

Mechatronics Engineering

In-System Programming of AVR Microcontrollers

Contents

- AVR Microcontroller Programming Methods
 - Self-programming
 - In-System Programming (ISP)
 - ❖ Hardware Connection
 - ❖ Software Setup
 - ❖ Fuse Bits Setup
 - ❖ Create Project
 - ❖ Download Program

Contents

- **AVR Microcontroller Programming Methods**

- Self-programming
- In-System Programming (ISP)
 - ❖ Hardware Connection
 - ❖ Software Setup
 - ❖ Fuse Bits Setup
 - ❖ Create Project
 - ❖ Download Program

AVR Microcontroller Programming Methods

- Many AVR Microcontrollers support self-programming.
- For these devices, there are two main options for programming a microcontroller:
 1. **Self-programming** through serial communication (as in Arduino boards).
 1. **In-System Programming (ISP)** (requires external programmer board):
 - ❖ Through Serial Peripheral Interface (**SPI**) communication protocol.
 - ❖ Through parallel programming interface.

Contents

- **AVR Microcontroller Programming Methods**
 - **Self-programming**
 - In-System Programming (ISP)
 - ❖ Hardware Connection
 - ❖ Software Setup
 - ❖ Fuse Bits Setup
 - ❖ Create Project
 - ❖ Download Program

Self-programming

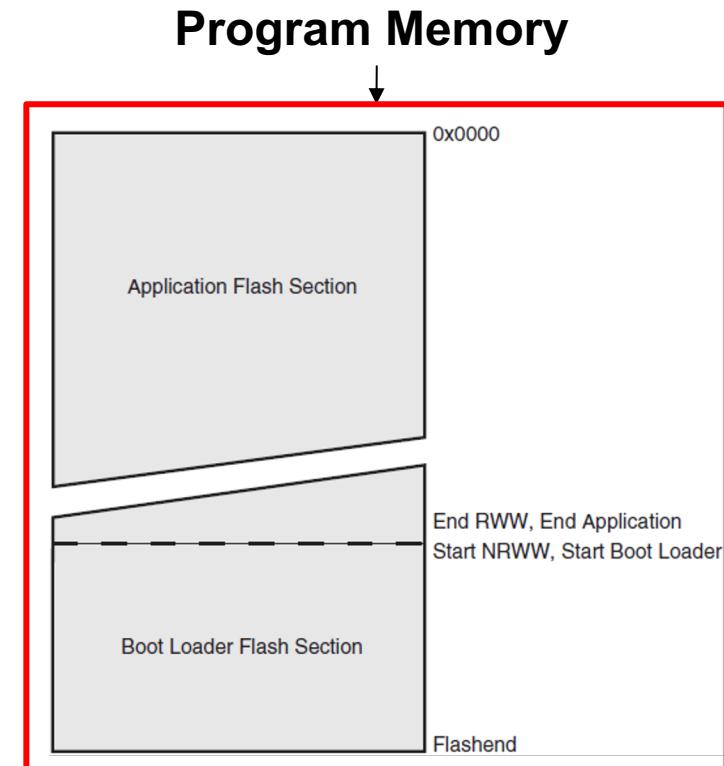
- Program memory is divided into two parts for MCU with self-programming capability:

1. Application Flash Section:

- ❖ Stores application code.

1. Boot Loader Section:

- ❖ Contains code (firmware) that allows reprogramming the program memory.
- ❖ **Example:** Arduino boot loader code reads serial data and writes it to program memory.



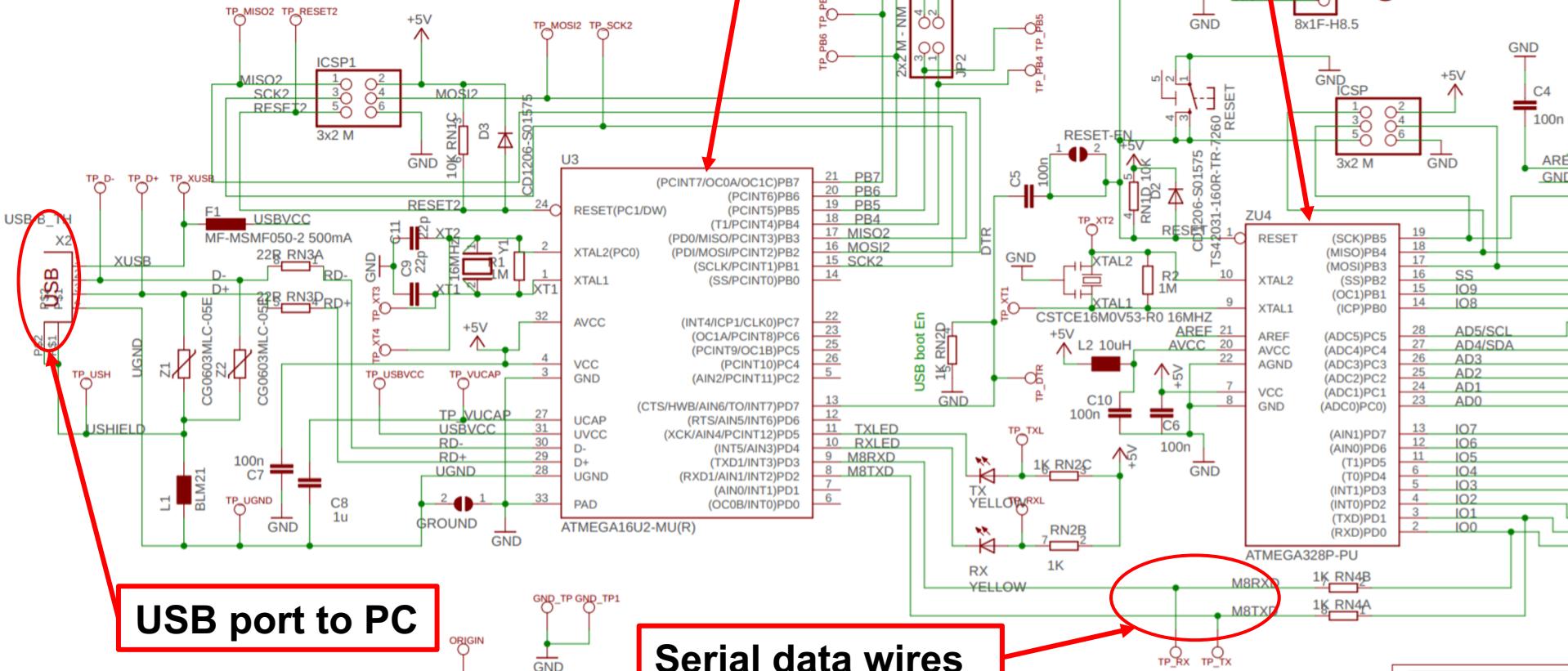
Self-programming

ATMEGA16U2 (bridge between PC USB and MCU)

