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

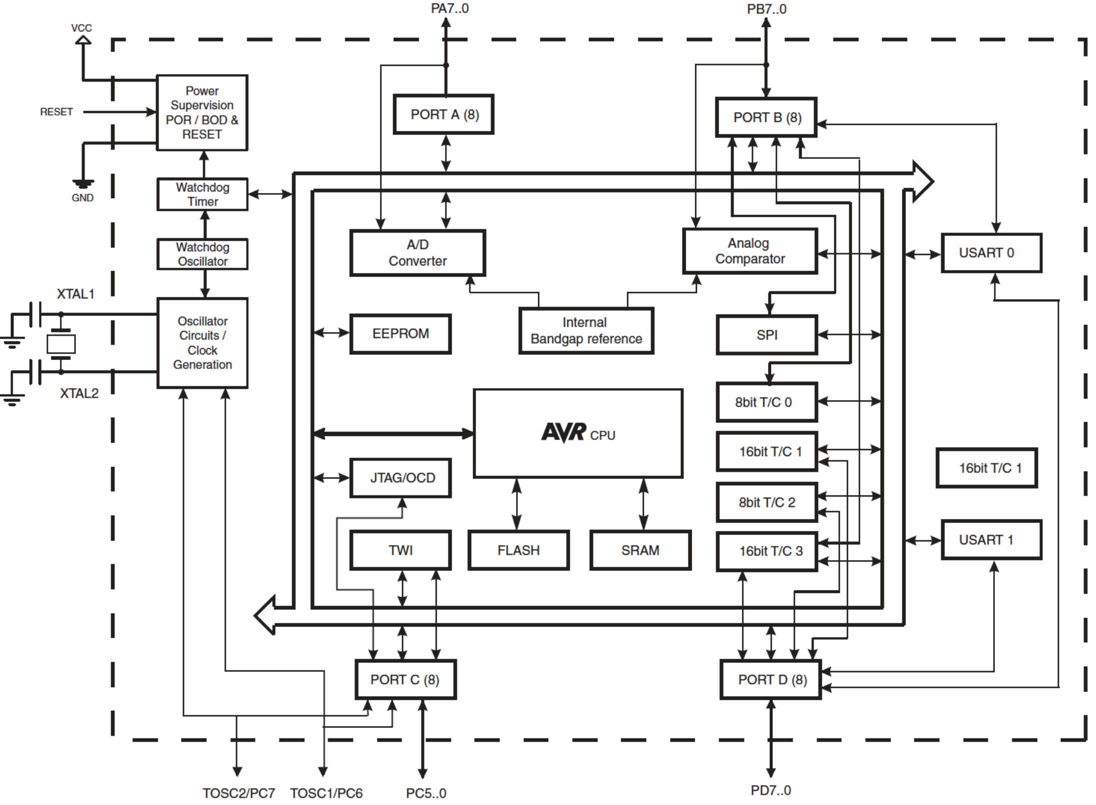
Admin

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

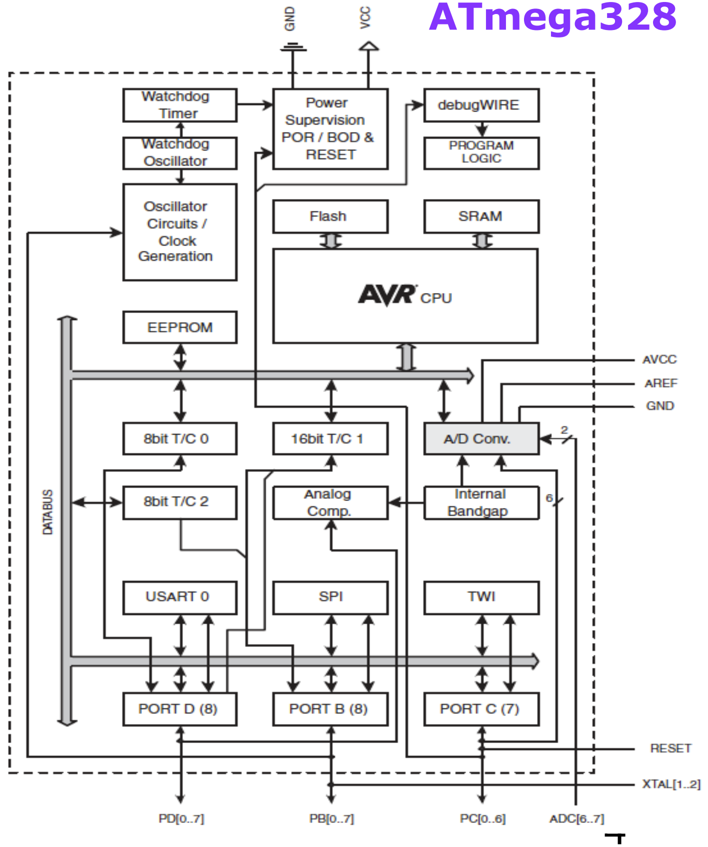
CSSE2010

AVR – What have we looked at so far

ATmega324A



ATmega328



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

A. Global interrupts should be enabled
B. SREG bit 7 should be set to 1
C. Interrupt should be enabled in the I/O device
D. '1' should be written to the particular interrupt flag register bit
E. '0' should be written to the particular interrupt flag register bit

