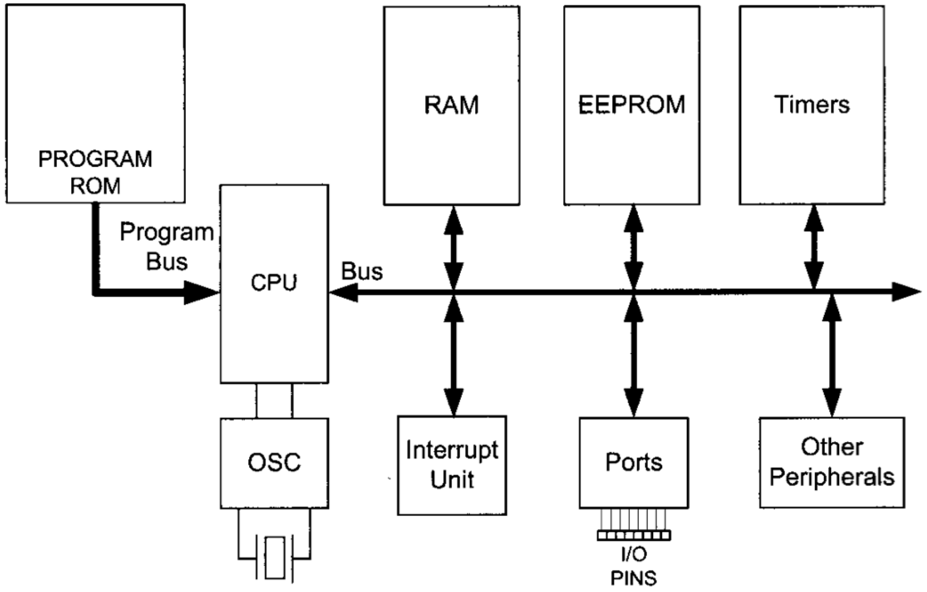
Outline

- Parallel vs Serial Communication
- LCD Interfacing (HD44780)
- UART Protocol
- AVR USART Programming

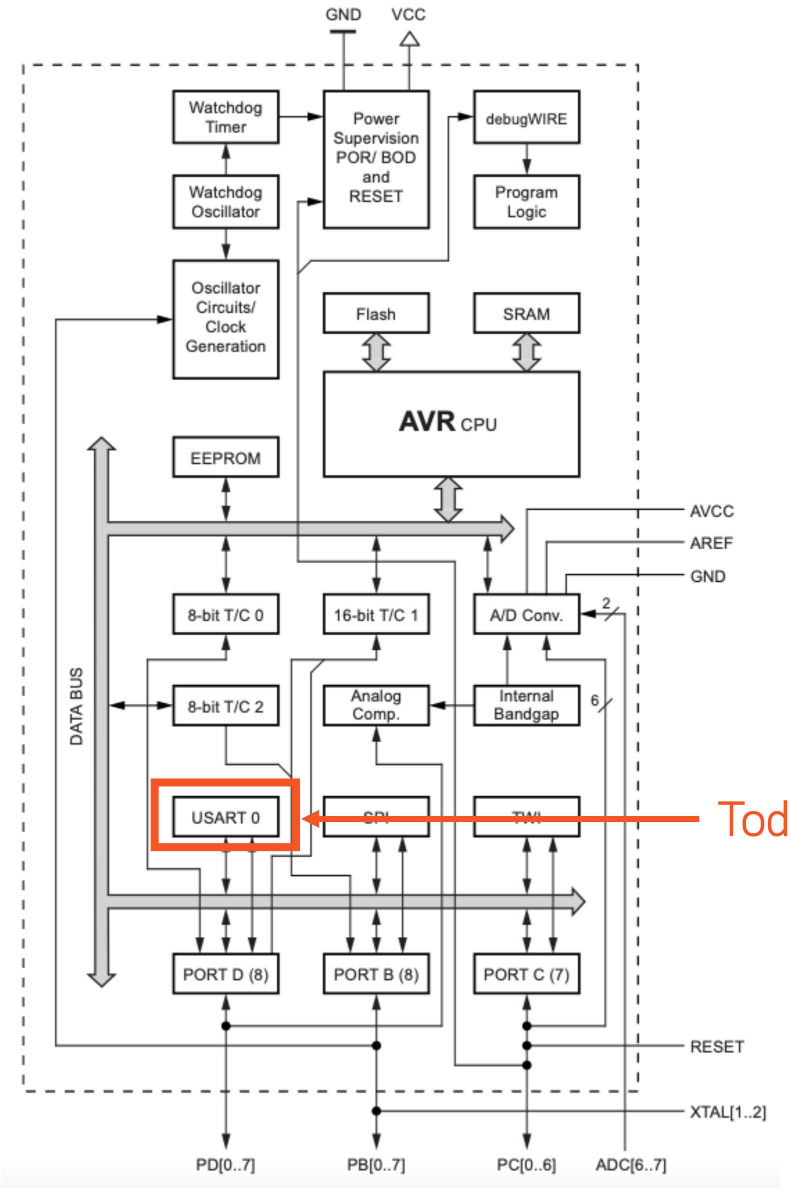
AVR USART

(Universal Synchronous and Asynchronous serial Receiver and Transmitter)