Atmega328P MCU with bootloader

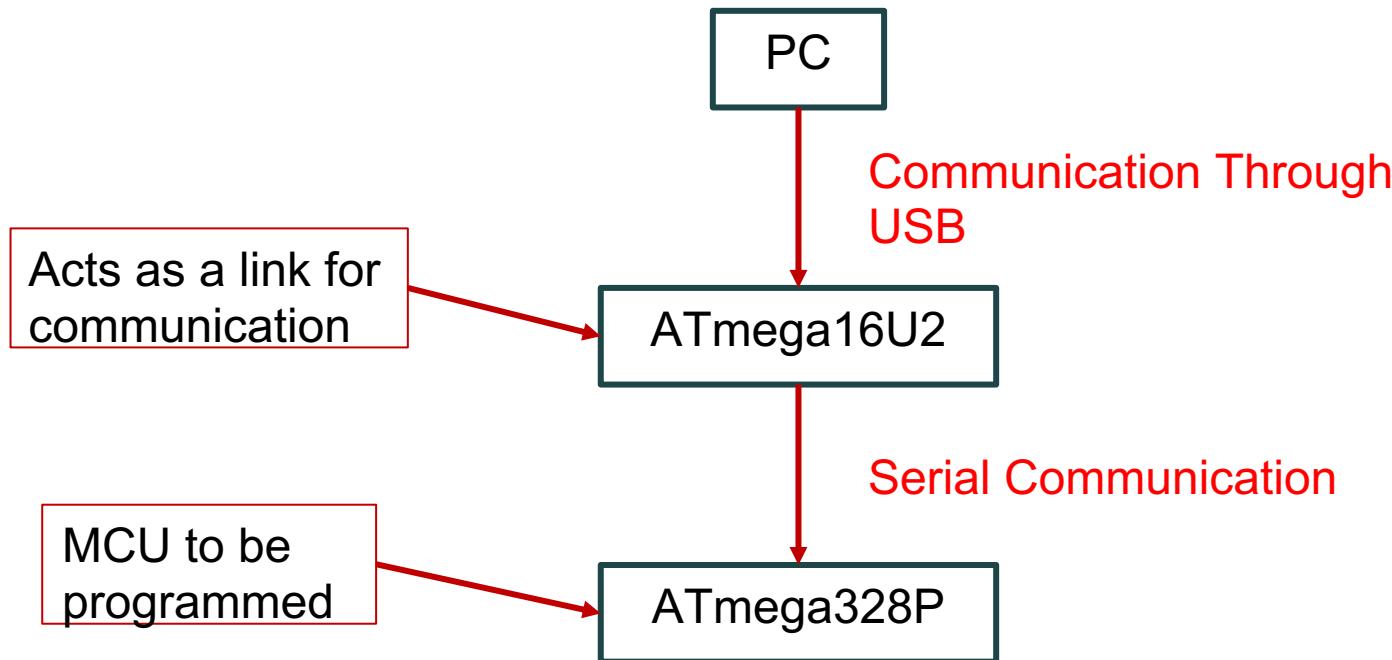
- Arduino Uno Board Circuit:

Arduino(TM) UNO Rev3



Self-programming

- Programming ATmega328P from PC on Arduino Uno Board:



- Code responsible of reprogramming the Atmega328P is written in the bootloader section.

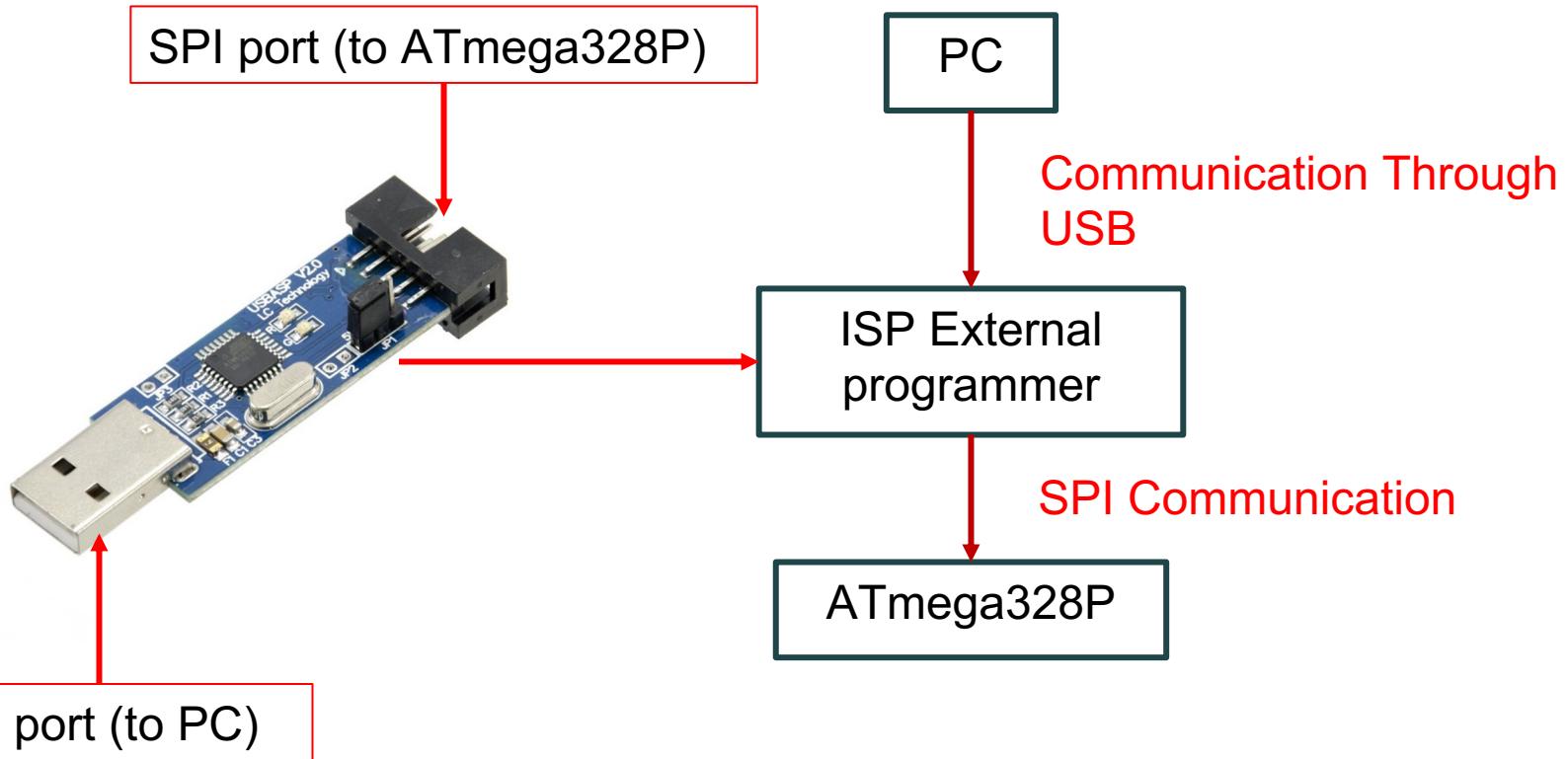
Contents

- **AVR Microcontroller Programming Methods**

- Self-programming
- **In-System Programming (ISP)**
 - ❖ Hardware Connection
 - ❖ Software Setup
 - ❖ Fuse Bits Setup
 - ❖ Create Project
 - ❖ Download Program

In-System Programming (ISP)

- Program Memory can be programmed using SPI communication protocol.
- Connection between ISP programmer (USBasp) and microcontroller:



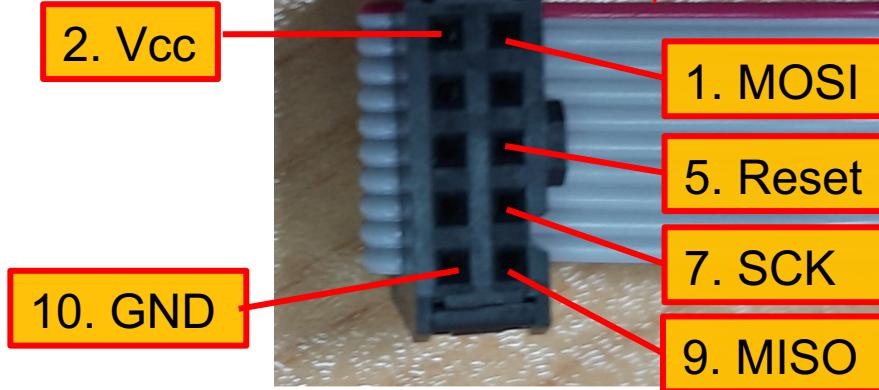
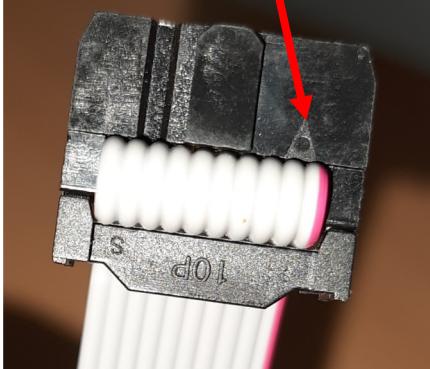
Contents

