Mechatronics Engineering

Lab 2

ATmega328P Programming

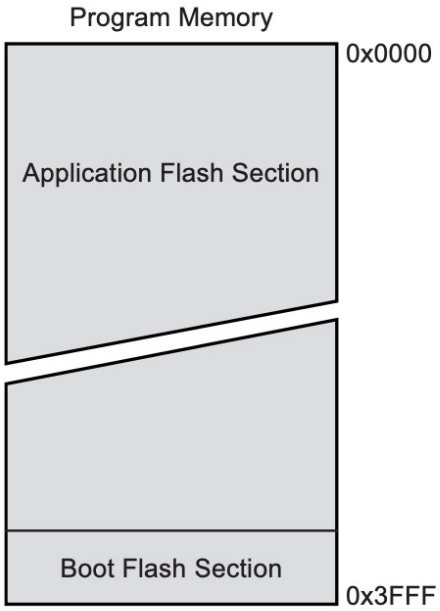
GUC German University in Cairo

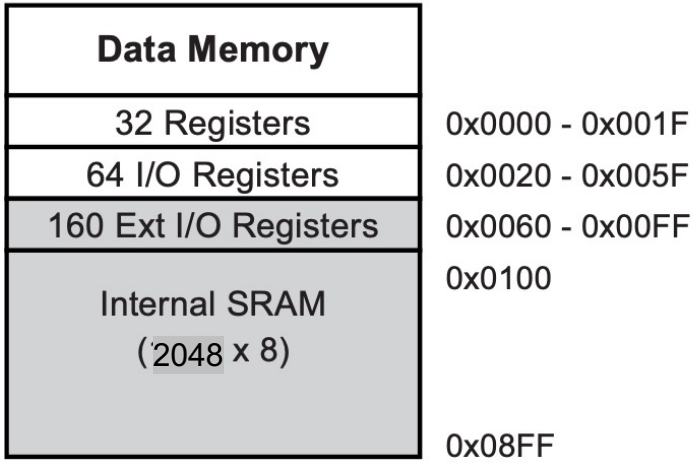
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Tutorial Contents

- ATmega328P Memory
- Input-Output Configuration Registers in ATmega328P
- In Lab Project
- ATmega328P Microcontroller Programming
- Lab 2 Validation

ATmega328P Memory



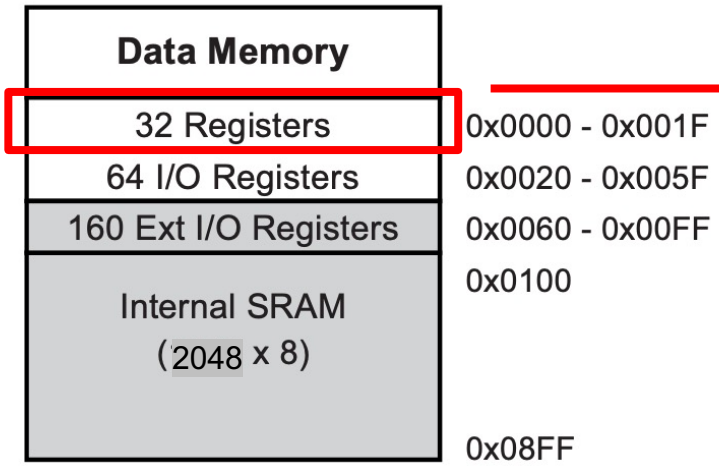


- Program Memory: 32K Bytes of programmable flash memory 1K Bytes of EEPROM

Data Memory

ATmega328P Memory

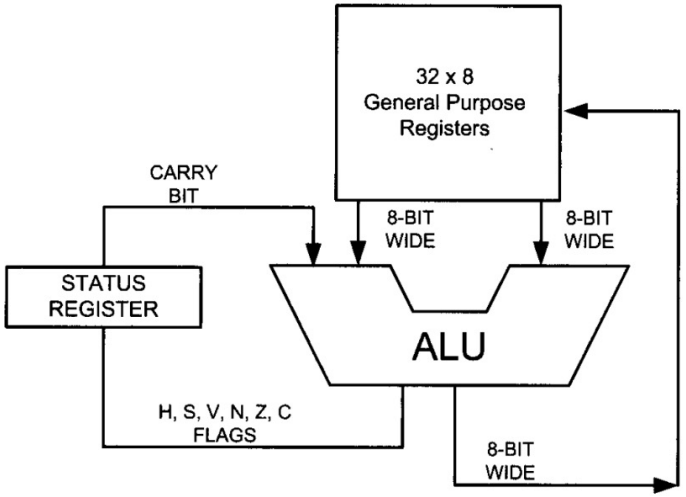
Data Memory:



General Purpose Working Registers

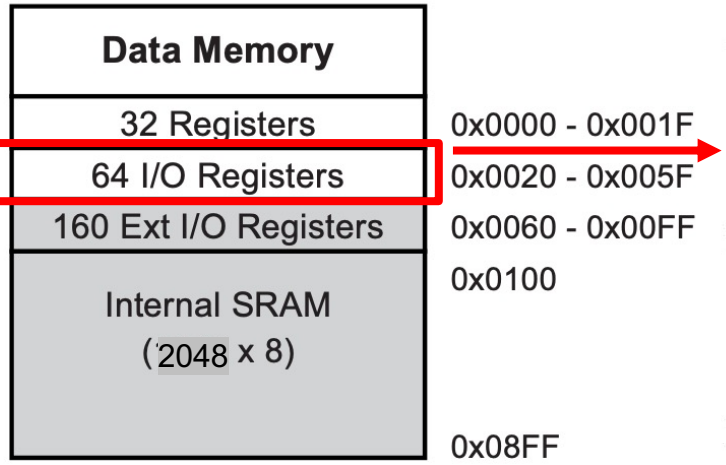
7	0	Addr.
R0		0x00
R1		0x01
R2		0x02
…		
R13		0x0D
R14		0x0E
R15		0x0F
R16		0x10
R17		0x11
…		
R26		0x1A
R27		0x1B
R28		0x1C
R29		0x1D
R30		0x1E
R31		0x1F

- GPWR: Registers used by the CPU to store data temporarily



ATmega328P Memory

Data Memory:



Address		Name
Mem.	I/O	
$20	$00	TWBR
$21	$01	TWSR
$22	$02	TWAR
$23	$03	TWDR
$24	$04	ADCL
$25	$05	ADCH
$26	$06	ADCSRA
$27	$07	ADMUX
$28	$08	ACSR
$29	$09	UBRRL
$2A	$0A	UCSRB
$2B	$0B	UCSRA
$2C	$0C	UDR
$2D	$0D	SPCR
$2E	$0E	SPSR
$2F	$0F	SPDR
$30	$10	PIND
$31	$11	DDRD
$32	$12	PORTD
$33	$13	PINC
$34	$14	DDRC
$35	$15	PORTC