Recall AVR Architecture



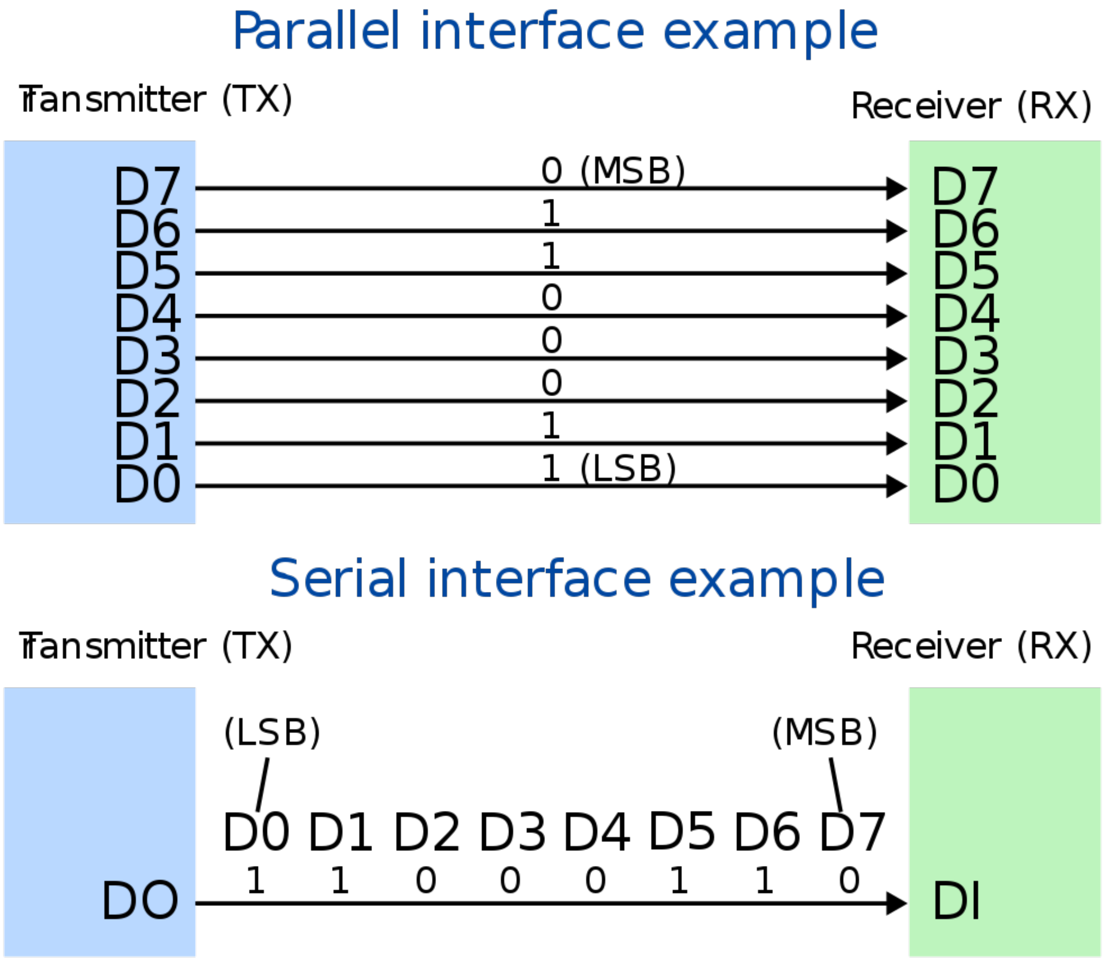
Simplified AVR Architecture



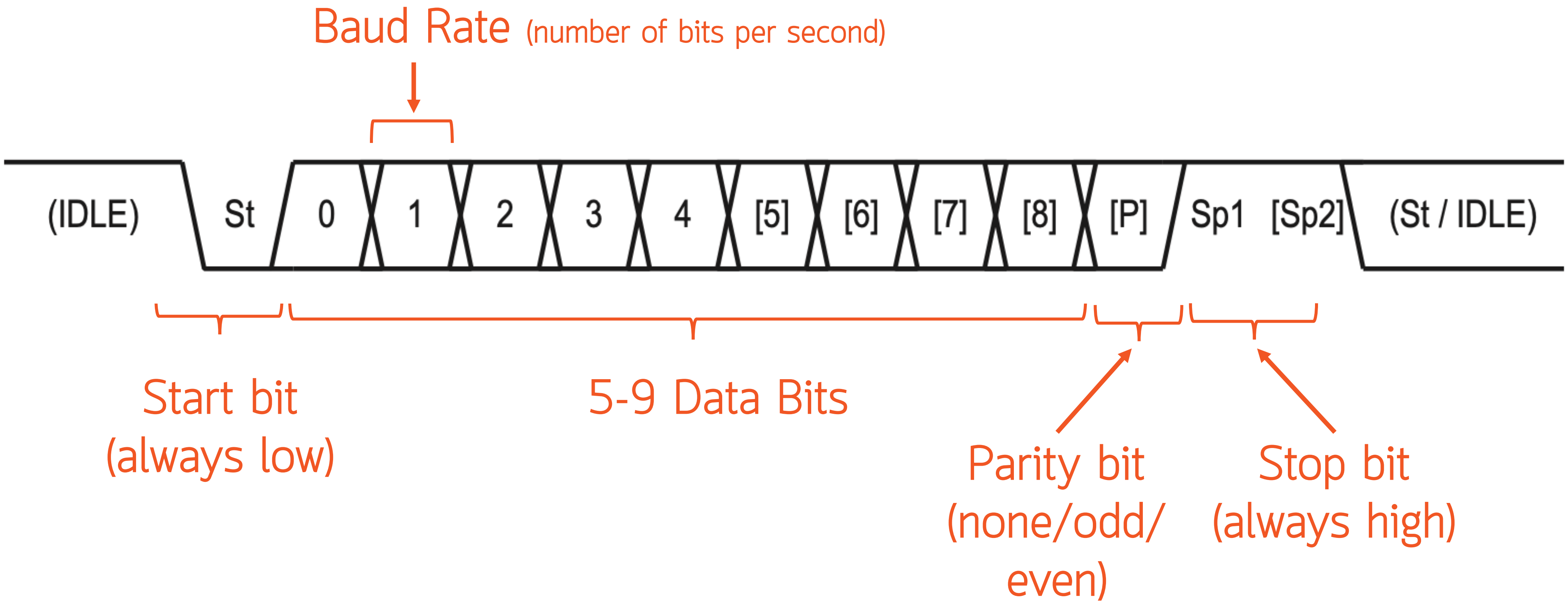
Today's Topic

ATMega328P Architecture

Serial vs Parallel Communication



UART Frame Format



Baud Rate (number of bits per second)

Start bit (always low)

5-9 Data Bits

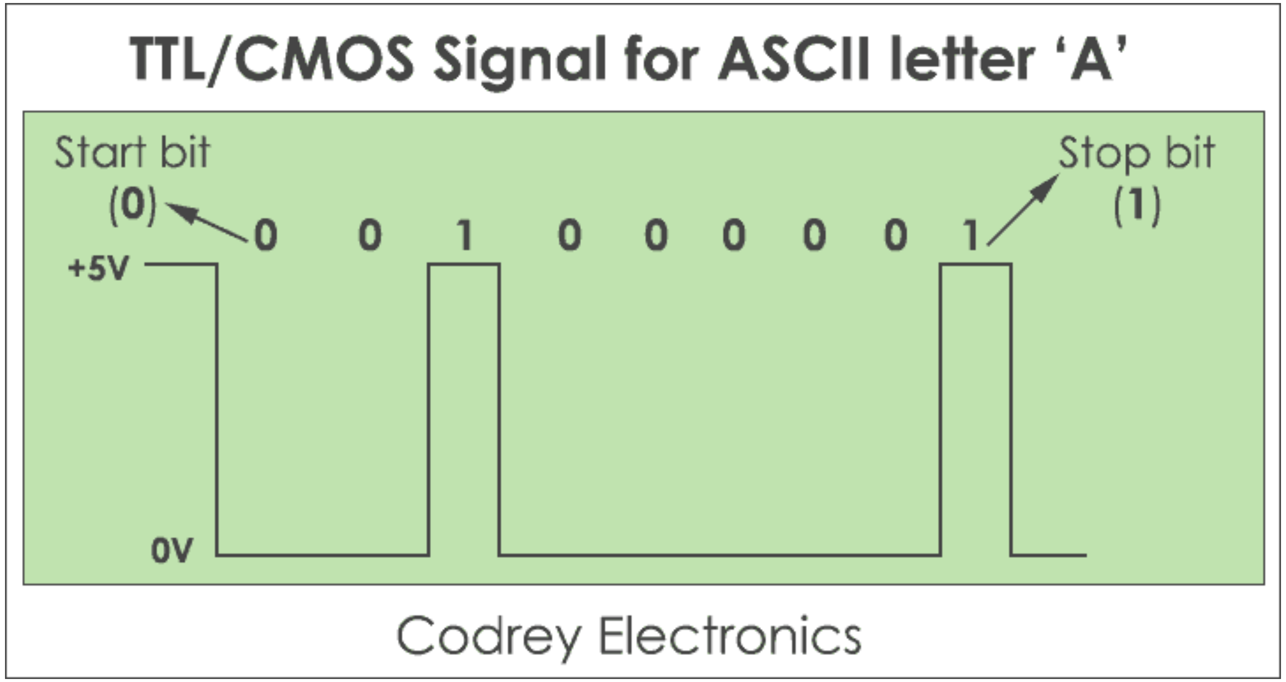
Parity bit (none/odd/ even)