CSSE2010

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

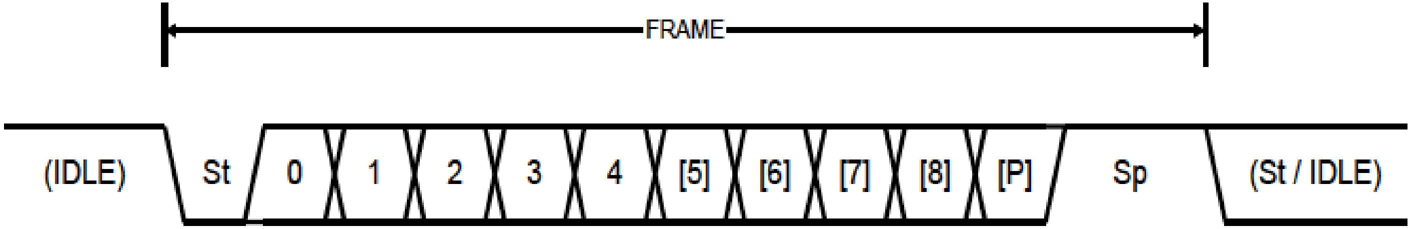
Synchronous vs Asynchronous

- 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

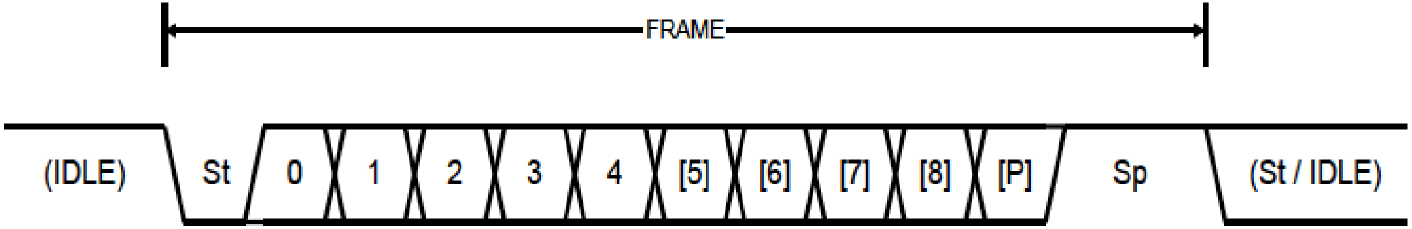
- **USART** = **U**niversal **S**ynchronous and **A**synchronous **R**eceiver/**T**ransmitter
- Key points for us
  - **Serial** communication (we communicate *frames*, 1 bit at a time)
  - **Asynchronous**
    - Receiver and transmitter can have different clock rates
    - Clock not transmitted with 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