Address		Name
Mem.	I/O	
$36	$16	PINB
$37	$17	DDRB
$38	$18	PORTB
$39	$19	PINA
$3A	$1A	DDRA
$3B	$1B	PORTA
$3C	$1C	EECR
$3D	$1D	EEDR
$3E	$1E	EEARL
$3F	$1F	EEARH
$40	$20	UBRRC
		UBRRH
$41	$21	WDTCR
$42	$22	ASSR
$43	$23	OCR2
$44	$24	TCNT2
$45	$25	TCCR2
$46	$26	ICR1L
$47	$27	ICR1H
$48	$28	OCR1BL
$49	$29	OCR1BH
$4A	$2A	OCR1AL

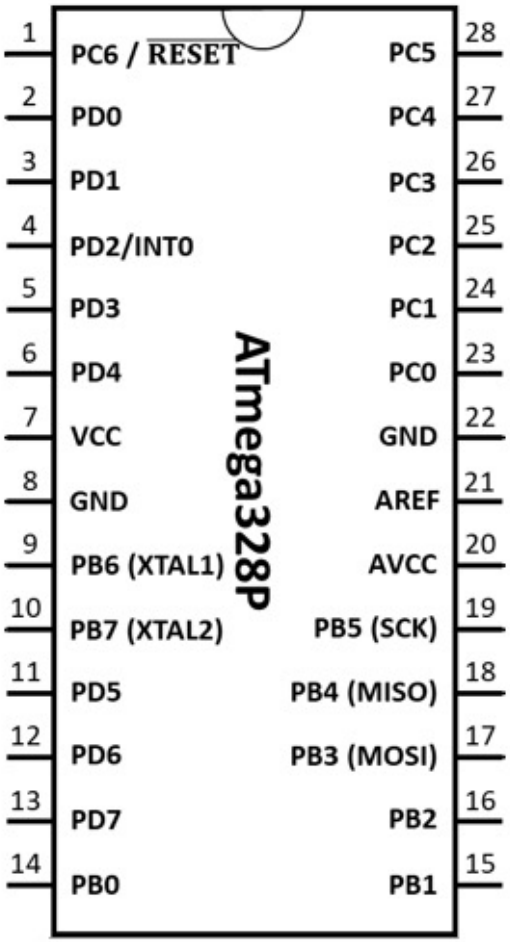
Address		Name
Mem.	I/O	
$4B	$2B	OCR1AH
$4C	$2C	TCNT1L
$4D	$2D	TCNT1H
$4E	$2E	TCCR1B
$4F	$2F	TCCR1A
$50	$30	SFIOR
$51	$31	OCDR
		OSCCAL
$52	$32	TCNT0
$53	$33	TCCR0
$54	$34	MCUCSR
$55	$35	MCUCR
$56	$36	TWCR
$57	$37	SPMCR
$58	$38	TIFR
$59	$39	TIMSK
$5A	$3A	GIFR
$5B	$3B	GICR
$5C	$3C	OCR0
$5D	$3D	SPL
$5E	$3E	SPH
$5F	$3F	SREG

Tutorial Contents

- ATmega328P Memory
- **Input-Output Configuration Registers in ATmega328P**
- In Lab Project
- ATmega328P Microcontroller Programming
- Lab 2 Validation

Input-Output Configuration Registers in ATmega328P

ATmega328p Microcontroller PIN details



Input-Output Configuration Registers in ATmega328P

- ATmega328P has 3 configurable bi-directional input-output ports. They have internal pull-up resistors which can be selected for each bit.
  - PORTB (PB7:0)
  - PORTC (PC6:0)
  - PORTD (PD7:0)

- Each Input-Output port has 3 registers associated with it. They are designated as:
  - PORTx (PORTx Data Register)
  - DDRx (PORTx Data Direction Register)
  - PINx (PORTx Input Pins Register)

- Each of these registers is 8 bits wide. So, each bit affects only one pin that it is associated with.

Input-Output Configuration Registers in ATmega328P

- For example, for PORTD:

PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

When specifiying a pin as input, we check pins in PINx register.

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

DDRDx	PORTDx	Configuration
0	0	Input
0	1	Input with pull-up resistor enabled
1	0	Output (LOW)
1	1	Output (HIGH)

Input-Output Configuration Registers in ATmega328P

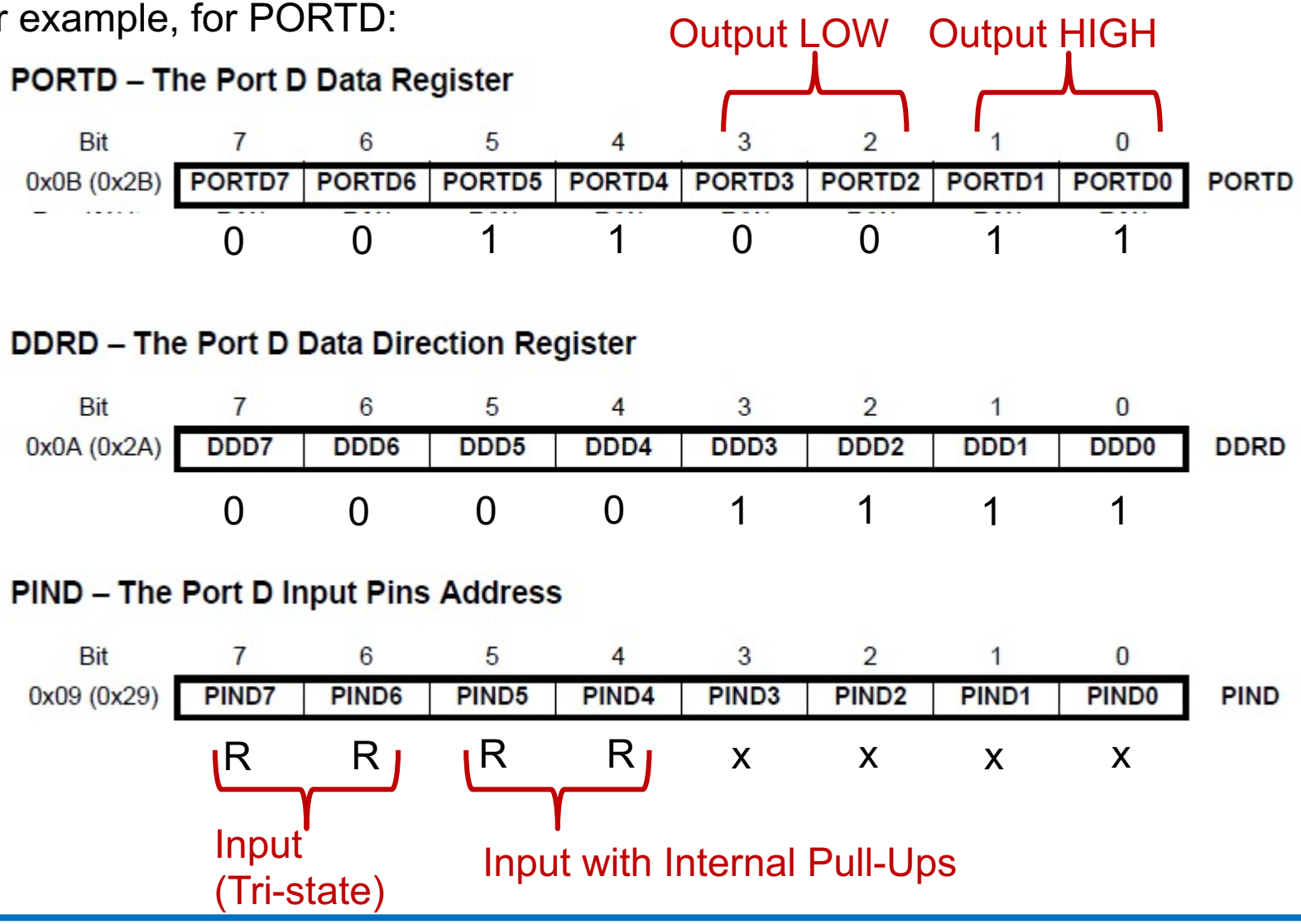
- **Role of DDRx Register:**
  - Since each pin in the 3 ports can be configured as input or output, the DDRx registers is responsible for declaring whether a pin in the corresponding PORTx is input or output.
  - To declare a pin as an output pin, we write 1 (logic HIGH) to its corresponding bit in DDRx.
  - To declare a pin as an input pin, we write 0 (logic LOW) to its corresponding bit in DDRx.