Stop bit (always high)

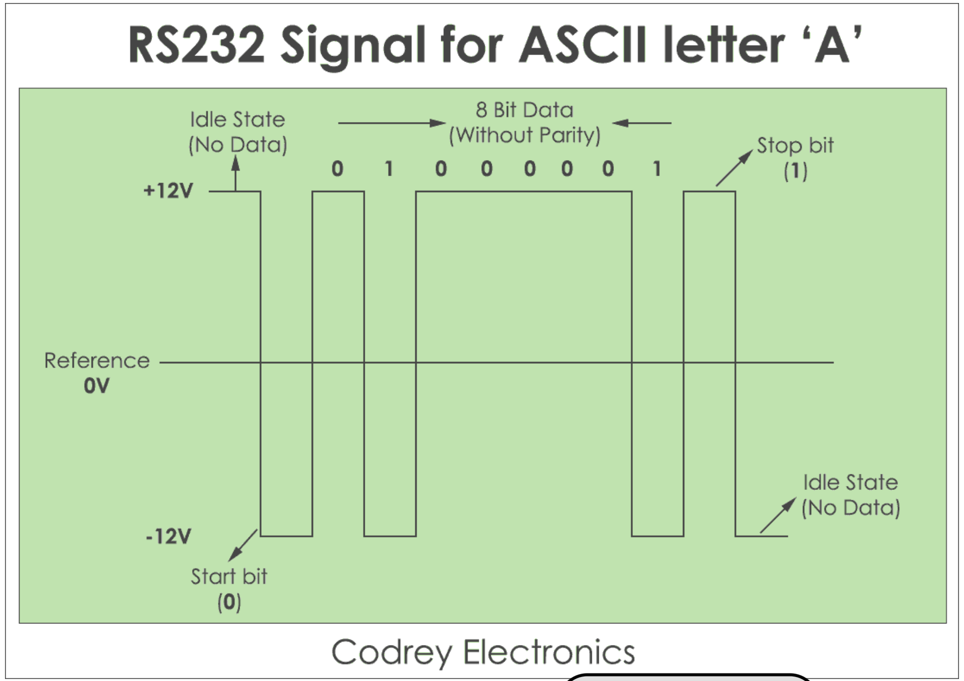
$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

UART Logic Level (TTL vs RS-232)

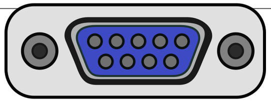
Most MCUs are operated at 3.3-5V and communicate using TTL/CMOS logic level



