Register Overview

Registers	
Data Direction Register (DDRx)	• Tells the microcontroller if a pin is an input or an output

Data Register (PORTx)	• If the pin is an Output
	  ○ 0 = LOW (0V = GND)
	  ○ 1 = HIGH (~5V)
	• If the pin is an Input
	  ○ 0 = no pull-up (can be used to read input)
	  ○ 1 = Pin is listening > weakly HIGH through internal pull-up

Input Pins Register (PINx)	• Reads the voltage of the pin

Port Overview

PORT Name	Port Description
PORTB	• Microcontroller Pins: PB0-PB7 • 8 pins total, only PB0-PB5 are connected to Uno R3 header pins • PB6 and PB7 are dedicated to the external clock crystal • Functions: digital I/O, SPI interface, timer PWM (Pulse Width Modulation) outputs
PORTC	• Microcontroller Pins: PC0-PC6 • 7 pins, only PC0-PC5 are exposed • PC6 is a RESET pin (used to restart the chip) • Functions: ADC inputs, I2C communication, digital I/O
PORTD	• Microcontroller Pins: PD0-PD7 • Functions: UART (serial communication), external interrupts, timer PWM (Pulse Width Modulation), digital I/O

Pin Configuration Overview

Pin Mode	Port Value	What happens?
Input (DDR = 0)	PORT = 0	Pin is listening → Reads whatever is connected.
Input (DDR = 0)	PORT = 1	Pin is listening → when nothing else is connected, pin reads HIGH and avoids random noise/floating values. When a button connects the pin to ground (GND), pin reads low
Output (DDR = 1)	PORT = 0	Pin drives GND
Output (DDR = 1)	PORT = 1	Pin Drive 5V

Pin Overview

Port B Pins

MCU Pin Name	Arduino Function	Arduino Header Pin	Registers
PB0	Digital 8	D8	1. DDRB 2. PORTB 3. PINB
PB1	Digital 9 (PWM)	D9	1. DDRB 2. PORTB 3. PINB
PB2	Digital 10 (PWM,SS)	D10	1. DDRB 2. PORTB 3. PINB
PB3	Digital 11 (PWM, MOSI)	D11	4. DDRB 5. PORTB 6. PINB
PB4	Digital 12 (MISO)	D12	1. DDRB 2. PORTB 3. PINB
PB5	Digital 13 (SCK / onboard LED)	D13	1. DDRB 2. PORTB 3. PINB
PB6	XTAL1 (Crystal input)	Not exposed to headers	1. DDRB 2. PORTB 3. PINB
PB7	XTAL2 (Crystal output)	Not exposed to headers	1. DDRB 2. PORTB 3. PINB

Port C Pins

MCU Pin Name	Arduino Function	Arduino Header Pin	Registers
PC0	Analog 0	A0	1. DDRC 2. PORTC 3. PINC
PC1	Analog 1	A1	1. DDRC 2. PORTC 3. PINC
PC2	Analog 2	A1	1. DDRC 2. PORTC 3. PINC
PC3	Analog 3	A2	1. DDRC 2. PORTC 3. PINC
PC4	Analog 4 (A4 / SDA)	A3	1. DDRC 2. PORTC 3. PINC
PC5	Analog 5 (A5 / SCL)	A4	1. DDRC 2. PORTC 3. PINC
PC6	RESET	RESET	1. DDRC 2. PORTC 3. PINC

Port D Pins

MCU Pin Name	Arduino Function	Arduino Header Pin	Register
PD0	Digital 0 (RX)	D0	1. DDRD 2. PORTD 3. PIND
PD1	Digital 1 (TX)	D1	1. DDRD 2. PORTD 3. PIND
PD2	Digital 2	D2	1. DDRD 2. PORTD 3. PIND
PD3	Digital 3 (PWM)	D3	1. DDRD 2. PORTD 3. PIND
PD4	Digital 4	D4	1. DDRD 2. PORTD 3. PIND
PD5	Digital 5 (PWM)	D5	1. DDRD 2. PORTD 3. PIND
PD6	Digital 6 (PWM)	D6	1. DDRD 2. PORTD 3. PIND
PD7	ADC Power	D7	1. DDRD 2. PORTD 3. PIND

Other pins

MCU Pin Name	Arduino Function	Arduino Header Pin
AVCC	ADC Power	+5V
AREF	Analog Reference	AREF
GND	Ground	GND
VCC	Power +5V	+5V