- **Role of PINx Register:**
  - When the DDRx declares the pin as an input pin, the input data present at the pin can be read from the corresponding bit of the PINx register.

- **Role of PORTx Register:**
  - When the DDRx declares the pin as an output pin, the output data sent to the pins can be written to the corresponding bit of the PORTx register.
  - When the DDRx declares the pin as an input pin, the internal pull-up resistors can be activated when the corresponding bit of the PORTx register is set to 1.

Input-Output Configuration Registers in ATmega328P

- For example, for PORTD:



Tutorial Contents

- ATmega328P Memory
- Input-Output Configuration Registers in ATmega328P
- **In Lab Project**
- ATmega328P Microcontroller Programming
- Lab 2 Validation

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Software and Hardware Synchronization

In-Lab Project:

It is required to build a circuit with a push button and a LED indicator.

Circuit Operation:

- When the user presses the push button, the LED indicator turns on.
- When the user releases the push button, the LED indicator turns off.

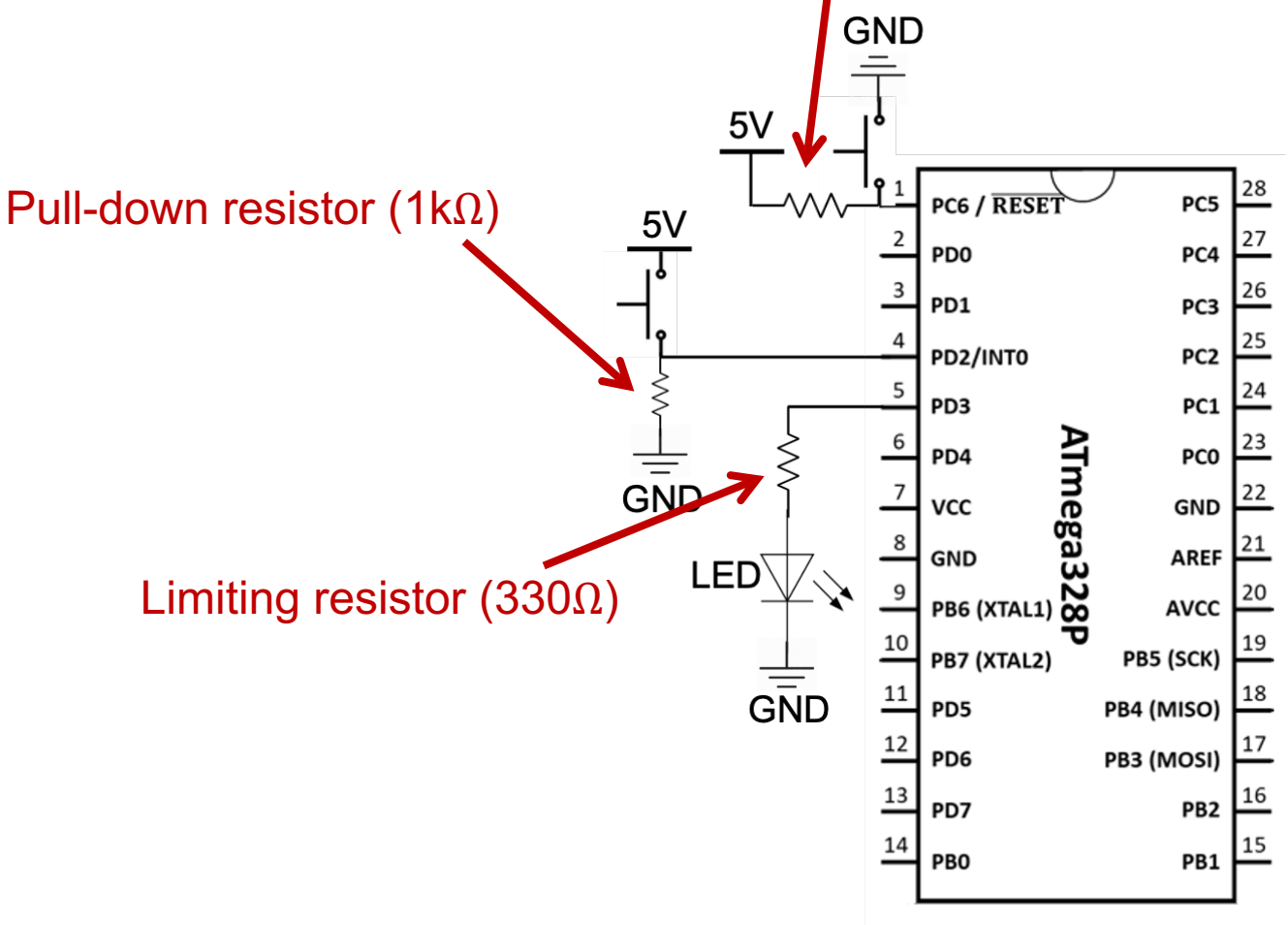
Design Requirements:

- Use AVR Embedded board.
- Use Embedded C for programming.
- The push button is connected to PD2 (PORTD – pin 2).
- The LED indicator is connected to PD3 (PORTD – pin 3).
- Include software button de-bouncing in your implementation.
- Include a reset push button in your hardware implementation.