MCUs and single board computer e.g. Raspberry Pi

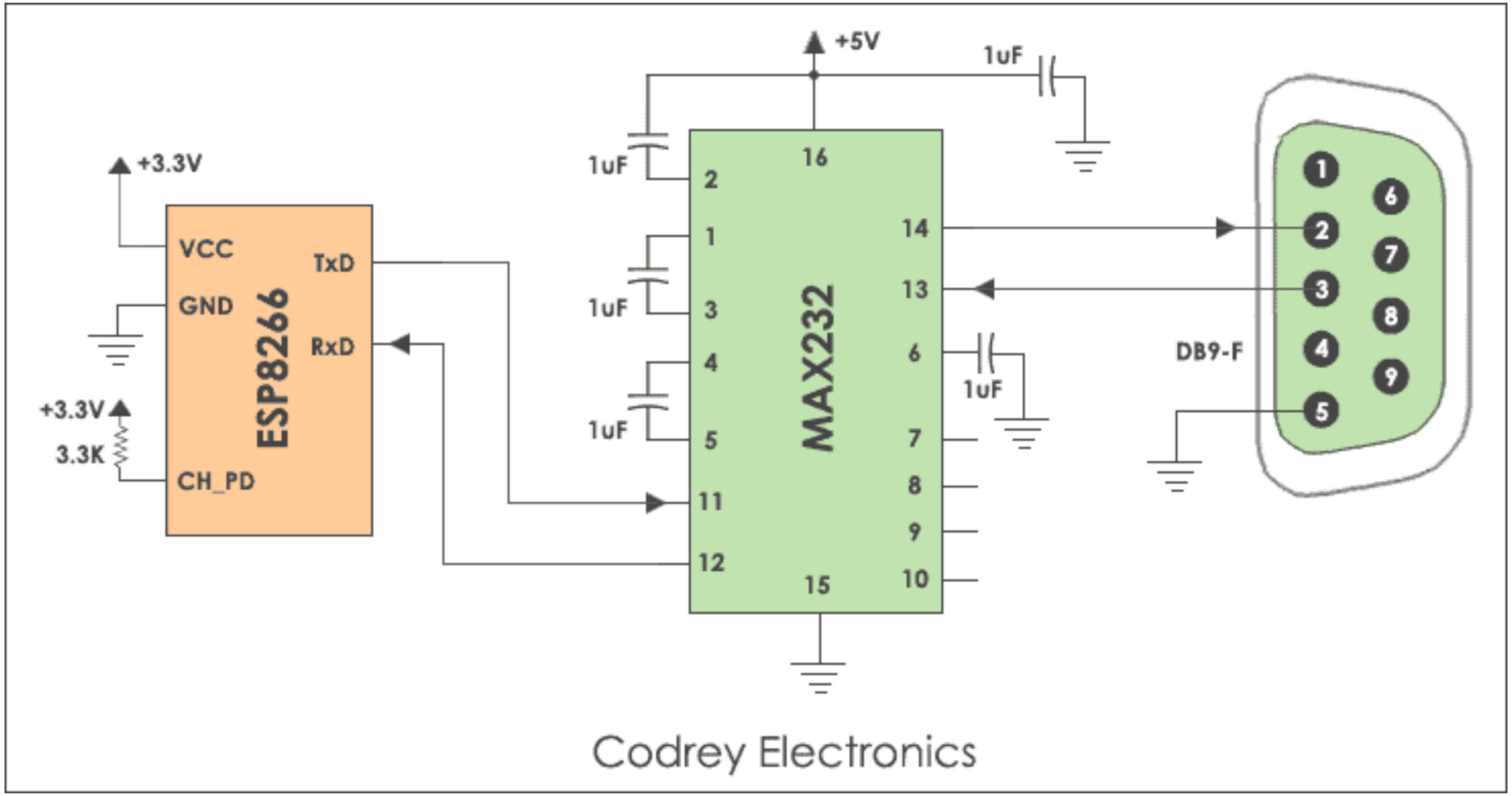


PC's serial port



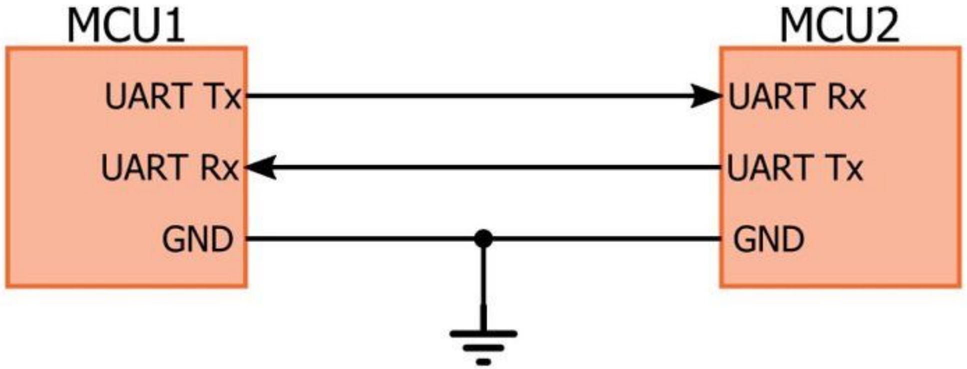
Source: https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/

Example of RS-232<->TTL Converter Circuit



Source: https://www.codrey.com/embedded-systems/uart-serial-communication-rs232/

UART Communication: Physical Connection

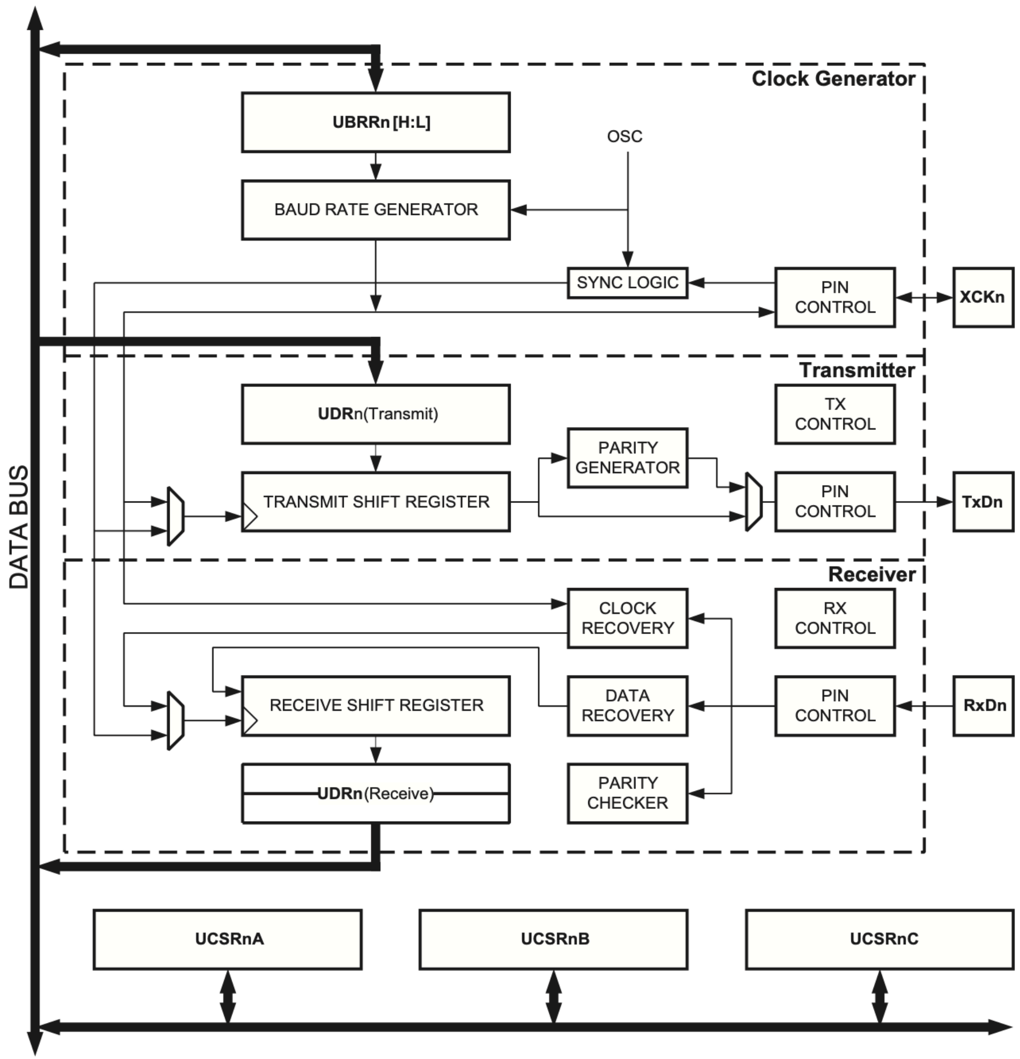


- UART protocol uses 3 wires (TX, RX, GND) for full duplex transmission
- AVR and most MCUs support synchronous operation with additional clock signal thus the module name is USART or Universal Synchronous and Asynchronous serial Receiver and Transmitter

AVR USART: General Features

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous (Master or Slave Clocked) Operation
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete

AVR USART: Block diagram



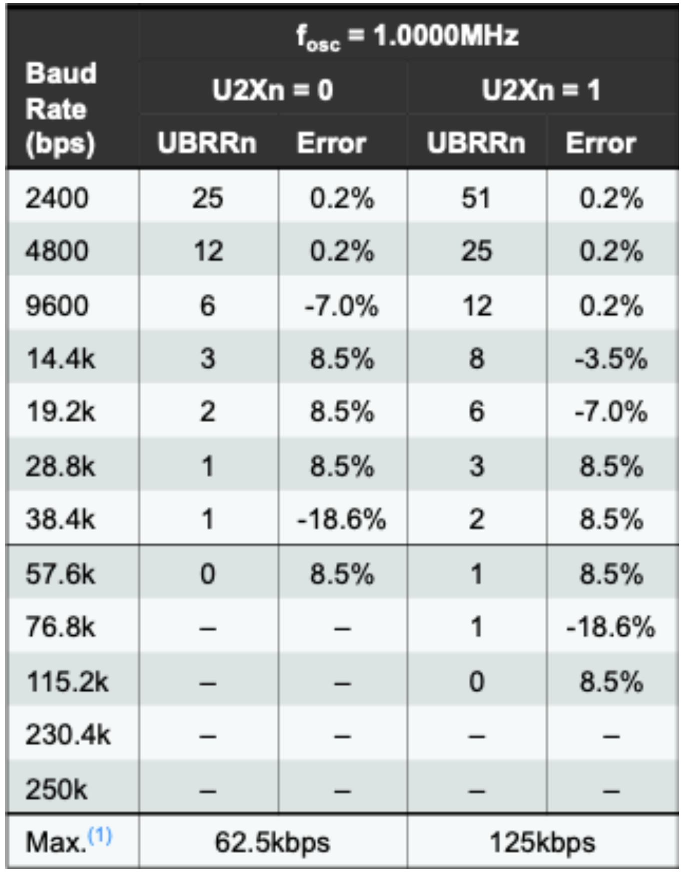
- AVR USART module consists of 3 main parts: Clock Generator, Transmitter and Receiver
- The module connects to 3 I/O pins: XCKn, TxDn, RxDn
- The module uses 6 registers in the I/O memory
  - UBRRn
  - UDRn (Read/Write)
  - UCSRnA/B/C

AVR USART: Programming (Initialization)

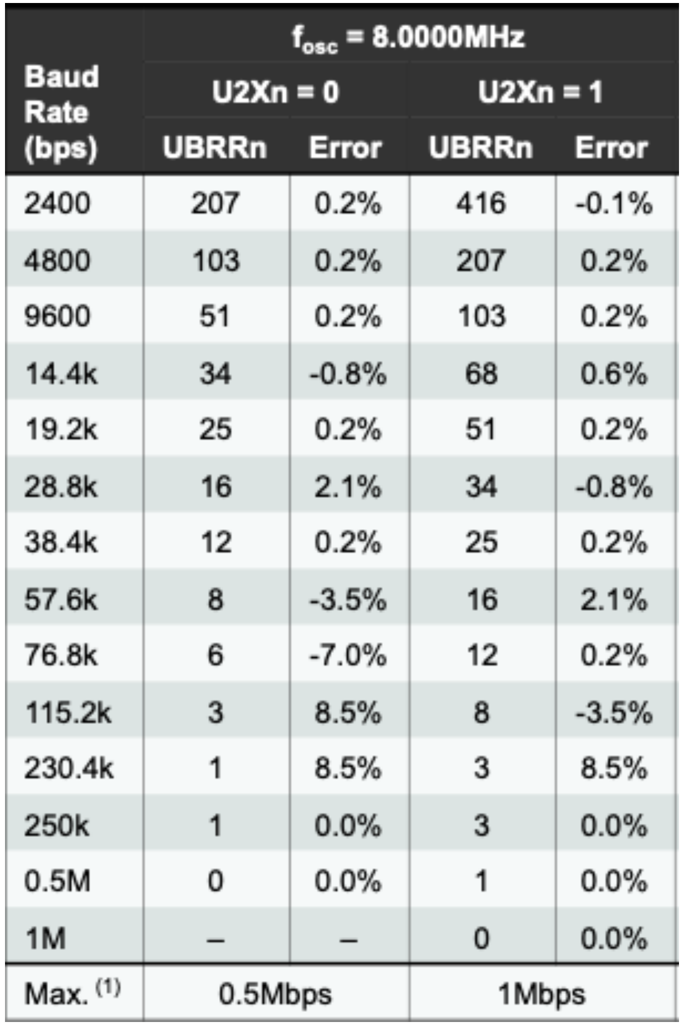
```c
void USART_Init(unsigned int ubrr) {

    /* Set baud rate */
    UBRR0H = (unsigned char)(ubrr>>8);
    UBRR0L = (unsigned char)ubrr;

    /* Enable receiver and transmitter */
    UCSR0B = (1<<RXEN0)|(1<<TXEN0);

    /* Set frame format: 8data, 2stop bit */
    UCSR0C = (1<<USBS0)|(3<<UCSZ00);

}
```

**** Refer to Register Description Page: 200 of ATmega328P datasheet*

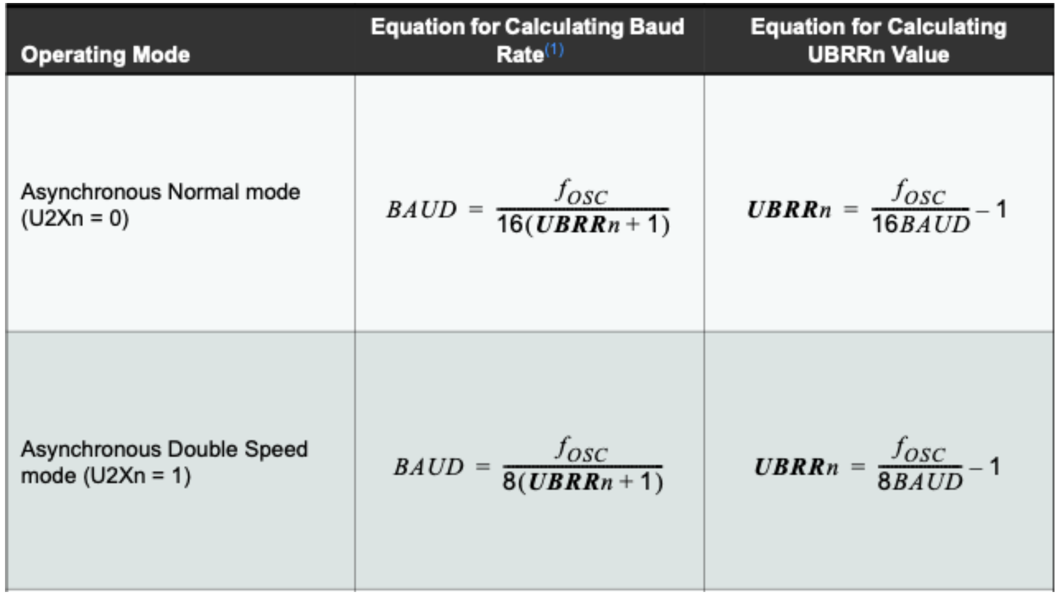
AVR USART: Examples of Baud Rate Setting