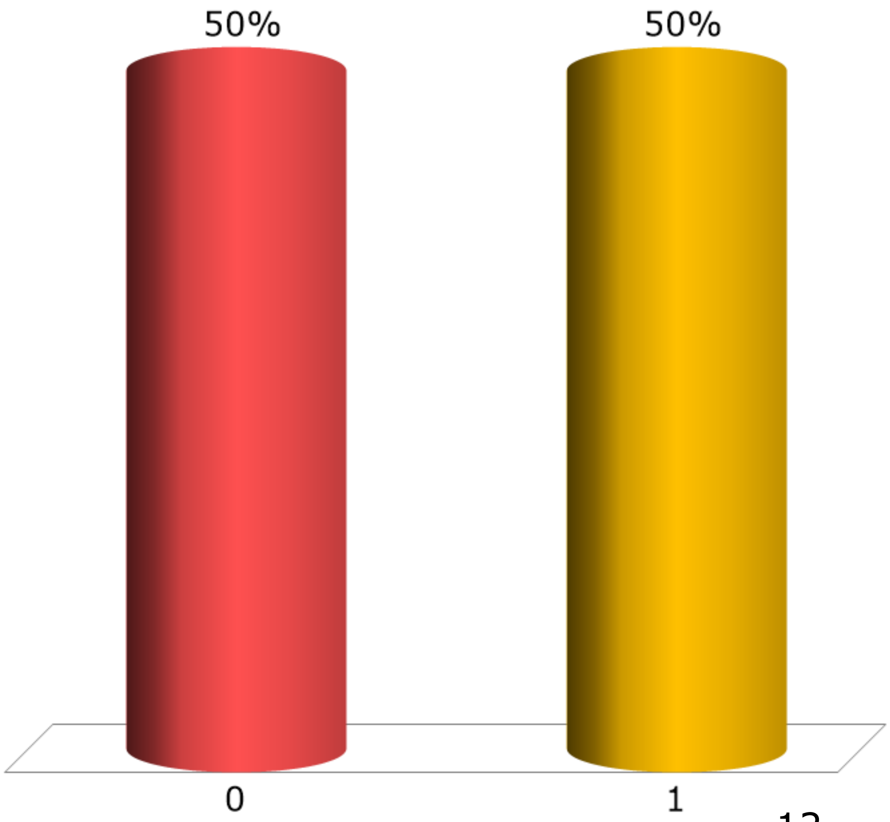


- Parity bit is used for error detection
- Even parity – the number of 1's (in the data bits) including the parity bit must be even
- Odd parity – the number of 1's (in the data bits) including the parity bit must be odd
- Parity bit is created at the Tx (transmitter) and parity check is performed at the Rx (receiver)

[From page 180 of ATmega324A datasheet]
[From page 184 ATmega328P datasheet]

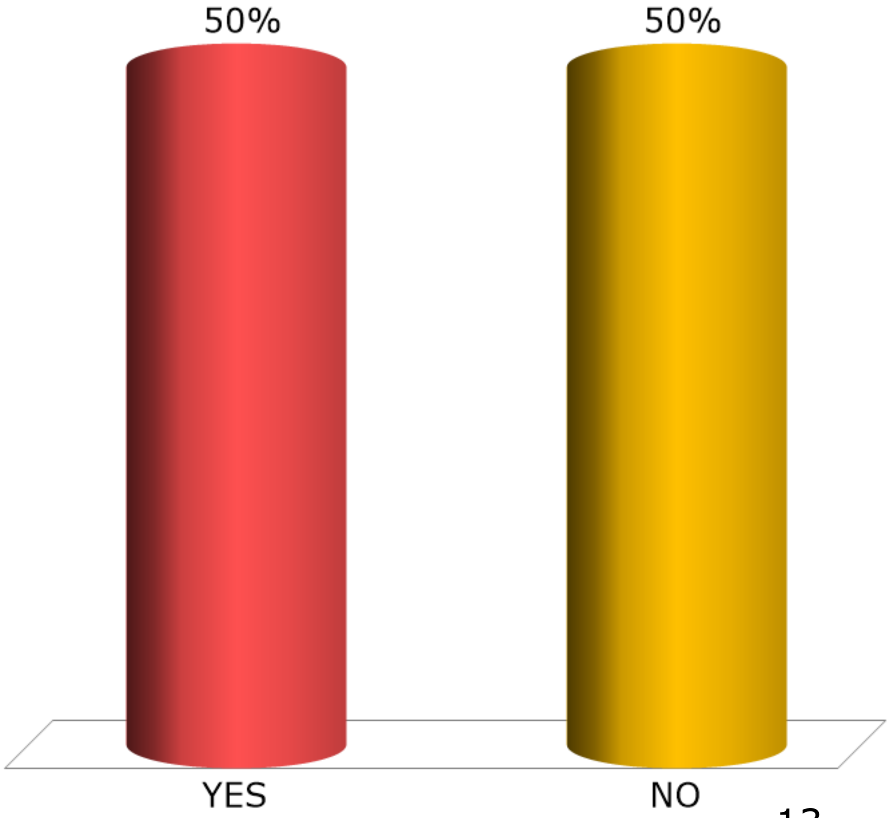
If 8 data bits 0xA7 to be transmitted with odd parity, what is the parity bit?

A. 0
B. 1



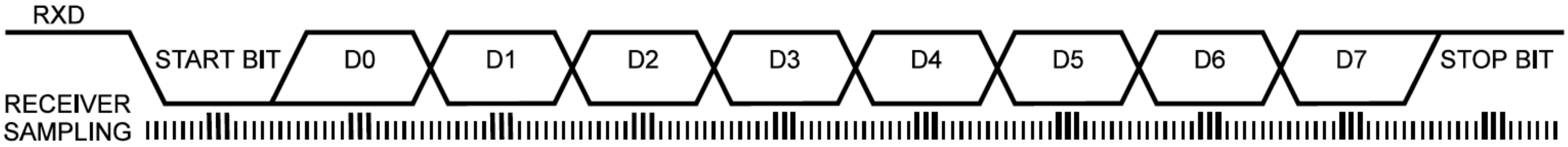
If 8 data bits 0xA9 have been received with parity bit 1 under even parity, is there any bit errors?