Software and Hardware Synchronization

In-Lab Project: Solution:

Here are the hardware connections:



Software and Hardware Synchronization

In-Lab Project: Solution 1:

```
#include <avr/io.h>
#define F_CPU 16000000L
#include <util/delay.h>

int main(void)
{
    // Set PORTD with PD3 as output and PD2 and the rest to be input
    DDRD = 0b00001000;
    // Turn off LED (PD3) at start
    PORTD = 0b00000000;
    while (1)
    {
        // check if button (PD2) is pressed
        if(PIND & 0b00000100)
        {
        // software de-bouncing
        _delay_ms(100);
         if (PIND & 0b00000100)
         {  // turn on the LED (PD3)
            PORTD |= 0b00001000;
         }
         }else {
         // otherwise turn off LED (PD3)
         PORTD &= 0b11110111;
        }
    }
    }
```

Some Logical Operators in Embedded C

- **Bitwise Logical AND (&):**
  - Example: 00001111 & 01000010 = 00000010
  - Example: byte & 00000000 = 00000000
  - Example: byte & 11111111 = byte

- **Bitwise Logical OR (|):**
  - Example: 00001111 | 01000010 = 01001111
  - Example: byte | 00000000 = byte
  - Example: byte | 11111111 = 11111111

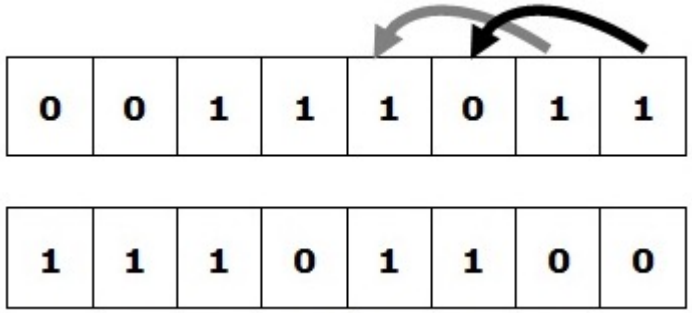
- **To set a bit 2 in PORTD to HIGH using bitmask:**
  PORTD = PORTD | 0b00000100 or PORTD |= 0b00000100

- **To set a bit 2 in PORTD to LOW using bitmask:**
  PORTD = PORTD & 0b11111011 or PORTD &= 0b11111011

- **To check if bit 3 in PIND is HIGH using bitmask:**
  if (PIND & 0b00001000)

- **To check if bit 3 in PIND is LOW using bitmask:**
  if (! (PIND & 0b00001000))

Some Logical Operators in Embedded C



- **Bitwise Shift Left (<<):**
  - Example: 00111011 << 2 = 11101100
  - Example: 1 << 3 = 1000 = 00001000
- **Bitwise Logical NOT (!):**
  - Example: ~01000010 = 10111101
  - Example: ~(1 << 3) = ~00001000 = 11110111
- **To set a bit 2 in PORTD to HIGH using bitmask:**
  PORTD = PORTD | (1<<2) or PORTD |= (1<<2)
- **To set a bit 2 in PORTD to LOW using bitmask:**
  PORTD = PORTD & ~(1<<2) or PORTD &= ~(1<<2)
- **To check if bit 3 in PIND is HIGH using bitmask:**
  if (PIND & (1<<3))
- **To check if bit 3 in PIND is LOW using bitmask:**
  if (~ (PIND & (1<<3)))

GUC German University in Cairo
Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Software and Hardware Synchronization

In-Lab Project: Another Solution (2):

```
#include <avr/io.h>
#define F_CPU 16000000L
#include <util/delay.h>

int main(void)
{
    // Set PORTD with PD3 as output (LED)
    DDRD |= (1<<DDD3);
    // Set PORTD with PD2 as input (Button)
    DDRD &= ~(1<<DDD2);
    // Turn off LED (PD3) at start
    PORTD &= ~(1<<PORTD3);
    while(1)
    {
        // check if button (PD2) is pressed
        if(PIND & (1<<PIND2))
        {
        // software de-bouncing
        _delay_ms(100);
         if (PIND & (1<<PIND2))
         {  // turn on the LED (PD3)
            PORTD |= (1<<PORTD3);
         }
         }else {
         // otherwise turn off LED (PD3)
         PORTD &= ~(1<<PORTD3);
        }
    }
}
```

Tutorial Contents

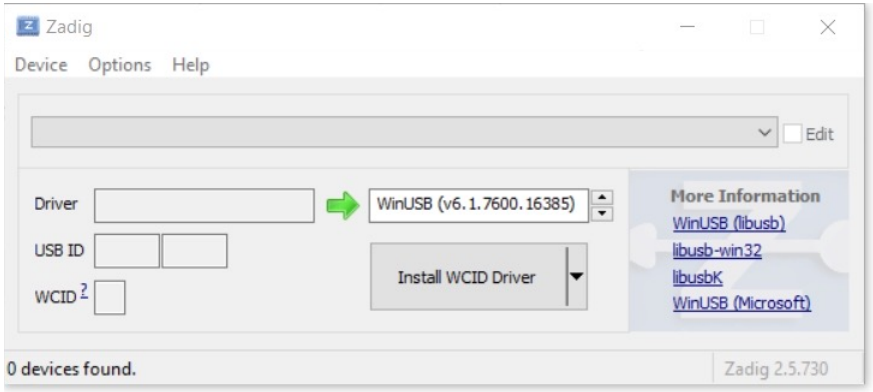
- ATmega328P Memory
- Input-Output Configuration Registers in ATmega328P
- In Lab Project
- **ATmega328P Microcontroller Programming**
- Lab 2 Validation

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Driver Installation

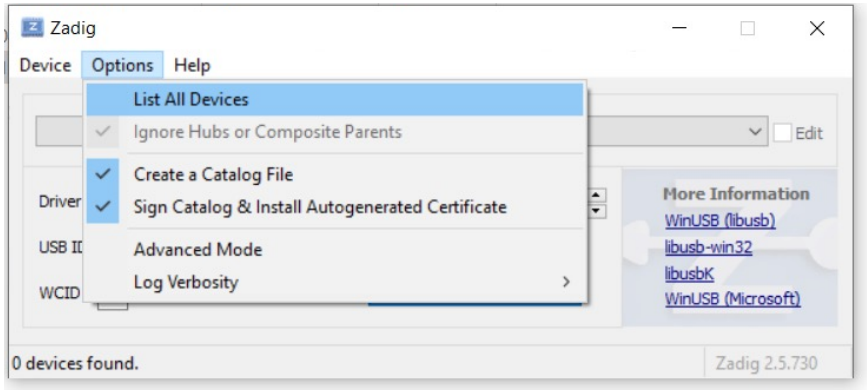
- Download Zadig latest version from https://zadig.akeo.ie/, tested version is v2.5.
- Plug the board in any USB port

- Start Zadig and you should see this screen

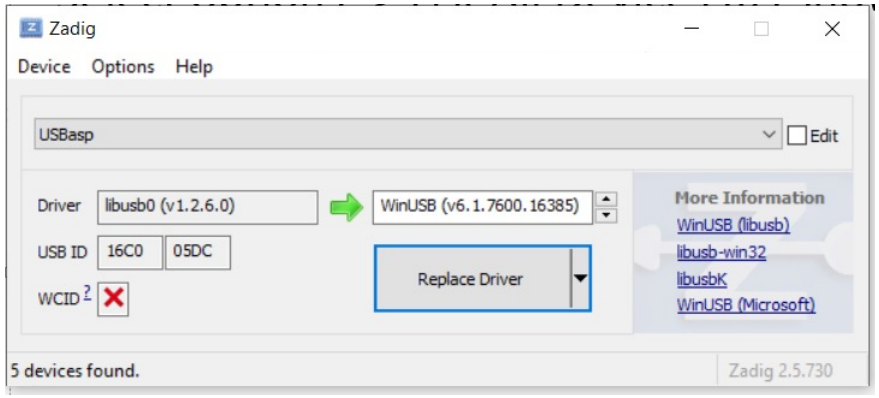


Driver Installation

- Select options -> List All Devices (as shown)