Baud Rate (bps)	f_{osc} = 1.0000MHz			
	U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error
2400	25	0.2%	51	0.2%
4800	12	0.2%	25	0.2%
9600	6	-7.0%	12	0.2%
14.4k	3	8.5%	8	-3.5%
19.2k	2	8.5%	6	-7.0%
28.8k	1	8.5%	3	8.5%
38.4k	1	-18.6%	2	8.5%
57.6k	0	8.5%	1	8.5%
76.8k	–	–	1	-18.6%
115.2k	–	–	0	8.5%
230.4k	–	–	–	–
250k	–	–	–	–
Max.[1]	62.5kbps		125kbps	



Baud Rate (bps)	f_{osc} = 8.0000MHz			
	U2Xn = 0		U2Xn = 1	
	UBRRn	Error	UBRRn	Error
2400	207	0.2%	416	-0.1%
4800	103	0.2%	207	0.2%
9600	51	0.2%	103	0.2%
14.4k	34	-0.8%	68	0.6%
19.2k	25	0.2%	51	0.2%
28.8k	16	2.1%	34	-0.8%
38.4k	12	0.2%	25	0.2%
57.6k	8	-3.5%	16	2.1%
76.8k	6	-7.0%	12	0.2%
115.2k	3	8.5%	8	-3.5%
230.4k	1	8.5%	3	8.5%
250k	1	0.0%	3	0.0%
0.5M	0	0.0%	1	0.0%
1M	–	–	0	0.0%
Max. [1]	0.5Mbps		1Mbps	



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

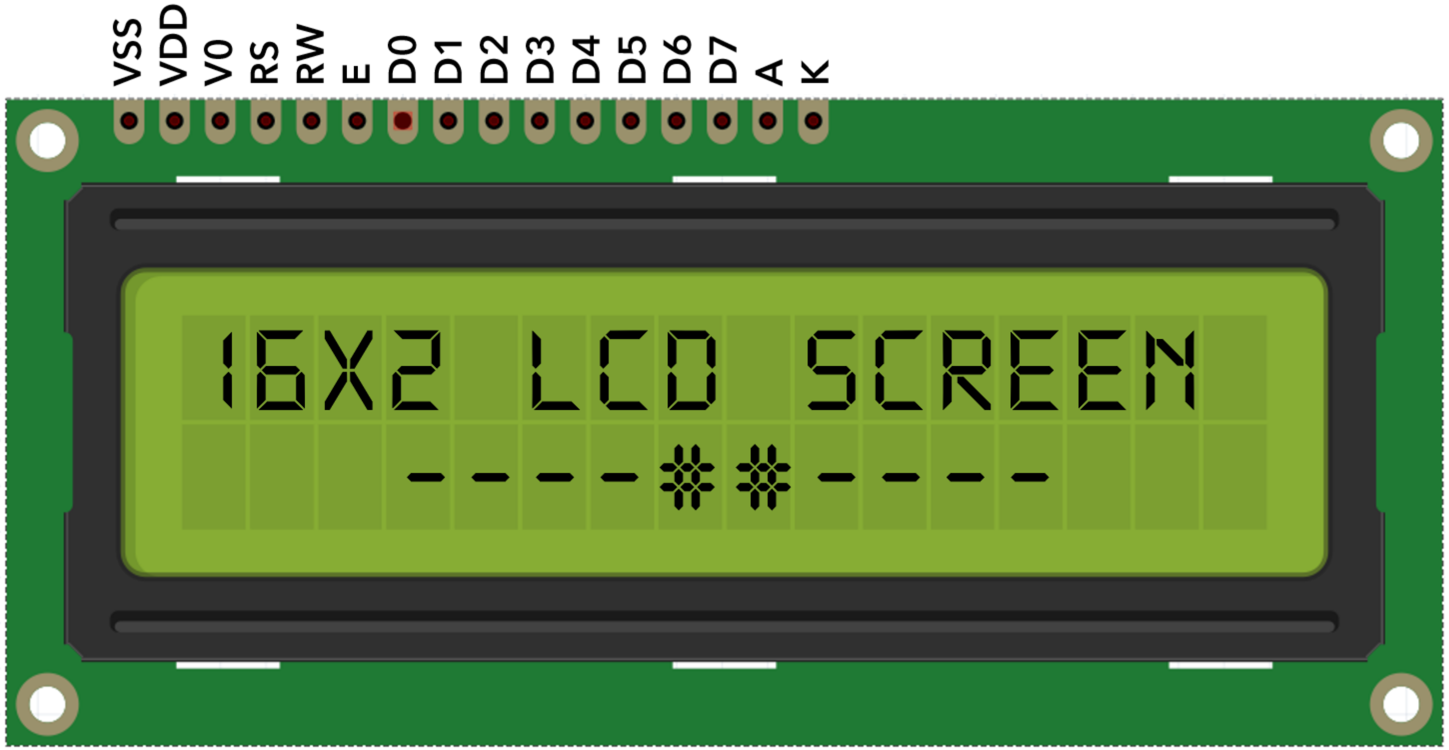
AVR USART: Programming (Send / Receive)

```
void USART_Transmit( unsigned char data ) {
    /* Wait for empty transmit buffer */
    while ( !( UCSRnA & (1<<UDREn)) ) ;

    /* Put data into buffer, sends the data */
    UDRn = data;
}

unsigned char USART_Receive() {
    /* Wait for data to be received */
    while ( !(UCSRnA & (1<<RXCn)) ) ;

    /* Get and return received data from buffer */
    return UDRn;
}
```