- **AVR Microcontroller Programming Methods**
 - Self-programming
 - **In-System Programming (ISP)**
 - ❖ **Hardware Connection**
 - ❖ Software Setup
 - ❖ Fuse Bits Setup
 - ❖ Create Project
 - ❖ Download Program

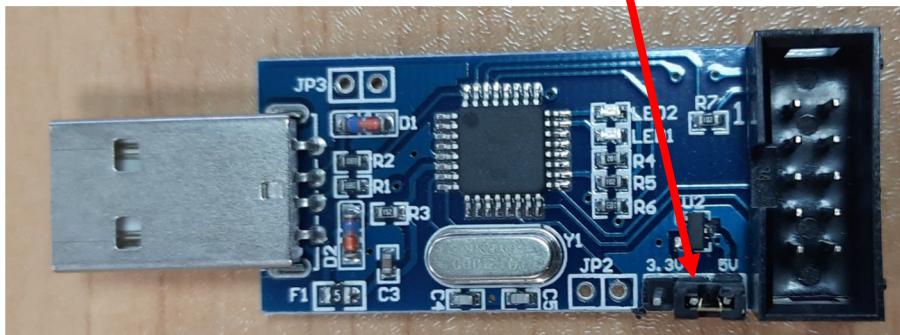
In-System Programming (ISP) Connector

- 10-pin Programmer Connector Label:

Usually, a triangle indicates pin 1 location

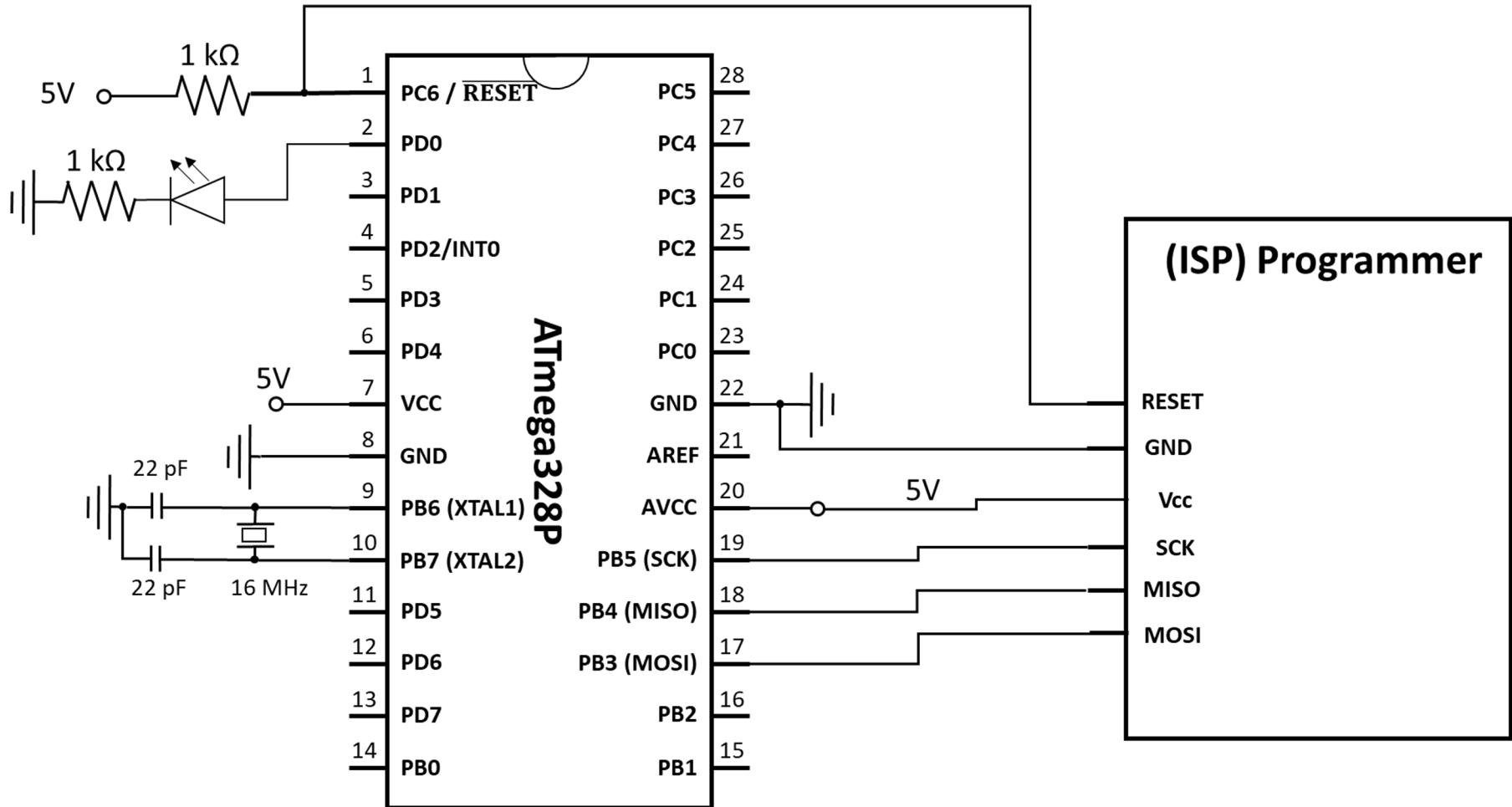


Set Jumper on programmer to 5v

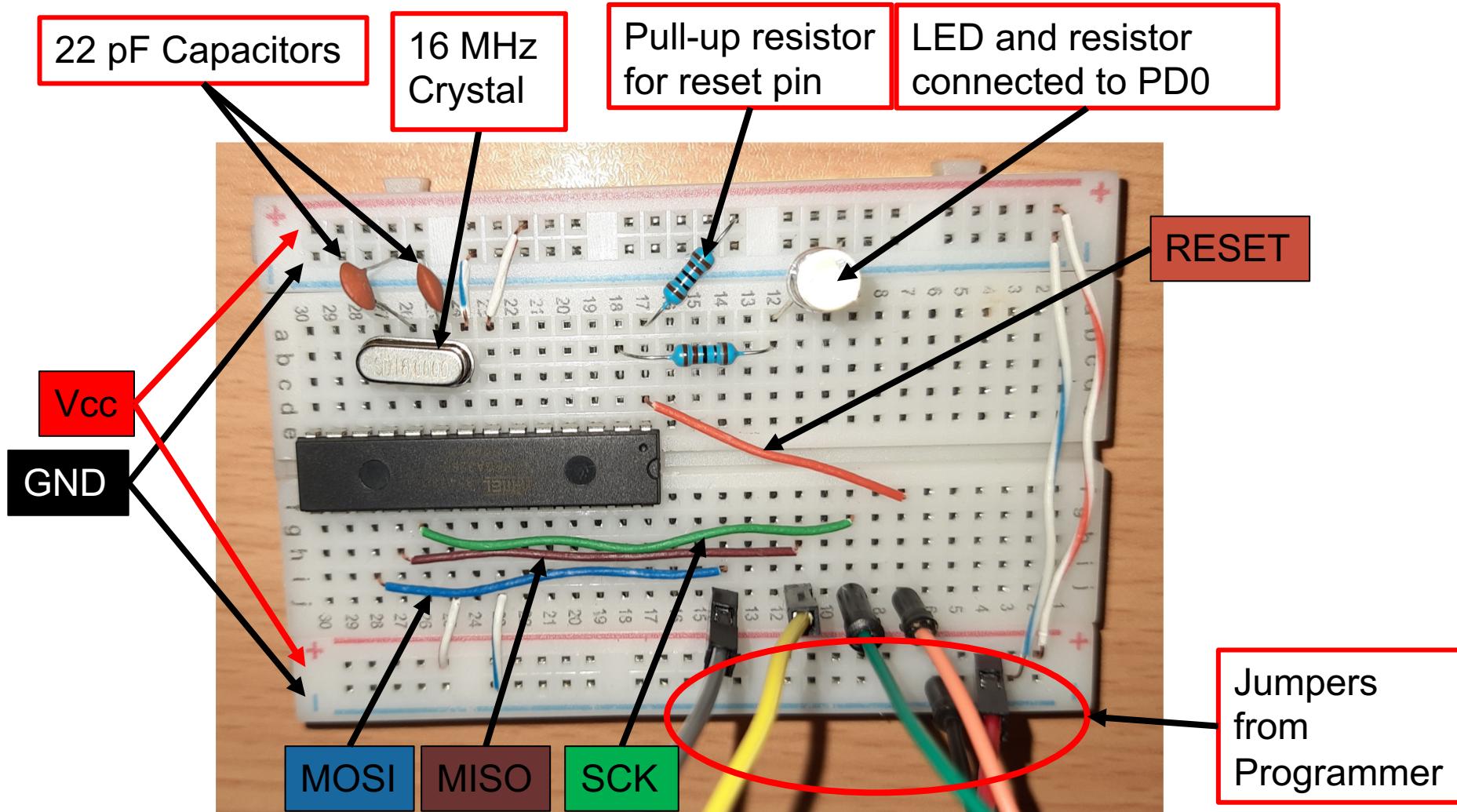


In-System Programming (ISP)

- Schematic diagram of connection between ISP programmer and MCU:



ISP Connection to ATmega328P



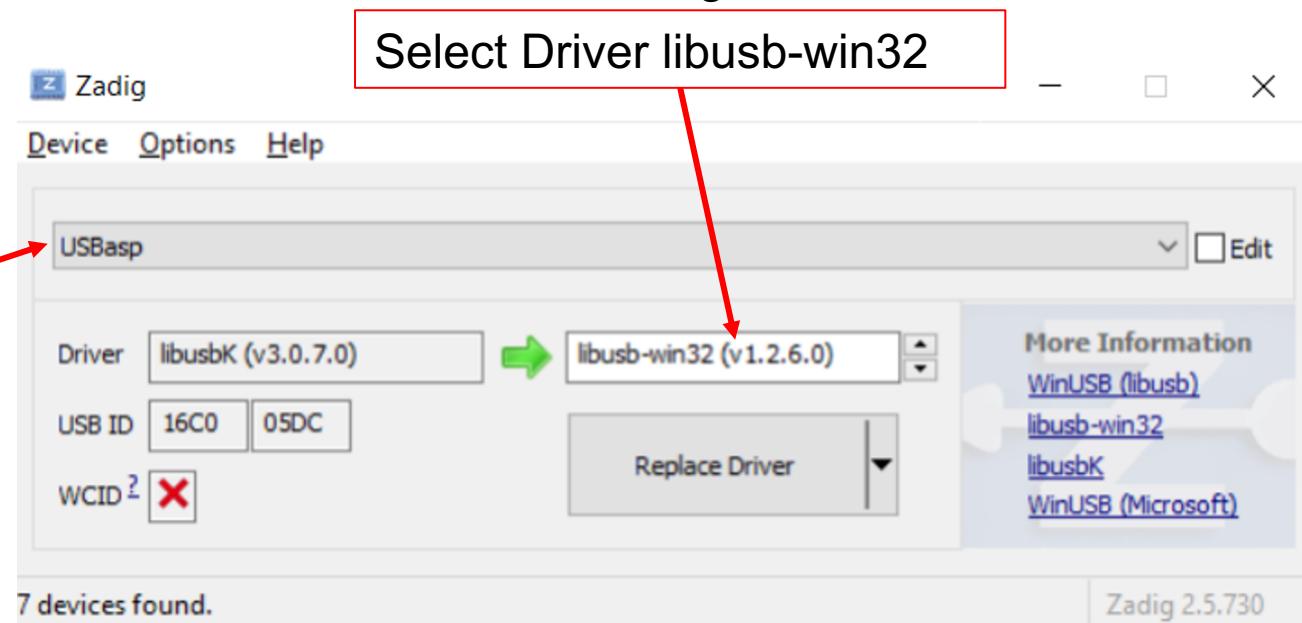
Contents

- **AVR Microcontroller Programming Methods**

- Self-programming
- **In-System Programming (ISP)**
 - ❖ Hardware Connection
 - ❖ **Software Setup**
 - ❖ Fuse Bits Setup
 - ❖ Create Project
 - ❖ Download Program

Setup USBasp Driver

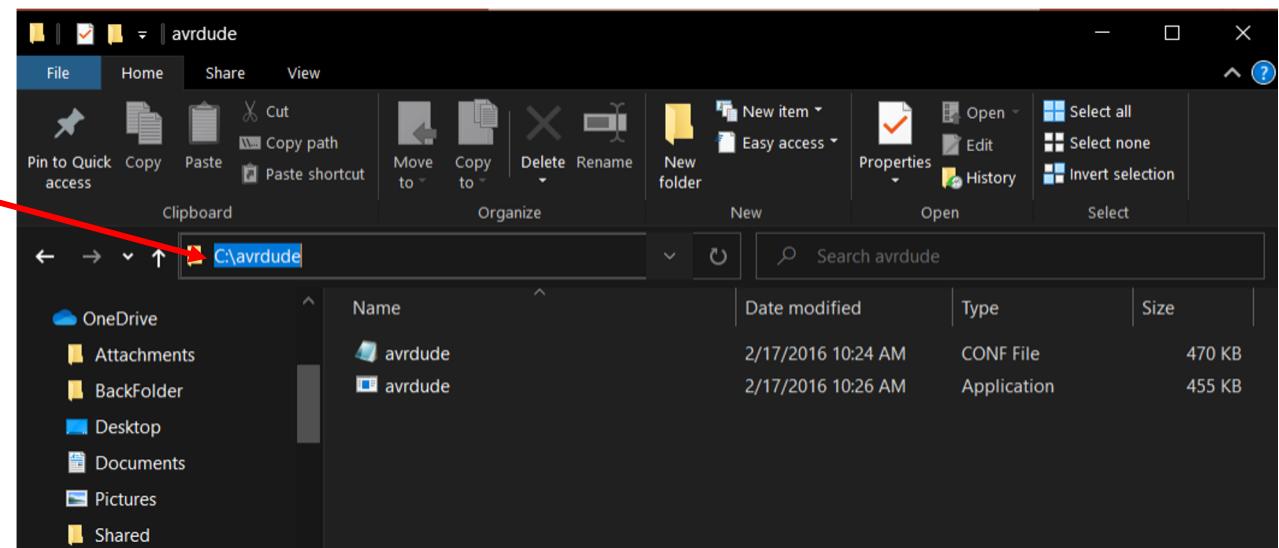
1. Connect ISP programmer (USBasp) to the PC through a USB port.
1. To install the driver, download Zadig from <https://zadig.akeo.ie/>
1. Open Zadig and select USBasp from the dropdown menu. Then set the driver to “libusb-win32” as shown in Figure.



Download

AVRdude

- Download AVRdude tool from:
<http://download.savannah.gnu.org/releases/avrdude/>
- If using windows, download the latest mingw32.zip version.
- Extract the archive and obtain the absolute path to “avrdude.exe” (the path will be required later) as shown.