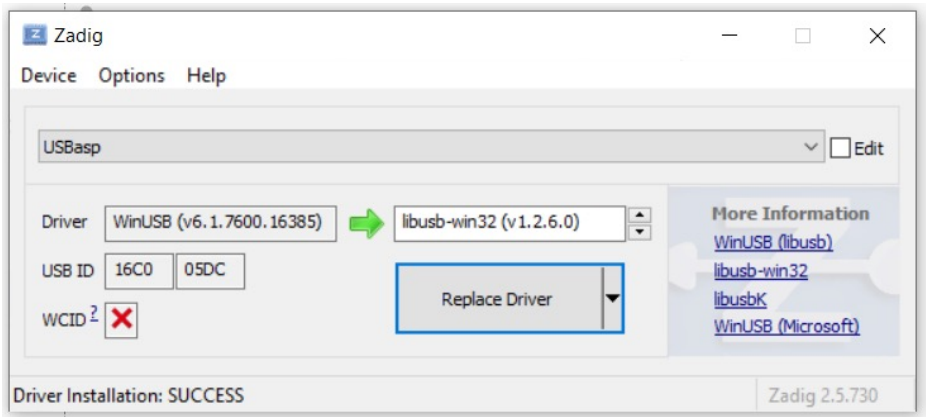


- The following screen appears (if USBasp is not selected use the drop down arrow to select it)



Driver Installation

- Change the driver to be installed (on the right) to be libusb-win32(v1.2.6.0) as shown below



- Click on “Replace Driver” (if no driver is installed already, it will be “Install Driver”)
- It will take some time and the driver will be successfully installed

Installing Avrdude

- Download Avrdude latest version from http://download.savannah.gnu.org/releases/avrdude/. Tested version is v6.3.

- Copy “avrdude-6.3-mingw32” folder and place it in the C:\ (root directory)

- Read more on avrdude at this link : https://www.nongnu.org/avrdude/user-manual/avrdude_4.html

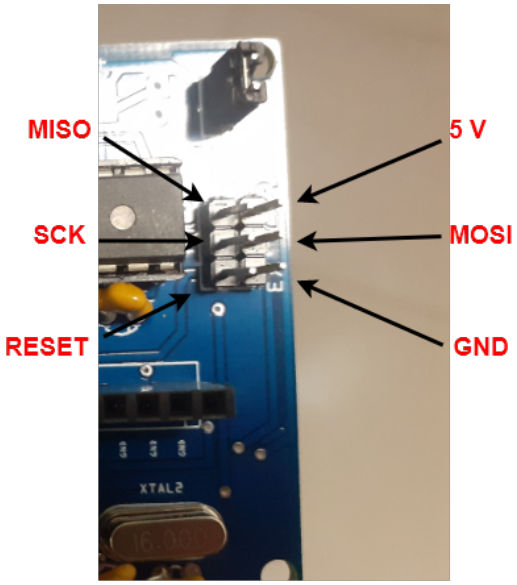
Fuse Settings

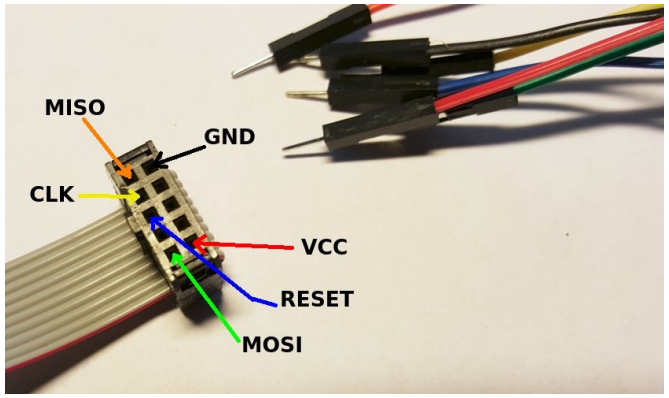
- ATmega8A Correct fuse setting to be read correctly by the pc driver
  - Low fuse : 0x9F
  - High fuse : 0xD9
  - Lock fuse : 0xFF

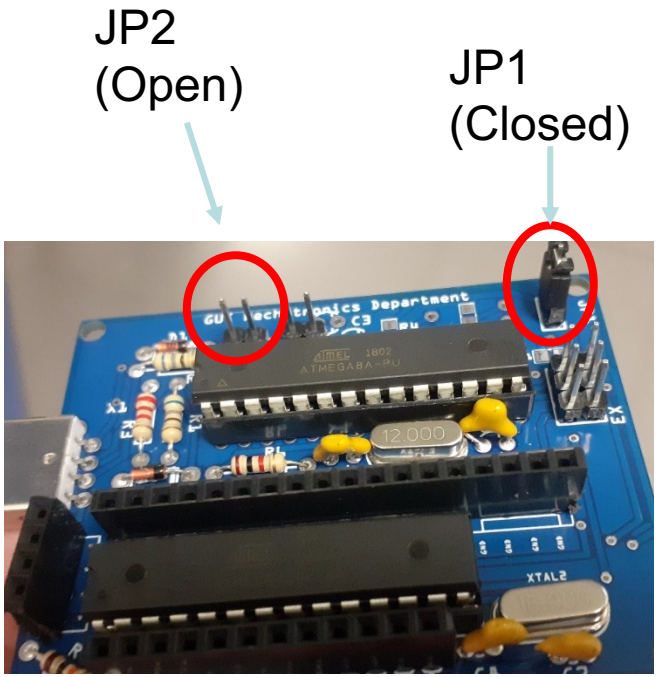
- ATmega328P Correct fuse setting
  - Low fuse : 0xF7
  - High fuse : 0xD9
  - Extended fuse : 0xFF
  - Lock fuse : 0xFF

Programming Schematic

- Connect the following schematic to program the ATmega8A

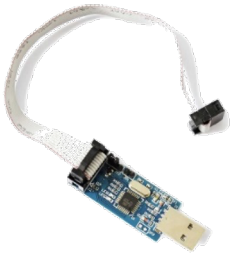






Writing Fuses

- In order to set the fuses another programmer is needed
- Plug the programmer connected with the previous schematic



- Close both jumpers JP1 and JP2 (like JP1 in the previous Slide)
- Open cmd
- Execute the following command to change directory
  cd C:\avrdude-6.3-mingw32
- Execute the following command to write fuses (Atmega8A)
  avrdude -c usbasp -p m8 -P usb -B 93.75 -U lfuse:w:0x9F:m -U hfuse:w:0xD9:m

Burn Firmware

- Download the firmware from https://www.fischl.de/usbasp/usbasp.2011-05-28.tar.gz
- Connect the programming schematic
- Close JP1 and JP2
- Open cmd
- Execute the following command to change directory