**** Refer to Register Description Page: 200 of ATmega328P datasheet*

Pre-Lab 3: HD44780 LCD Interfacing

LCD Pinout

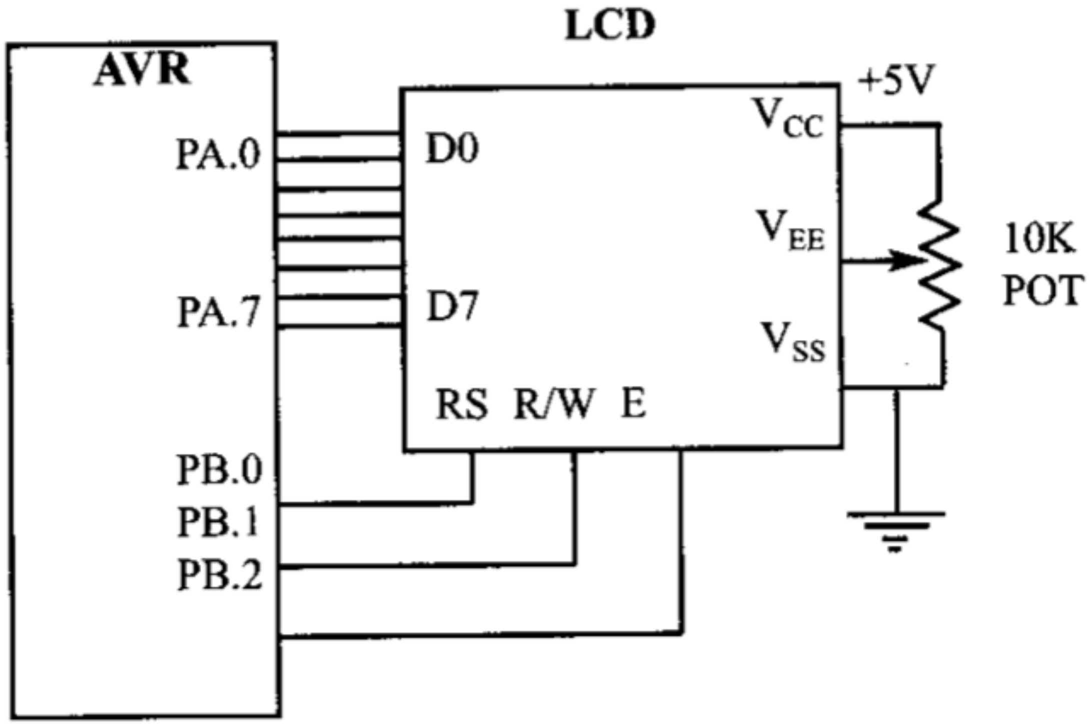


No	Symbol	Function
1	VSS	Ground
2	VDD	5V +
3	V0	Contrast
4	RS	Register
5	RW	Read/Write
6	E	Enable
7	D0	Data bus
8	D1	Data bus
9	D2	Data bus
10	D3	Data bus
11	D4	Data bus
12	D5	Data bus
13	D6	Data bus
14	D7	Data bus
15	A	Anode (5V+)
16	K	Cathode (GND)

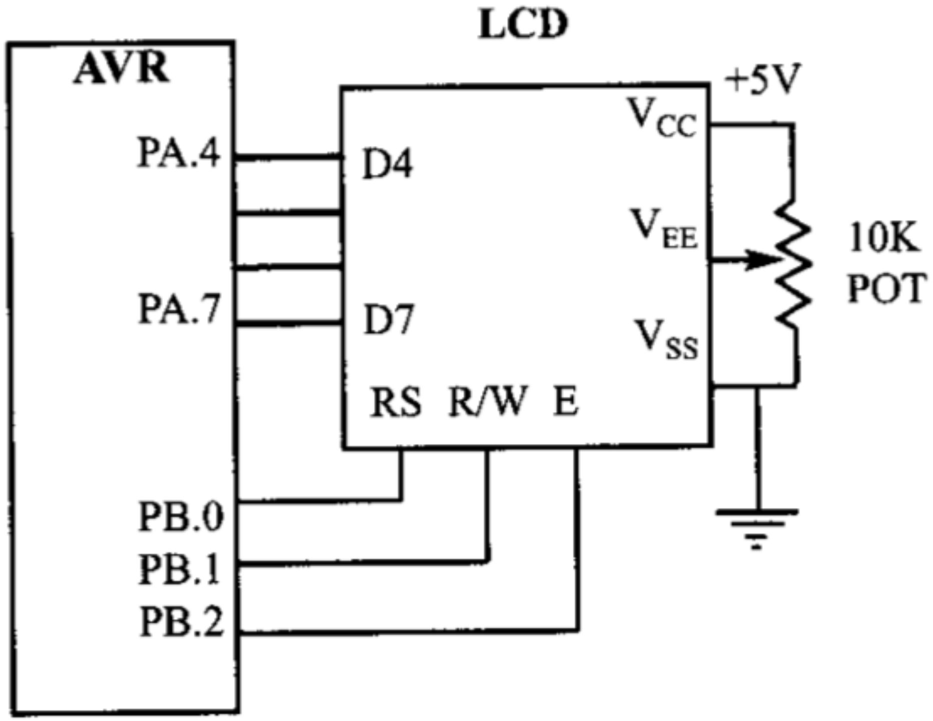
RS
0 - Command Register
1 - Data Register

R/W
0 - Write
1 - Read

Connection Diagram



LCD connections for 8-bit mode

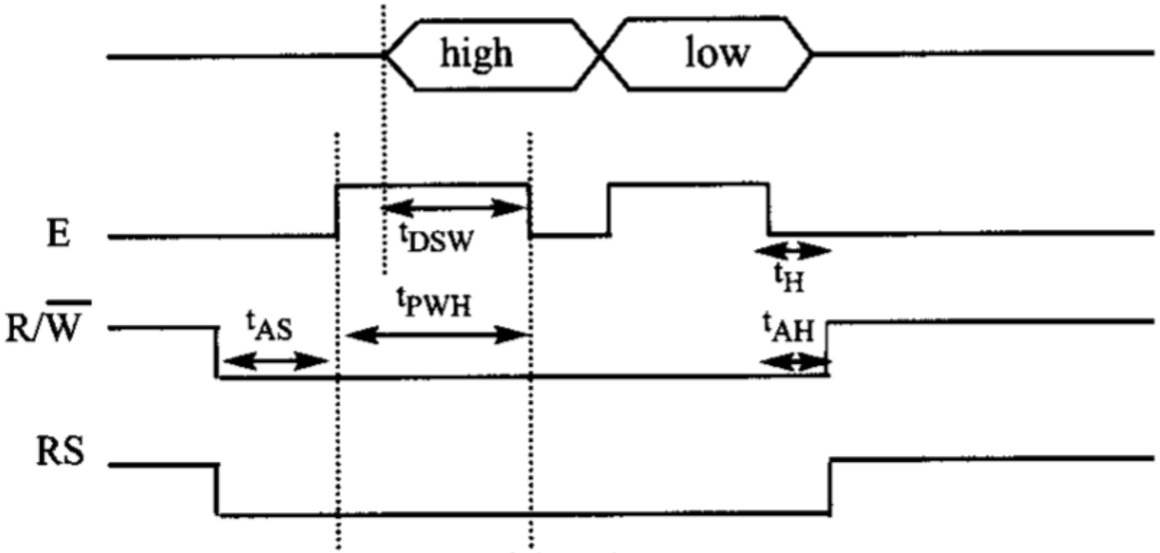


LCD connections for 4-bit mode

Basic LCD Programming

1. Initialize the LCD by sending command to the LCD
   - 0x38, 0x0E, 0x01 for 8-bit mode
   - 0x33, 0x32, 0x28, 0x0E, 0x01 for 4-bit mode
2. Send additional commands to setup the LCD
3. Send the character to be shown on the LCD