Absolute path

Contents

- **AVR Microcontroller Programming Methods**

- Self-programming
- **In-System Programming (ISP)**
 - ❖ Hardware Connection
 - ❖ Software Setup
 - ❖ Fuse Bits Setup**
 - ❖ Create Project
 - ❖ Download Program

Set Fuse bits for ATmega328P

- The fuse bits are a group of bits responsible for the configuration of the microcontroller.
- They can be either selected manually from the datasheet or using an AVR fuse calculator such as: <https://www.engbedded.com/fusecalc/>
- Notes:
 - By default, fuse bits for unprogrammed MCUs are set to use an internal oscillator “**Int. RC Osc 8MHz**”. To use an external crystal oscillator the fuse bits must be modified.
 - Make sure that “**Serial program downloading (SPI) enabled**” is ticked. If not enabled, ISP will not be able to program chip.
 - Make sure that “**Reset Disabled (Enable PC6 as i/o pin)**” is **not** ticked. If reset is disabled, programming with SPI cannot be initiated.
 - The last two options allow serial programming of the MCU, otherwise ISP will not work for the programmed chip.

Set Fuse bits for ATmega328P

- Example configuration of the calculator

[Engbedded Atmel AVR® Fuse Calculator](#)

Clock Settings (Ext. Full-swing Crystal) has better performance in noisy environments therefore it is selected.

Device selection

Select the AVR device type you want to configure. When changing this setting, default fuse settings will automatically be applied. Presets (hexadecimal representation of the fuse settings) can be reviewed and even be set in the last form at the bottom of this page.

AVR part name: **ATmega328P** (141 parts currently listed)

Feature configuration

Select AVR Device

This allows easy configuration of your AVR device. All changes will be applied instantly.

Features

Ext. Full-swing Crystal; Start-up time PWRDWN/RESET: 16K CK/14 CK + 65 ms; [CKSEL=0111 SUT=11]

Clock output on PORTB0; [CKOUT=0]

Divide clock by 8 internally; [CKDIV8=0]

Boot Reset vector Enabled (default address=\$0000); [BOOTRST=0]

Boot Flash section size=2048 words Boot start address=\$3800; [BOOTSZ=00] ; default value

Preserve EEPROM memory through the Chip Erase cycle; [EESAVE=0]

Watch-dog Timer always on; [WDTON=0]

Serial program downloading (SPI) enabled; [SPIEN=0]

Debug Wire enable; [DWEN=0]

Reset Disabled (Enable PC6 as i/o pin); [RSTDISBL=0]

Brown-out detection disabled; [BODLEVEL=111]

Press this button after choosing features

Manual fuse bits configuration

Set Fuse bits for ATmega328P

- Copy the AVRdude arguments to use them and set the fuse bits.

arguments

Current settings

These fields show the actual hexadecimal representation of the fuse settings from above. These are the values you have to program into your AVR device. Optionally, you may fill in the numerical values yourself to preset the configuration to these values. Changes in the value fields are applied instantly (taking away the focus)!

Low	High	Extended	Action	AVRDUDE arguments
0x F7	0x D9	0x FF *	<input type="button" value="Apply values"/> <input type="button" value="Defaults"/>	<pre>-U lfuse:w:0xf7:m -U hfuse:w:0xd9:m -U efuse:w:0xff:m</pre> Select (try triple-click) and copy-and-paste this option string into your avrdude command line. You may specify multiple -U arguments within one call of avrdude. * Note that some numerical values refer to fuses containing undefined bits (set to '1' here). Depending on the target device these fuse bits will be read either as '0' or '1'. Verification errors will occur if the values are read back with undefined bits set to '0'. Everything is fine if the values read from the device are either the same as programmed, or the following values (undefined set to '0'): Extended: 0x07.

- Open the command prompt and navigate to the directory of avrdude.

Set Fuse bits for ATmega328P

- While in the avrdude directory, type the following command while USBasp is connected to PC and atmega328p:

avrdude -c usbasp -p m328p <fuse settings copied from calculator>

- Example:

avrdude -c usbasp -p m328p -U lfuse:w:0xf7:m -U hfuse:w:0xd9:m -U efuse:w:0xff:m

- The command prompt should show positive results indicating that the fuse bits were successfully modified.

```
cmd Command Prompt
avrdude: load data hfuse data from input file 0xd9:
avrdude: input file 0xd9 contains 1 bytes
avrdude: reading on-chip hfuse data:

Reading | ##### | 100% 0.01s

avrdude: verifying ...
avrdude: 1 bytes of hfuse verified
avrdude: reading input file "0xff"
avrdude: writing efuse (1 bytes):

Writing | ##### | 100% 0.01s

avrdude: 1 bytes of efuse written
avrdude: verifying efuse memory against 0xff:
avrdude: load data efuse data from input file 0xff:
avrdude: input file 0xff contains 1 bytes
avrdude: reading on-chip efuse data:

Reading | ##### | 100% 0.01s

avrdude: verifying ...
avrdude: 1 bytes of efuse verified

avrdude: safemode: Fuses OK (E:FF, H:D9, L:F7)

avrdude done. Thank you.

C:\avrdude>
```

