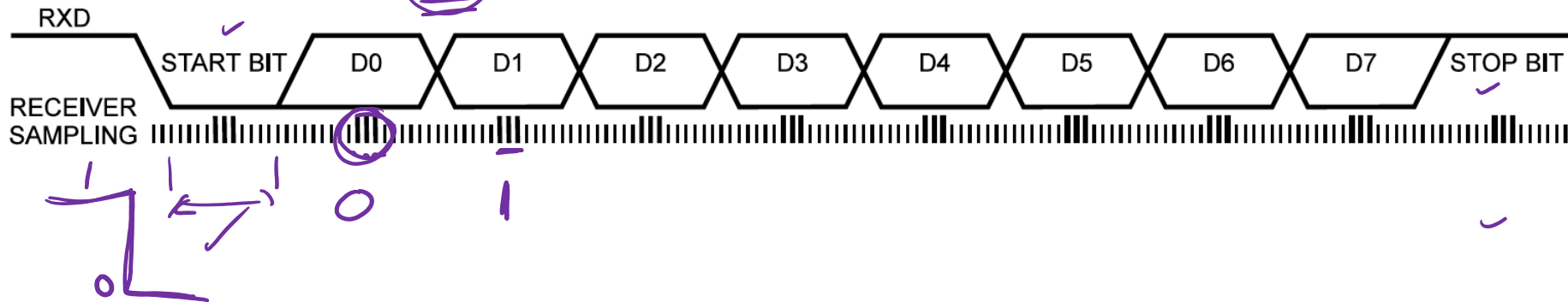


# Recovering the data

- Receiver must know baud rate
  - but clocks don't have to be synchronised
- Start bit (1→0 transition) triggers sampling
- Multiple samples per bit
- AVR uses 16 (or 8 on double speed mode)

✓ Stop bits  
 ✓ data  
 ✓ parity  $< E$   
 ✓ Baud rate ←

9600 baud/sec  
 TX



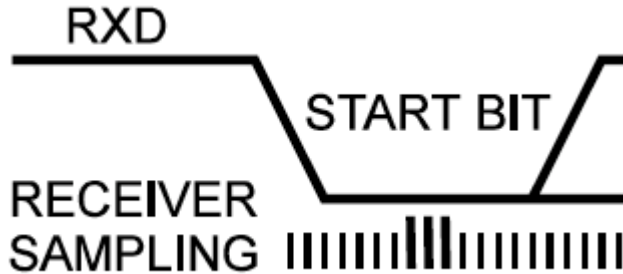
# Baud Rate

- Baud rate = symbols per second
- Not the same as bit rate
  - e.g. frames of 8-bits, no parity, one start bit, one stop bit
    - 10 symbols per 8 bits
    - 9600 baud = 7680 bits per second  
= 960 bytes per second
- Number of industry standard baud rates:
  - 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200

$$\rightarrow 9600 \times \frac{8}{10} \text{ bits/sec}$$

# AVR Baud Rate

- Baud rate register is set based on
  - Device clock speed (8MHz for IN and 16MHz for EX)
  - Number of clock cycles between samples – 16 samples per symbol
  - Can calculate – or see datasheet tables
    - Pages 198 to 201 of ATmega324A or pages 196 to 199 of ATmega328P



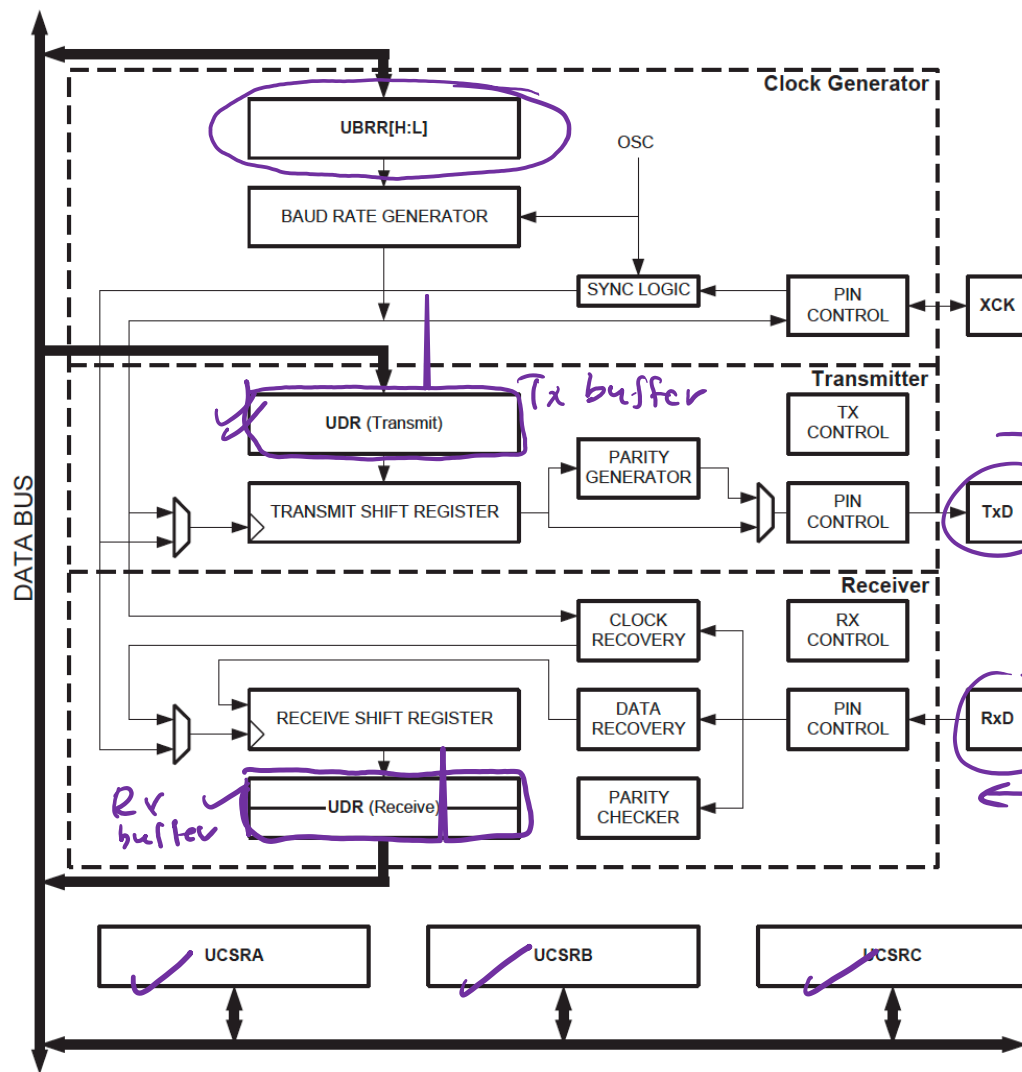
$$\underline{UBRR_n} = \frac{f_{OSC}}{\underline{16BAUD}} - 1$$