Timing Diagram for 4-bit Write



t_{PWH} = Enable pulse width = 450 ns (minimum)

t_{DSW} = Data setup time = 195 ns (minimum)

t_H = Data hold time = 10 ns (minimum)

t_{AS} = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

- Set RS = 0 to write a command and RS = 1 to write a data
- We should wait at least 100us after issue each command (except Clear LCD and Return Home which need 2ms to execute)
- R/W can be tied to GND if we only need to write to the LCD

List of LCD Commands

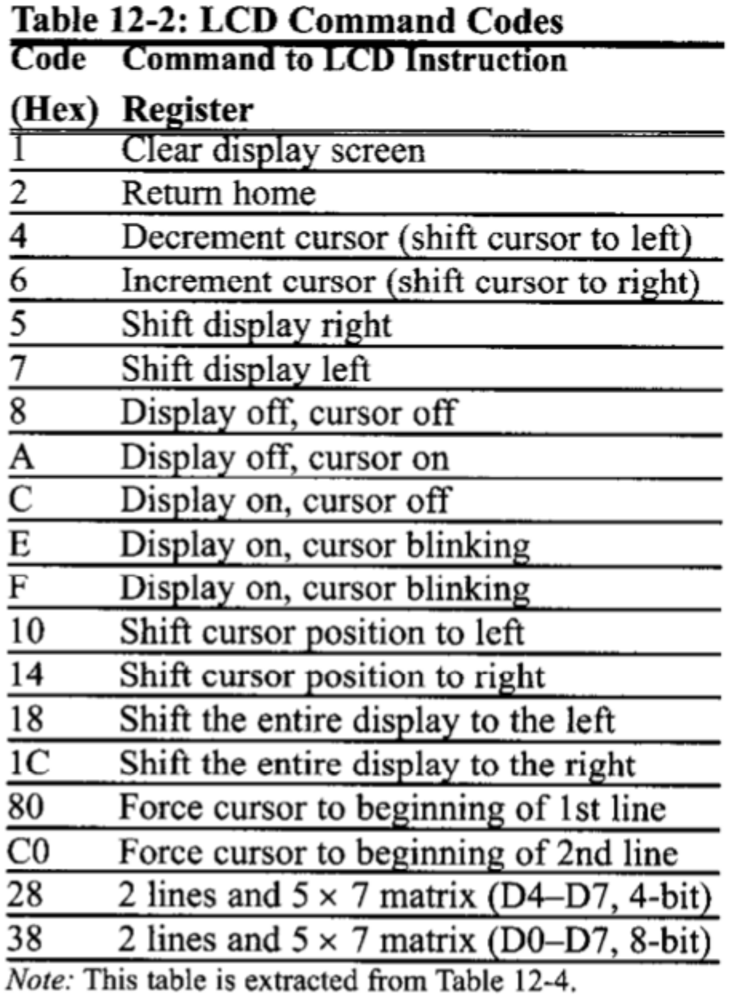
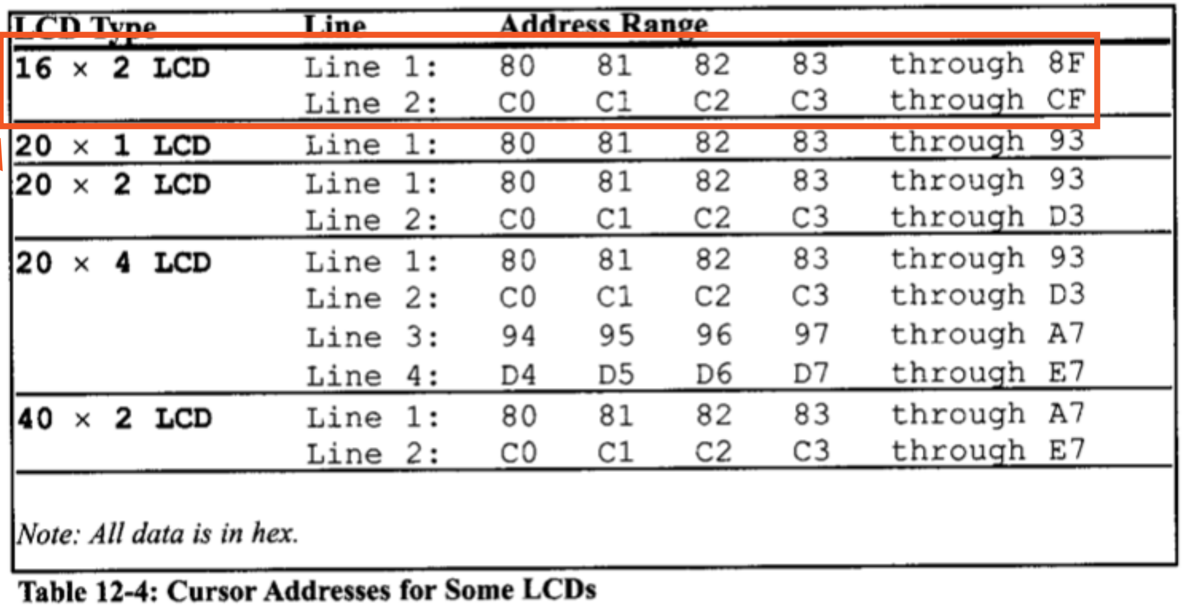


Table 12-2: LCD Command Codes

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking
F	Display on, cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
28	2 lines and 5 × 7 matrix (D4–D7, 4-bit)
38	2 lines and 5 × 7 matrix (D0–D7, 8-bit)

Note: This table is extracted from Table 12-4.



LCD Type	Line	Address Range				
16 × 2 LCD	Line 1:	80	81	82	83	through 8F
	Line 2:	C0	C1	C2	C3	through CF
20 × 1 LCD	Line 1:	80	81	82	83	through 93
20 × 2 LCD	Line 1:	80	81	82	83	through 93
	Line 2:	C0	C1	C2	C3	through D3
20 × 4 LCD	Line 1:	80	81	82	83	through 93
	Line 2:	C0	C1	C2	C3	through D3
	Line 3:	94	95	96	97	through A7
	Line 4:	D4	D5	D6	D7	through E7
40 × 2 LCD	Line 1:	80	81	82	83	through A7
	Line 2:	C0	C1	C2	C3	through E7

Note: All data is in hex.

Table 12-4: Cursor Addresses for Some LCDs

Lab 3: AVR Parallel & UART Programming

1. Write a program to receive your name from the serial port and send “Hello <your name>” back to your PC. Your program should run indefinitely i.e. you should echo “Hello …” back for every name received.
2. Extend the program from the first problem to display “Hello <your name>” on the LCD screen after received name from the serial port. The message on the LCD screen should update correctly after received new name.