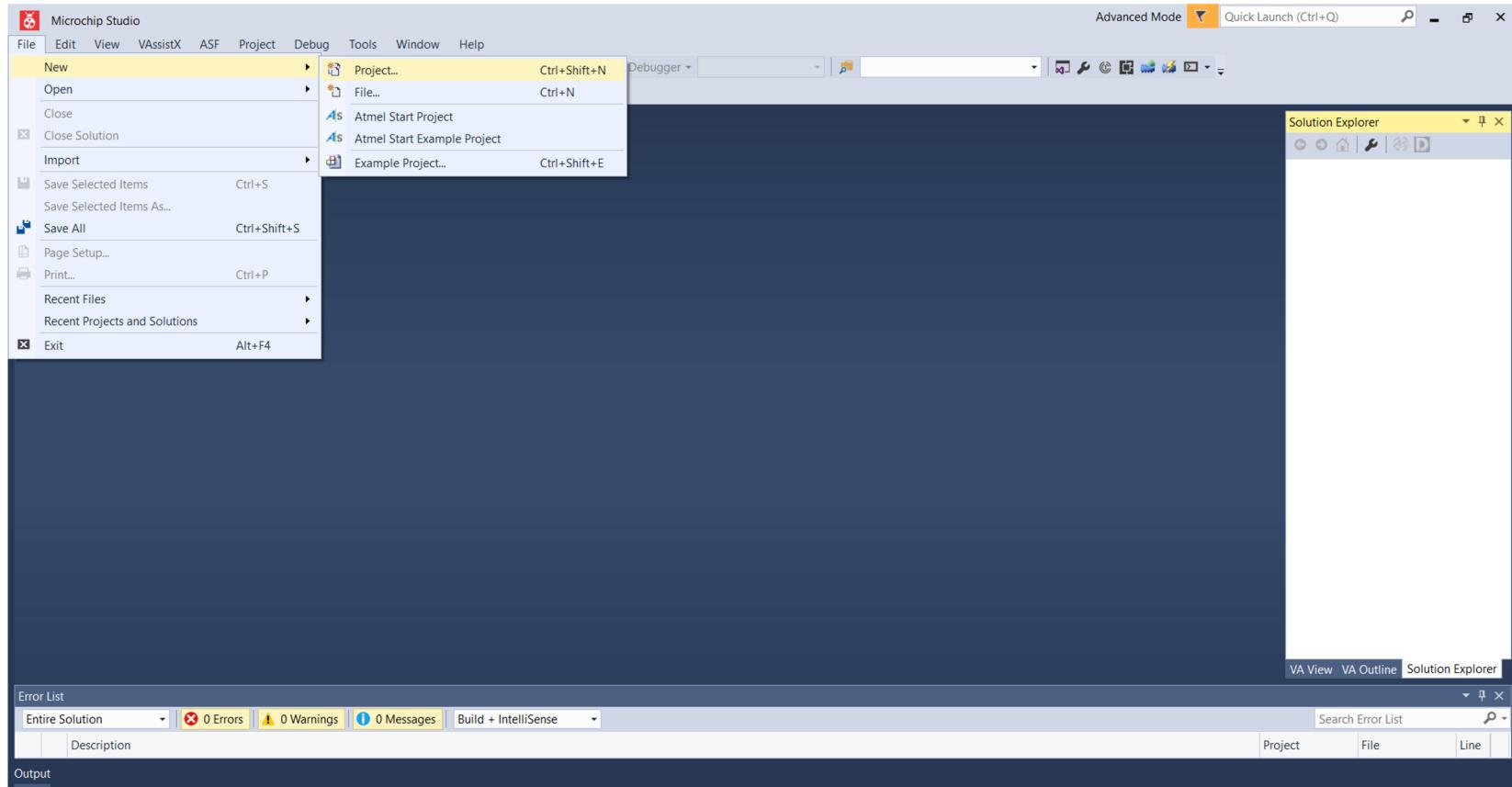
Contents

- **AVR Microcontroller Programming Methods**

- Self-programming
- **In-System Programming (ISP)**
 - ❖ Hardware Connection
 - ❖ Software Installation
 - ❖ Fuse Bits Setup
 - ❖ **Create Project**
 - ❖ Download Program