A. YES
B. NO



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)

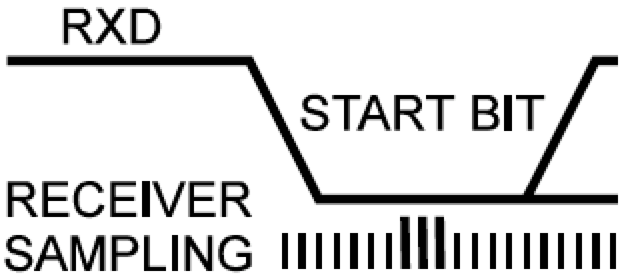


Baud Rate

- 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

AVR Baud Rate

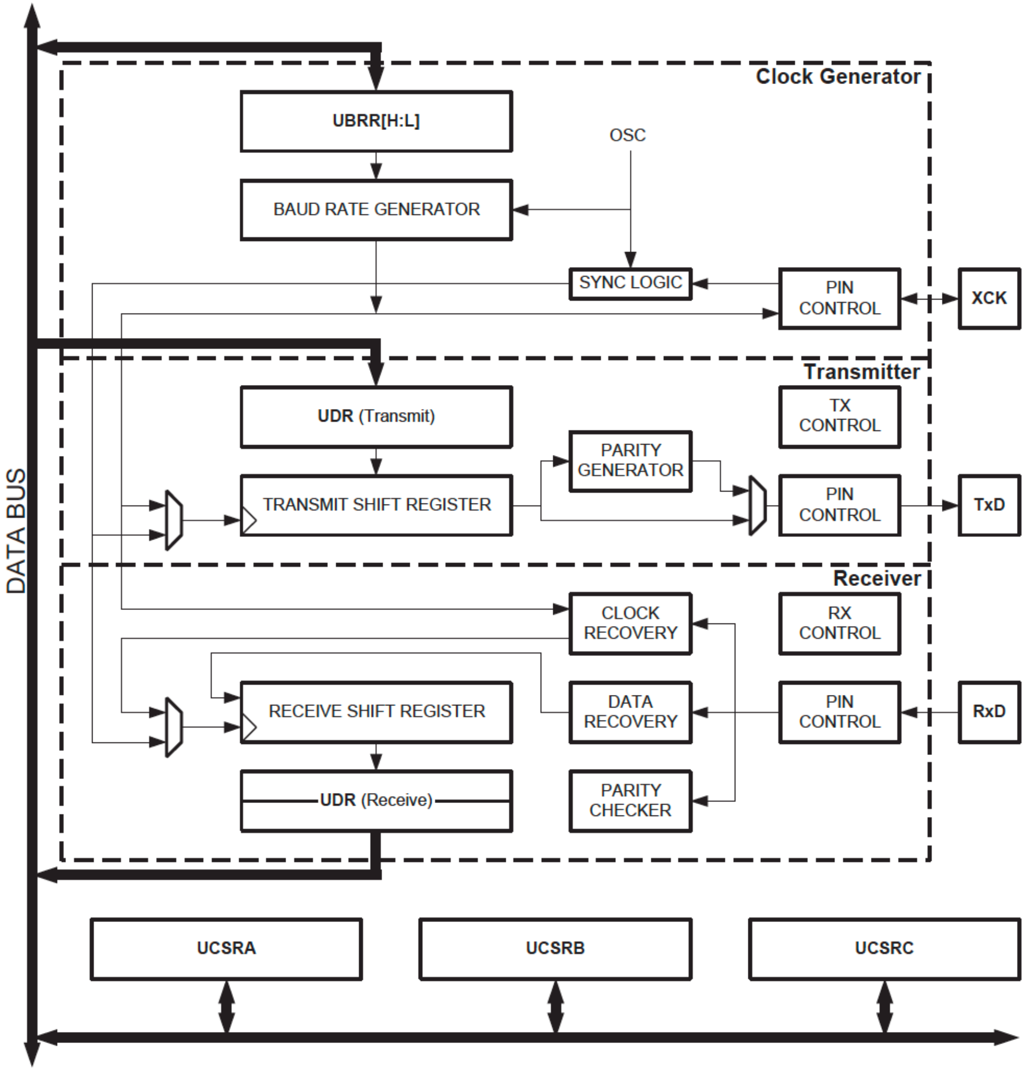
- Baud rate register is set based on
  - Device clock speed (8MHz for IN and 16MHz for EX)
  - Number of clock cycles between samples – 16 samples per symbol
  - Can calculate – or see datasheet tables
    - Pages 198 to 201 of ATmega324A or pages 196 to 199 of ATmega328P



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

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

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



ATmega324ADatasheet
Page 176

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

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
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	0	0	0	

[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	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
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	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
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	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
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 (Vref) converted to digital value
- ADC has some resolution
  - e.g. 10 bits (range 0 to 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