Handwritten notes:  $8\text{ MHz} - \text{IN}$  and  $16\text{ MHz} - \text{EX}$  with arrows pointing to the formula. A handwritten  $9600$  is written below the denominator with an arrow pointing to it. There are also checkmarks next to the formula and the handwritten values.

[From page 178 of ATmega324A datasheet]  
[From page 182 ATmega328P datasheet ]

We will mainly use the "Asynchronous Normal Mode" of the USART.





**ATmega324A:**

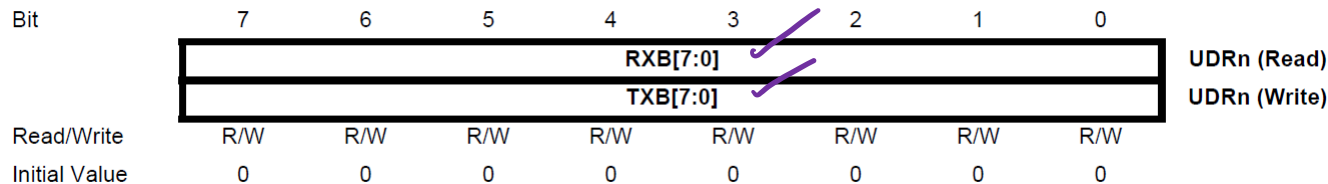
**USART0: PD0 (RXD0)  
and PD1(TXD0)**

**ATmega328P:**

**USART0: PD0 (RXD0)  
and PD1(TXD0)**

# Interacting with AVR UART0 – Key I/O Registers

- USART Data Register UDR0
  - Actually two registers - one for reading, one for writing
  - Data written to this register is transmitted
  - Data arriving over the serial port can be read from this register



[From page 193 of ATmega324A datasheet]

[From page 200 ATmega328P datasheet ]

# Interacting with AVR UART0 – Key I/O Registers

- USART Control and Status Registers (UCSR0A, UCSR0B, UCSR0C)

Bit	7	6	5	4	3	2	1	0	
	<b>RXCn</b>	<b>TXCn</b>	<b>UDREN</b>	<b>FEn</b>	<b>DORn</b>	<b>UPEn</b>	<b>U2Xn</b>	<b>MPCMn</b>	<b>UCSRnA</b>
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	<b>RXCIE<sub>n</sub></b>	<b>TXCIE<sub>n</sub></b>	<b>UDRIE<sub>n</sub></b>	<b>RXEN<sub>n</sub></b>	<b>TXEN<sub>n</sub></b>	<b>UCSZ<sub>n2</sub></b>	<b>RXB8<sub>n</sub></b>	<b>TXB8<sub>n</sub></b>	<b>UCSRnB</b>
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	

Bit	7	6	5	4	3	2	1	0	
	<b>UMSEL<sub>n1</sub></b>	<b>UMSEL<sub>n0</sub></b>	<b>UPM<sub>n1</sub></b>	<b>UPM<sub>n0</sub></b>	<b>USBS<sub>n</sub></b>	<b>UCSZ<sub>n1</sub></b>	<b>UCSZ<sub>n0</sub></b>	<b>UCPOL<sub>n</sub></b>	<b>UCSRnC</b>
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	1	1	0	