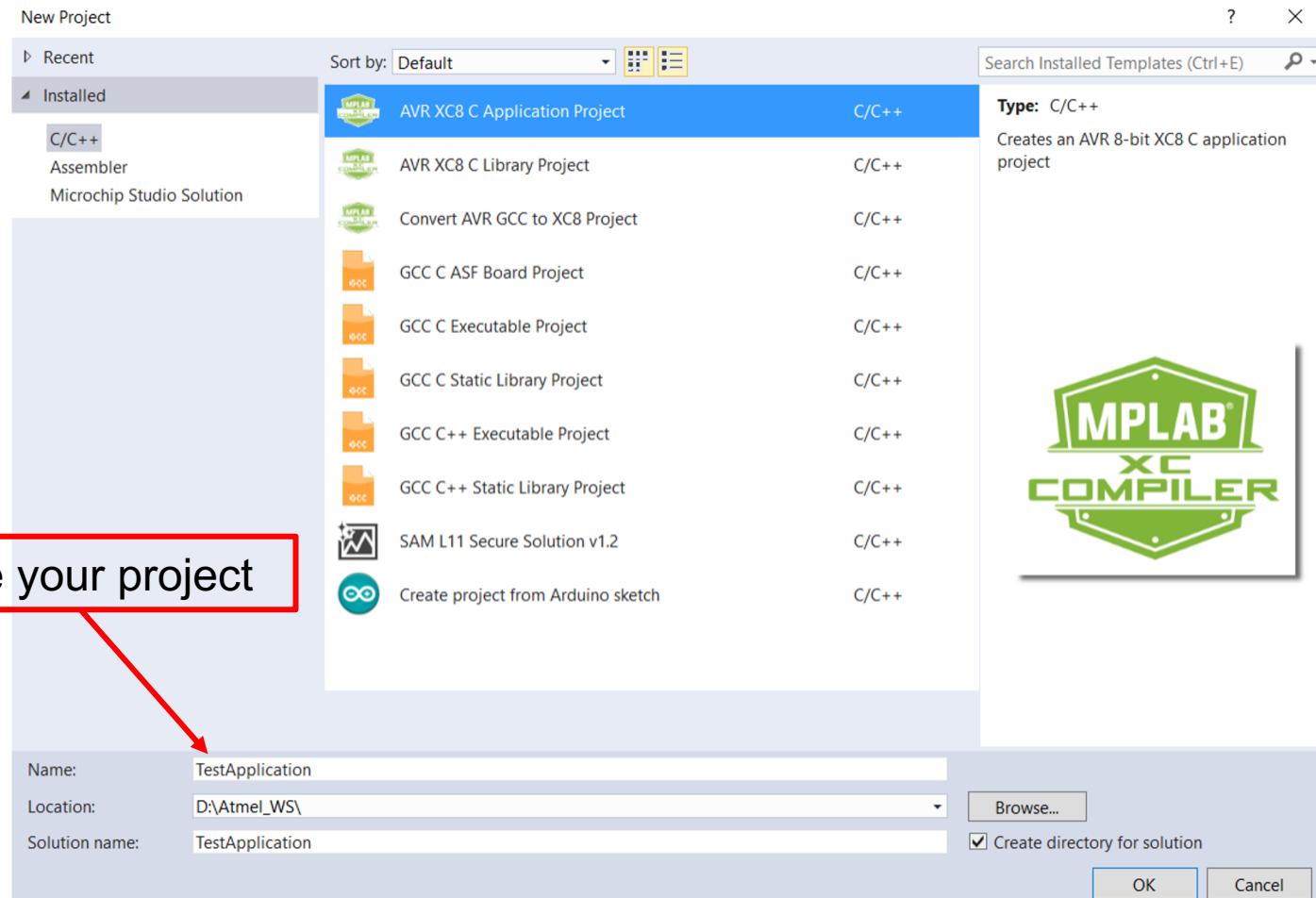
Create New Project in Microchip Studio

- Open Microchip Studio
- Select **File>New>Project**



Create New Project in Microchip Studio

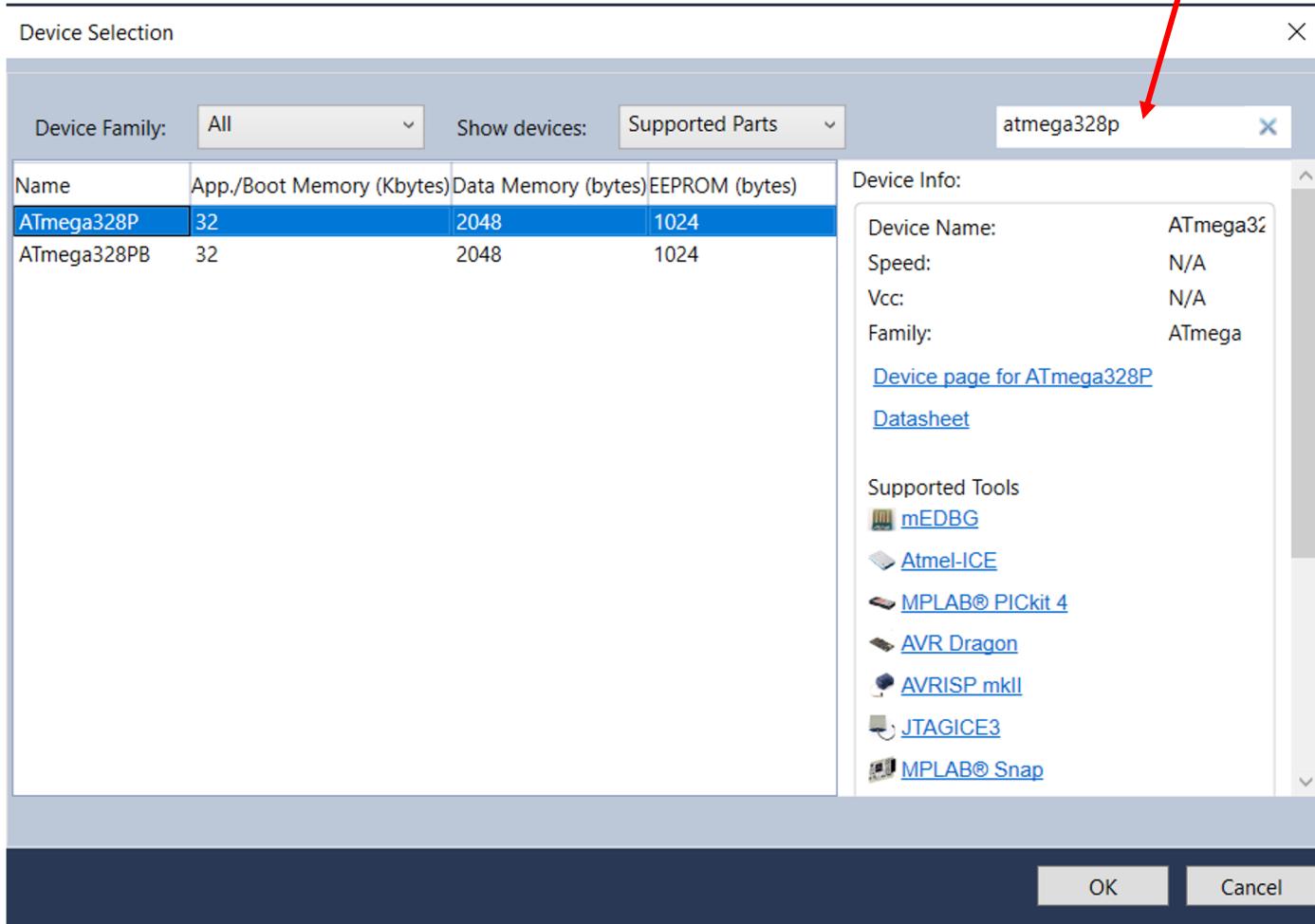
- Select AVR XC8 Application project



Create New Project in Microchip Studio

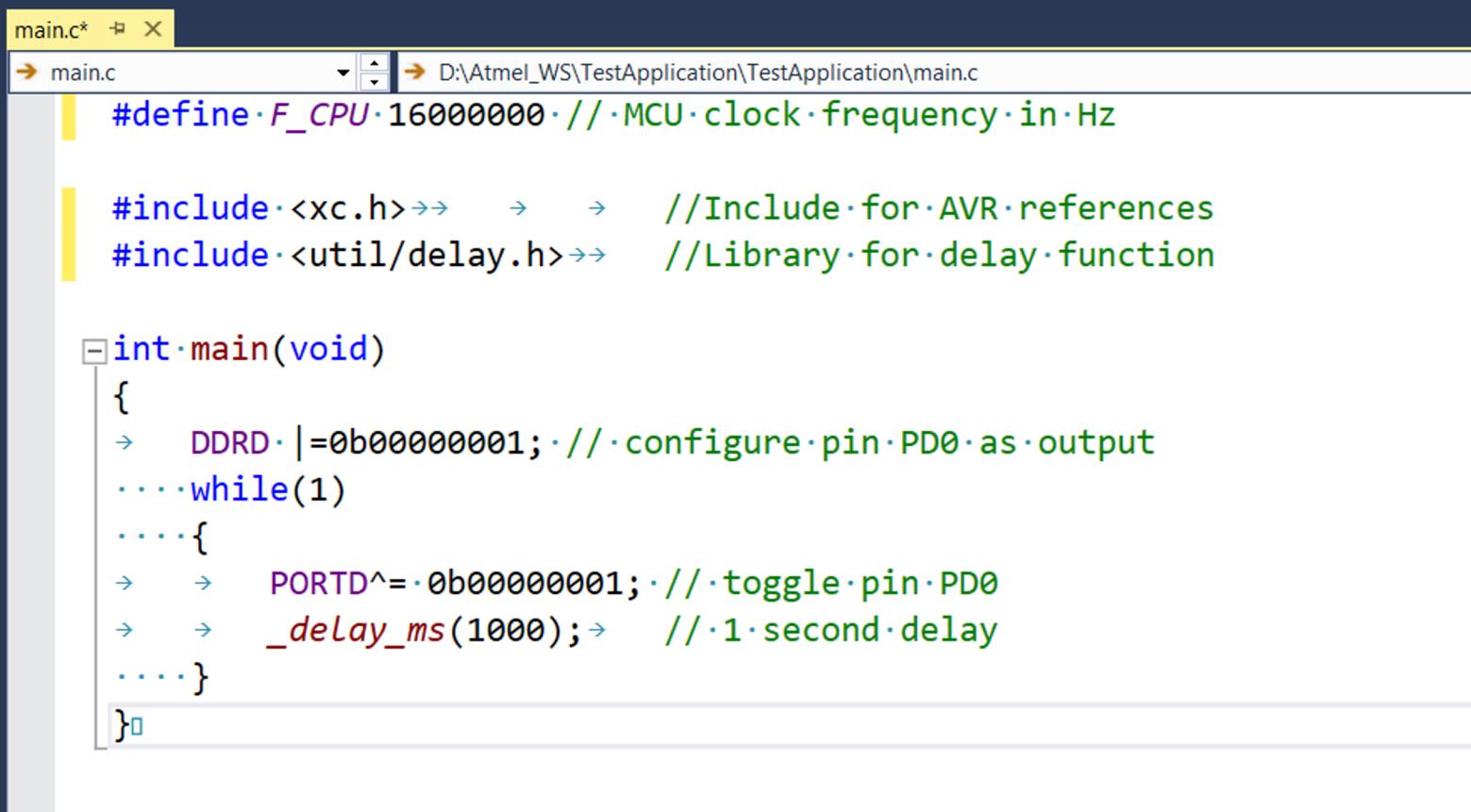
- Search for ATmega328P and select it.

Use this search bar



Create New Project in Microchip Studio

- Write your code in “main.c” (The code shown toggles pin PD0 every 1 second).



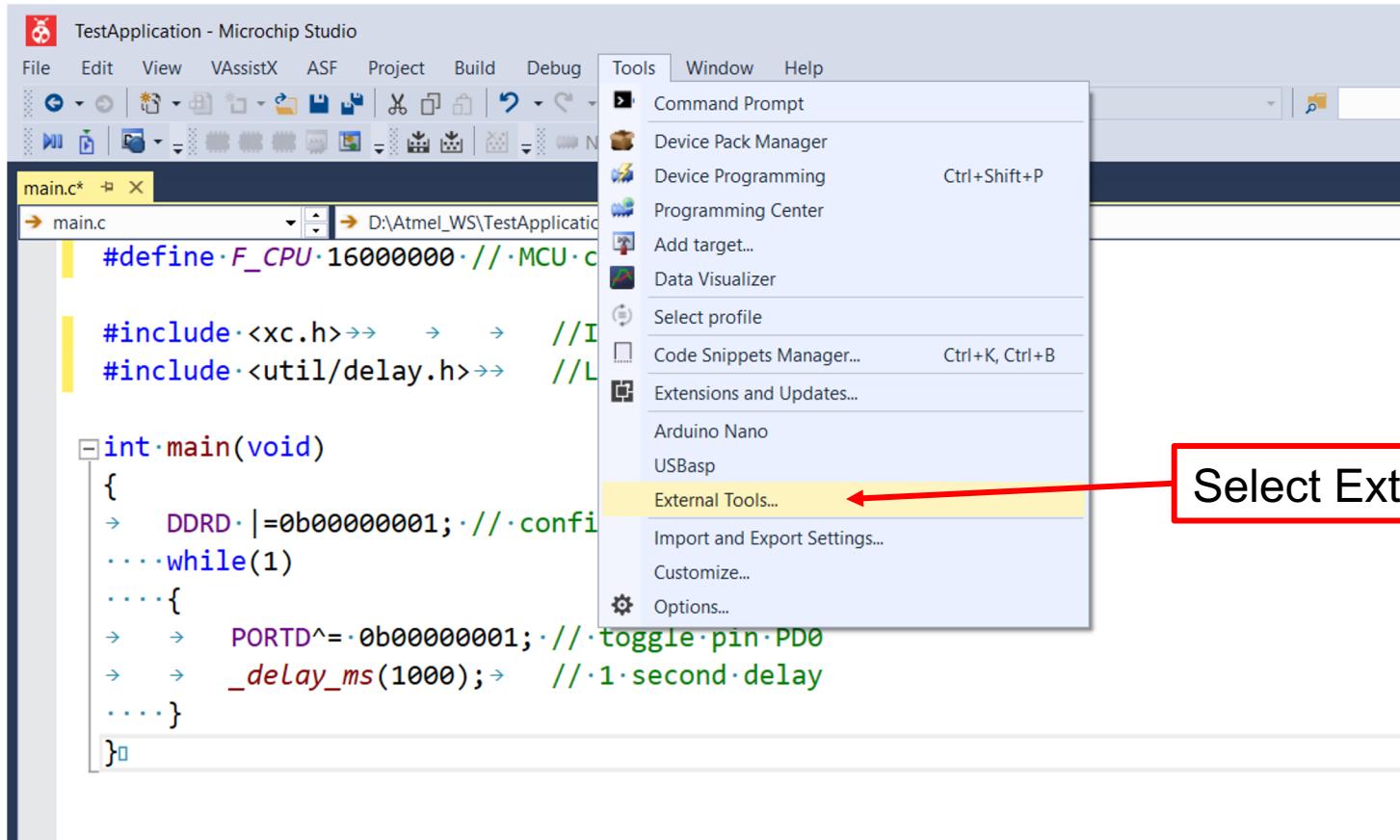
The screenshot shows the Microchip Studio interface with the main.c file open. The code defines a constant for the MCU clock frequency and includes necessary AVR references and a delay library. It then defines a main function that configures pin PD0 as an output, enters a loop, and toggles the pin every 1 second using the _delay_ms(1000) function.

```
#define F_CPU 16000000 // MCU clock frequency in Hz

#include <xc.h> // AVR references
#include <util/delay.h> // Library for delay function

int main(void)
{
    DDRD |= 0b00000001; // configure pin PD0 as output
    while(1)
    {
        PORTD ^= 0b00000001; // toggle pin PD0
        _delay_ms(1000); // 1 second delay
    }
}
```