Pin Function Description

- PWM (Pulse Width Modulation) - A technique where a digital pin switches rapidly between HIGH and LOW to simulate an analog output
    - The pin is always either 5V (HIGH) or 0V (LOW), but the ratio of high time to low time changes
    - The ratio between HIGH time and LOW time is called duty cycle
    - Duty Cycle (%) = $\frac{Time\ HIGH}{Total\ period}$ x 100

Duty Cycle	What it does	Average Voltage (5V system)
0%	Always LOW	0V
25%	High ¼ of the time	1.25V
50%	High ½ of the time	2.5V
75%	High ¾ of the time	3.75V
100%	Always high	5V

    - What is PWM used for
        - LED brightness control - Dim an LED smoothly by changing duty cycle
        - Motor speed control - More high time > motor spins faster
        - Analog-like voltage output - Can be filtered to make a steady voltage for circuits
        - Audio / tone generation - Rapid PWM can generate sound frequencies for buzzers for speakers

- SS (Slave Select) - A digital control line used in SPI communications. This pin is used by the master to tell a specific slave that the master wants to communicate with them

- MOSI (Master Out, Slave In)
    - Used in SPI Communication
    - This is the line where the master sends data to the slave
    - Example: Master sends a byte to an SPI sensor

- MISO (Master In, Slave Out)
    - Used in SPI communication
    - This is the line where the slaves sends data back to the master
    - Example: Sensor sends a reading back to Arduino

- SCK (Serial Clock)
    - Clock line for SPI
    - The master generates this clock to synchronize data transfer on MOSI and MISO

    - Data is only valid on clock edges, so timing is controlled by SCK

- Crystal Input
    - Pins for the external crystal oscillator
    - Provide the microcontroller with a stable clock
    - Not exposed as headers - used internally to run the MCU at precise timing

- SDA (Serial Data)
    - Used in I2C communication
    - Carries the data line for all I2C devices on the bus
    - Can be used to talk to sensors, displays, or other microcontrollers over I2C

- SCL (Serial Clock)
    - Clock line for I2C
    - Master generates this clock to synchronize data on SDA

- RX (Receive)
    - Serial UART receive pin
    - Reads incoming data from another device over UART/Serial
    - Example: Receiving data from a computer or GPS module

- TX (Transmit)
    - Serial UART transmit pin
    - Sends data to another device over UART/Serial
    - Example: Sending sensor readings to a computer

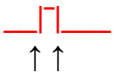
- ADC (Analog-to-Digital Converter)
    - Converts an analog voltage into a digital number
    - ATmega328P has 10-bit ADC, so 0-1023 values correspond to 0-5V
    - Example: Reading a potentiometer

SS, MOSI, MISO, SCK & System Clock

System Clock & Serial Clock (SCK)

System Clock

- A repeating square wave (digital signal) that toggles between HIGH (1) and LOW (0) at a fixed frequency. Each transition from high to low represents an edge.



System clock edges

(8 edges, each associated with a high-low, low-high transition)

- On the ATmega328P, the system clock is 16 MHz (frequency, 16 million cycles per second)
- On full cycle is one complete oscillation of the clock signal: low, high, low
- This clock is used inside the chip to drive instruction timing, timers and peripherals

Serial Clock (SCK)

- The SCK is the SPI serial clock, which derives its timing from the system clock using a prescaler/divider.

- For example, if the system clock is 16 MHz, then the serial clock (SCK) can be 4 MHz if a system clock divider of 4 is used. In this case every 4 system clock cycles, the SCK will undergo a single clock cycle. Each SCK clock cycle corresponds to two bit edges, and at each bit edge one bit can be sent:

SCK (4 MHz) Single Cycle
 *2 SCK Bit Edges*
System Clock (16 MHz) 4 Cycles
 *8 System clock edges*

Master vs Slave (in SPI, with ATmega328P)

- The roles of master-slave are not tied to the silicon itself (aka ATmega328P is not always the master)

- The Master - ATmega328P
    1. Generates the clock speed signal (SCK)
    2. Selects which slave to talk to using SS (Slave Select)
    3. Sends data on MOSI (Master Out Slave In)
    4. Reads data on MISO (Master In Slave Out)

- The Slave - SPI peripheral
    1. Listens for clock from master
    2. Waits until its SS line is pulled LOW
    3. Reads data on MOSI (Master Out Slave In)
    4. Sends data on MISO (Master In Slave Out)