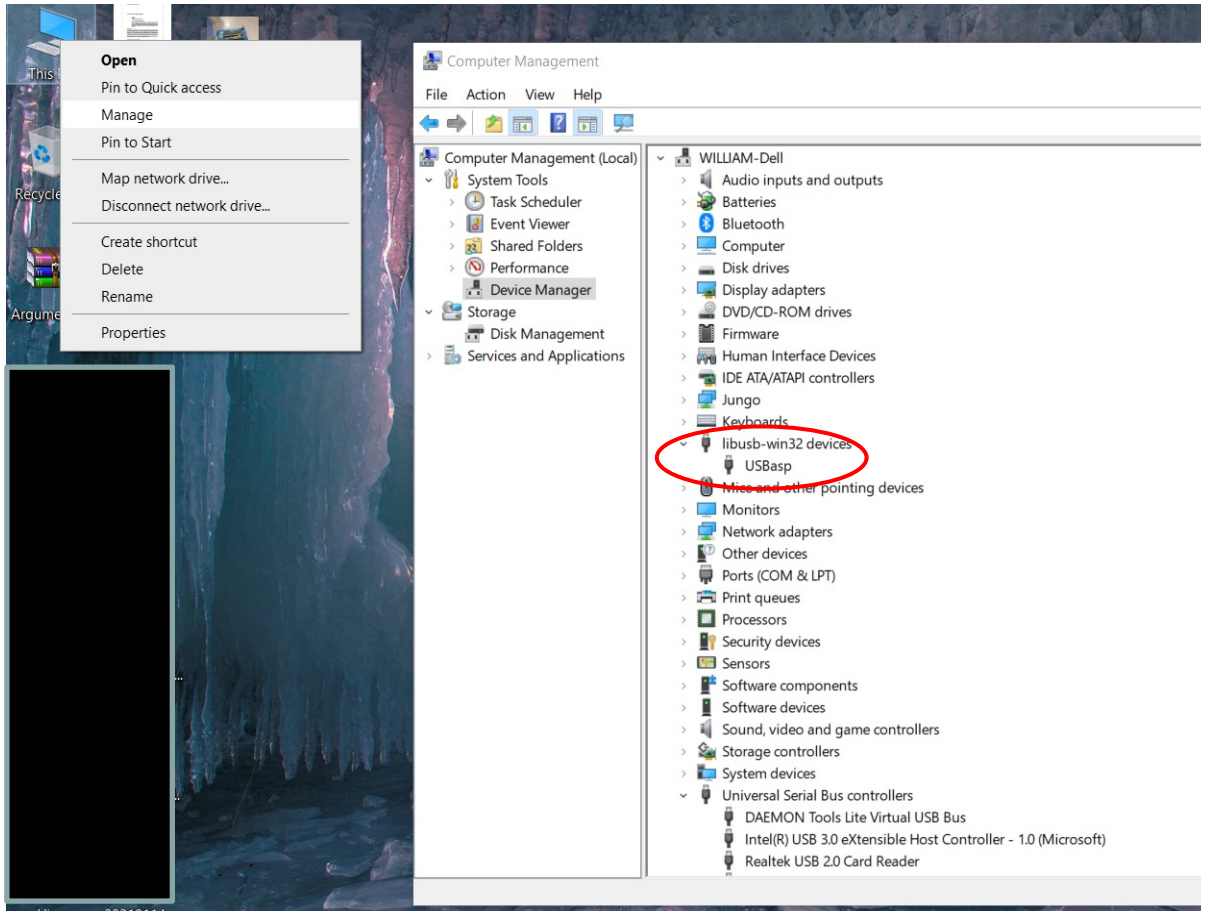
  cd C:\avrdude-6.3-mingw32

- Execute the following command to write firmware (Atmega8A)

  avrdude -c usbasp -p m8 -P usb -B 93.75 -U flash:w:your path\usbasp.2011-05-28\bin\firmware\usbasp.atmega8.2011-05-28.hex:a

Programmer Built Successfully

- To confirm the correct functionality of the programmer
  - Disconnect the external AVR programmer
  - Open JP2
  - Close JP1
  - Plug the kit into your pc
  - Open device manager and make sure the device is loaded as shown



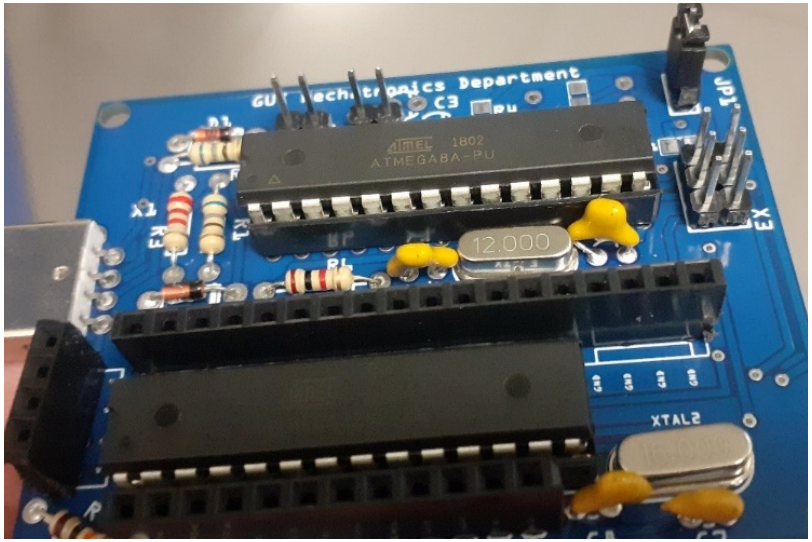
GUC German University in Cairo

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Atmega328P Programming Configuration

Jumpers and connectors should be set as shown in the figure:

- JP1 -> Closed
- JP2 -> Open
- JP3 -> Open
- SPI External Interface -> Disconnected



Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Atmega328P Fuses Setting

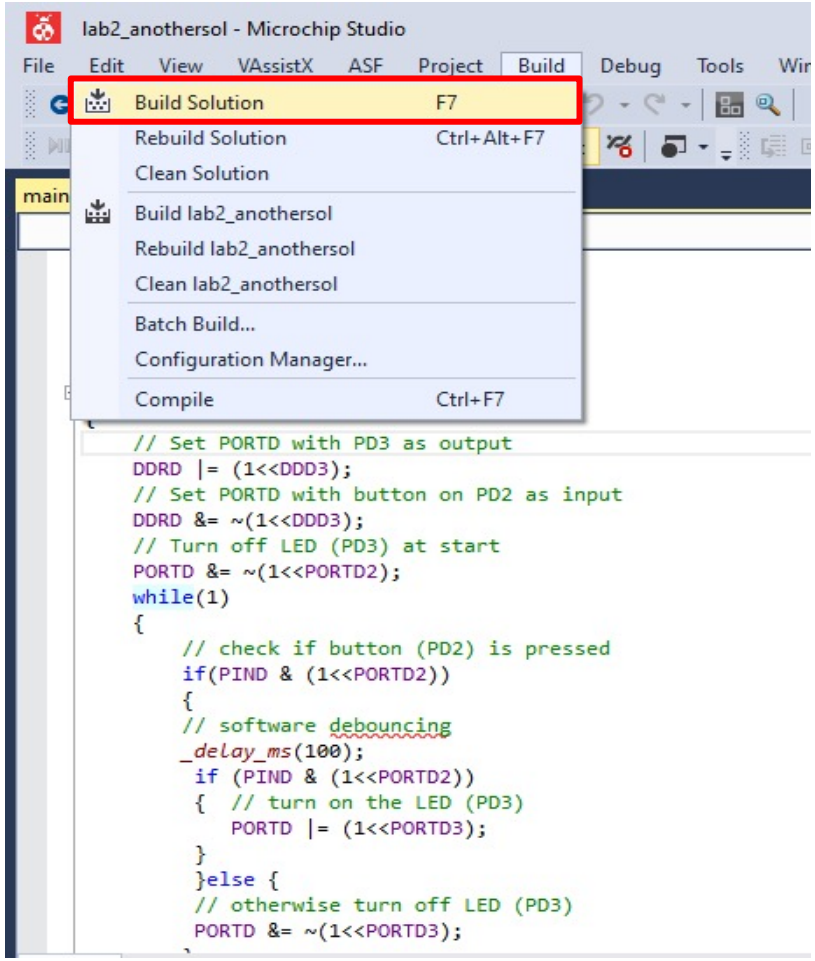
- Atmega328P fuses are now to be set using the programmer already built.
- Set the kit to the Atmega328P programming configuration as in the previous slide.
- Plug the kit into the PC
- Open cmd
- Execute the following command to change directory

  cd C:\avrdude-6.3-mingw32
- Execute the following command to change directory

  avrdude -c usbasp -p m328p -P usb -B 93.75 -U lfuse:w:0xF7:m -U hfuse:w:0xD9:m -U efuse:w:0xFF:m -U lock:w:0xFF:m

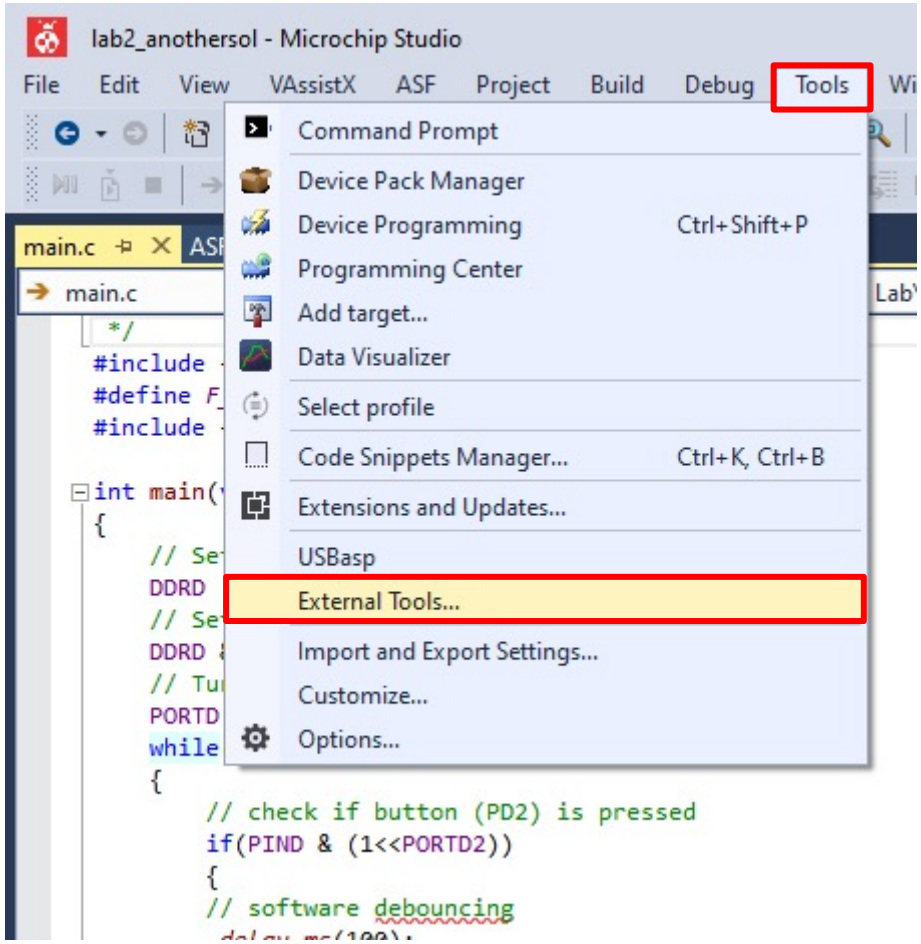
Setting up Microchip Studio

- Build your project as shown below



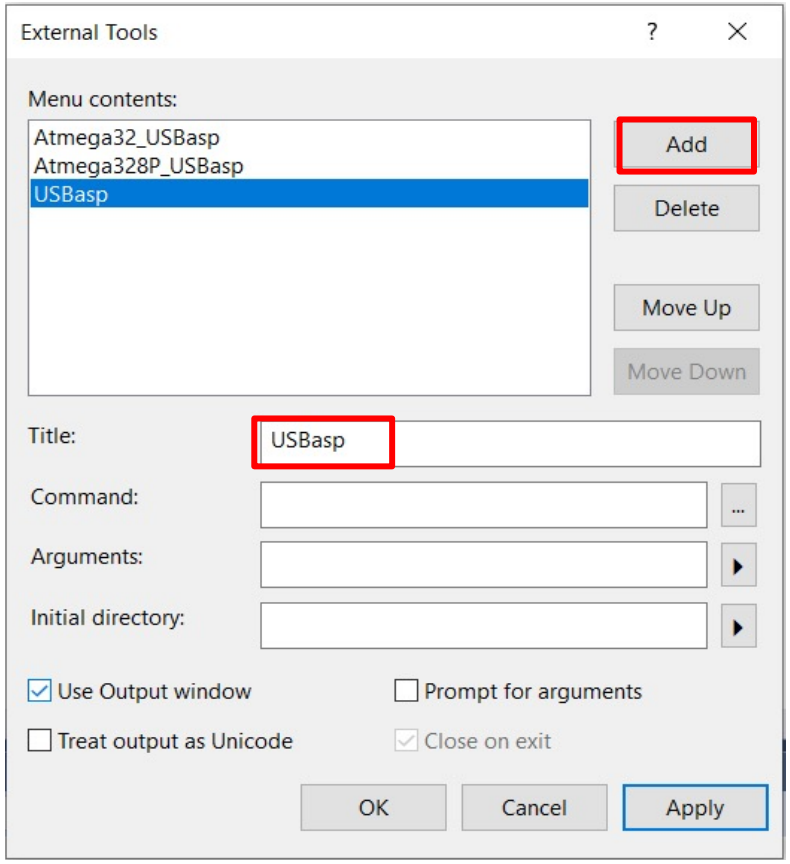
Setting up Microchip Studio

- Add external tool in Atmel Studio in order to be able to program ATmega328P directly.