[From page 193-196 of ATmega324A datasheet] ✓

[From page 200-203 ATmega328P datasheet] ✓

# Interacting with AVR UART0 – Key I/O Registers

- USART Baud rate registers (UBRR0H, UBRR0L)
- Either compute the Baud rate register value using the formula given or use the common example values given in the datasheet

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRnH
	UBRR[7:0]								UBRRnL
	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	



[From page 197 of ATmega324A datasheet]

[From page 204 ATmega328P datasheet ]

# Serial Interrupts

- 3 interrupt sources associated with each AVR serial port

- ✓ ✓
  - Receive complete —

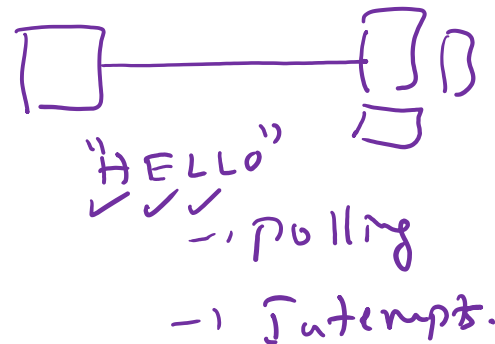
- Frame (usually character) received

- ✓
  - ✓
    - Transmit complete

- Frame sent and no data waiting to be sent

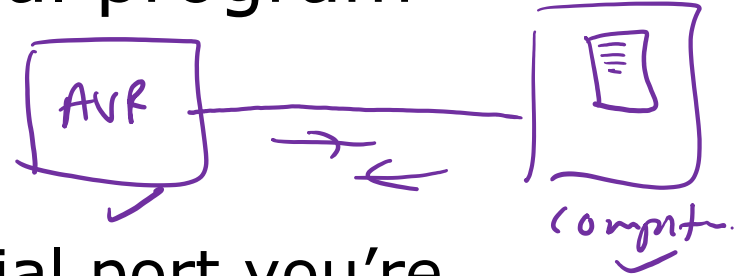
- ✓
  - ✓
    - Data Register empty

- Ready to accept new data for transmission



# Serial communications on the PC

- IO Board provides a USB-serial device
- On PC we use a terminal program
  - ✓ ■ e.g. Putty
- Need to specify
  - COM port (which PC serial port you're communicating over)
    - May need to use device manager to determine this
  - ✓ ■ Baud rate, start/stop/parity bits etc



# RS232

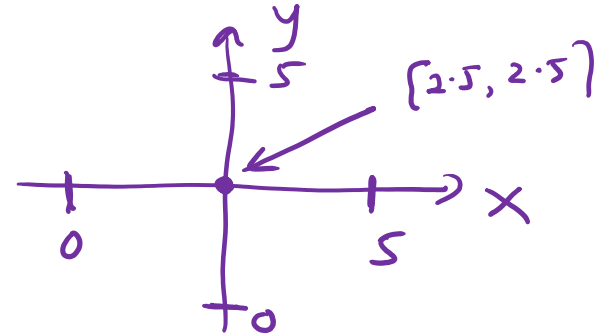
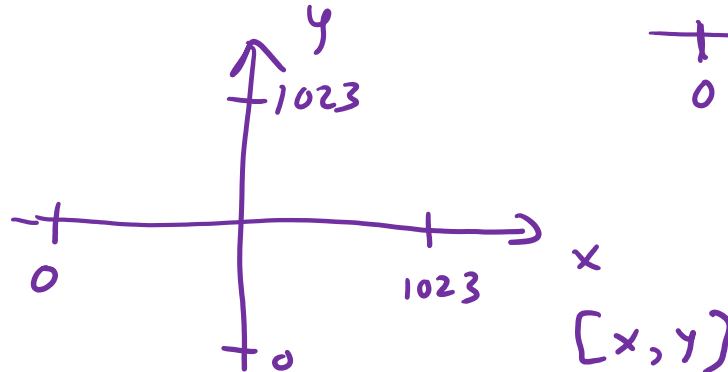
- Serial Communication Standard
  - Standardised over 40 years ago, still in use
  - Typically uses a 9-pin D-sub connector
- Voltage levels
  - Logical one = Negative voltage (-3 to -15V)
  - Logical zero = Positive voltage (+3 to +15V)
- Full duplex (can transmit and receive simultaneously)
- Various subsets of signals possible
  - Need at least
    - Transmit Data (TxD)
    - Receive Data (RxD)
    - Ground
- Can turn our 0-5V serial into RS232 using a converter chip (e.g. MAX-232)



# Analog to Digital Conversion (ADC)

- Voltage between 0 and some maximum voltage ( $V_{ref}$ ) converted to digital value
- ADC has some resolution
  - e.g. 10 bits (range 0 to 1023)

0  
⋮  
1023





# ATmega324A/328P ADC

- Port A pins on ATmega324A or Port C pins on ATmega328P can be used as analog inputs
- One input can be converted at a time (using an analog multiplexer)
- Can choose Vref - we'll pick Vcc = 5V
- ADC resolution = 10 bits (0 to 1023)
- Conversion takes time
  - Up to 3200 clock cycles (0.4ms at 8MHz) ✓
  - Can trade-off accuracy for speed