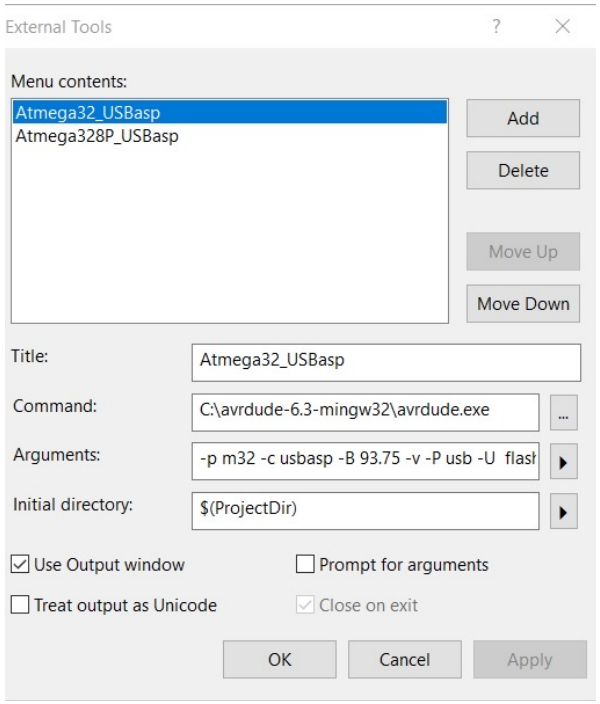
Setting up Microchip Studio

- Click “Add” then name the tool as follows



Setting up Microchip Studio

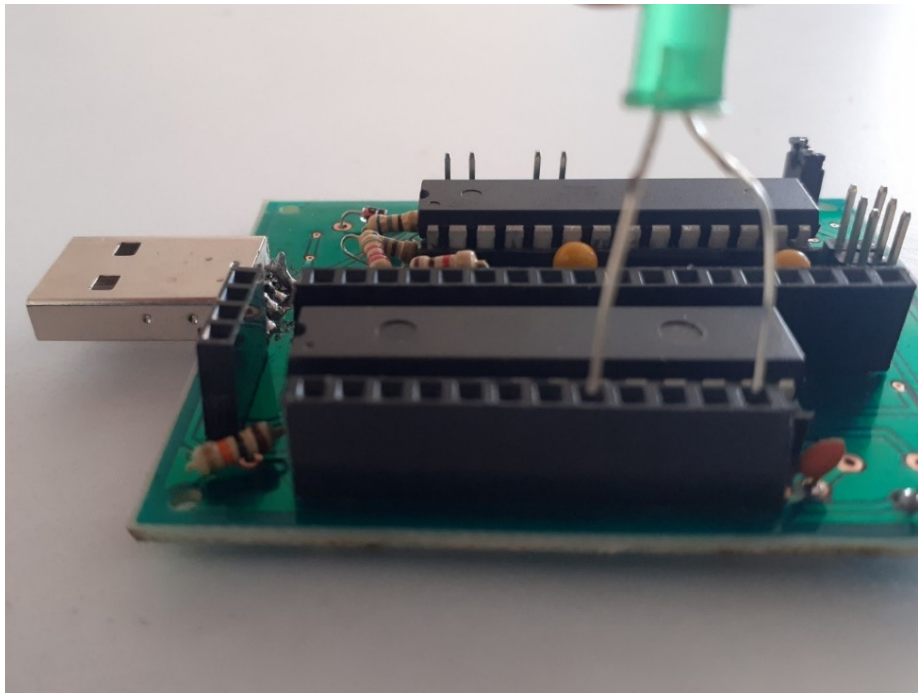
- Add the following line in the command area
  C:\avrdude-6.3-mingw32\avrdude.exe
- Add the following line in the Arguments area
  -p m328p -c usbasp -B 93.75 -v -P usb -U flash:w:"$(OutDir)Debug\$(TargetName).hex:a"
- Add the following line in the Initial Directory area
  $(ProjectDir)
- The final configuration should be as shown in the snippet
- Click “ok”



Testing Hardware Connection

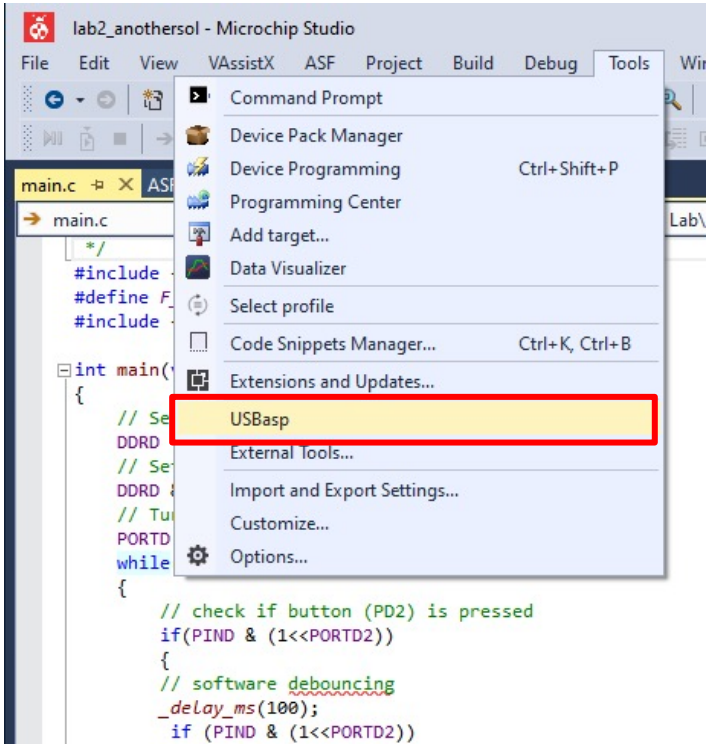
- Connect the anode of an LED to D3 pin (PORTD3) and the cathode to the ground pin as shown in the image below

- Plug the Kit into your pc



Testing Hardware Connection

- Upload Code from Atmel Studio
- You will find the tool created with the given name as shown



- In order to program the ATmega328P, click the tool name (USBasp)

Tutorial Contents

- ATmega328P Memory
- Input-Output Configuration Registers in ATmega328P
- In Lab Project
- ATmega328P Microcontroller Programming
- **Lab 2 Validation**

Prof. Ayman A. El-Badawy
Department of Mechatronics Engineering
Faculty of Engineering and Material Science

Lab 2 Validation: (To be submitted in your Lab next week)

It is required to build a circuit with one push button and two LEDs.

Circuit Operation:

- When the user presses the push button, the LED indicator 1 turns on.
- After the user presses the push button five times, the LED indicator 2 turns on.

Design Requirements:

- Use AVR Embedded board.
- Use Embedded C for programming.
- The push button is connected to PD2.
- The LED indicators are connected to PB0 and PB1.
- Include software button de-bouncing in your implementation.
- Include a reset push button in your hardware implementation.