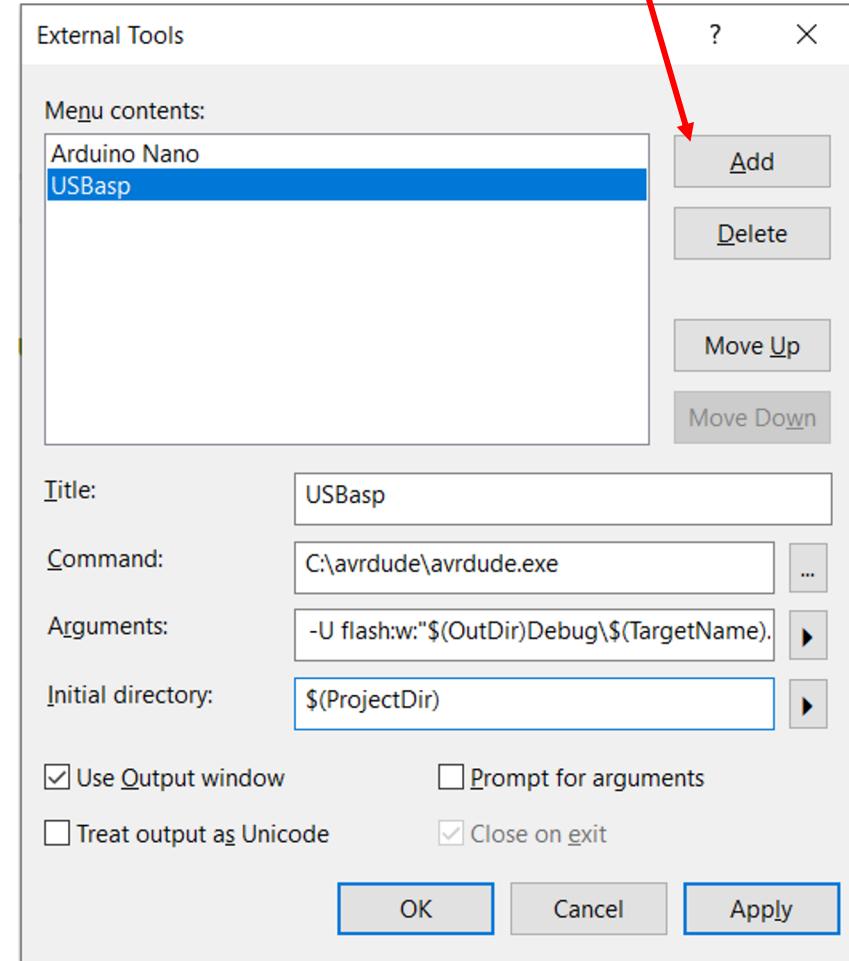
Create USBasp Tool in Microchip Studio



Create USBasp Tool in Microchip Studio

- Click on the “Add” button.
- The tool should be created with the following specifications:
 - Command should be **avrduude.exe** with absolute path. In this example it is “C:\avrdude\avrdude.exe”.
 - The arguments should be as follows:
-c usbasp -p m328p -P usb -U flash:w:"\$(OutDir)Debug\\$(TargetName).hex"
 - Initial Directory: \$(ProjectDir)
 - Mark “Use Output Window”

Add new tool



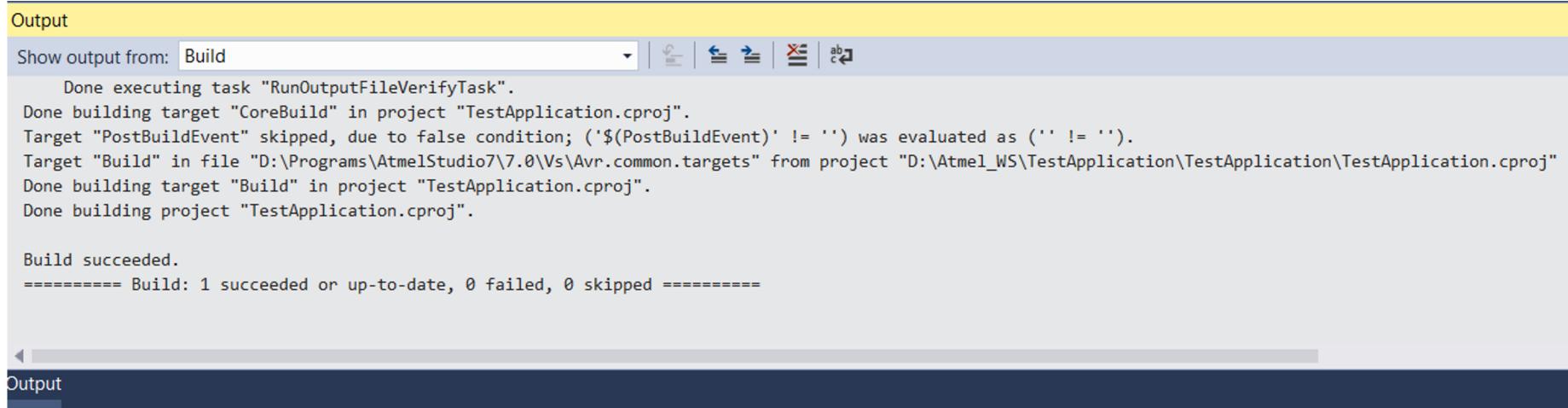
Note: -c argument defines programmer board, while –p defines the AVR device

Contents

- **AVR Microcontroller Programming Methods**
 - Self-programming
 - **In-System Programming (ISP)**
 - ❖ Hardware Connection
 - ❖ Software Setup
 - ❖ Fuse Bits Setup
 - ❖ Create Project
 - ❖ Download Program**

Download Code to MCU

- Build the project from **Build>Build Solution**
- The output window should show “Build Succeeded”.



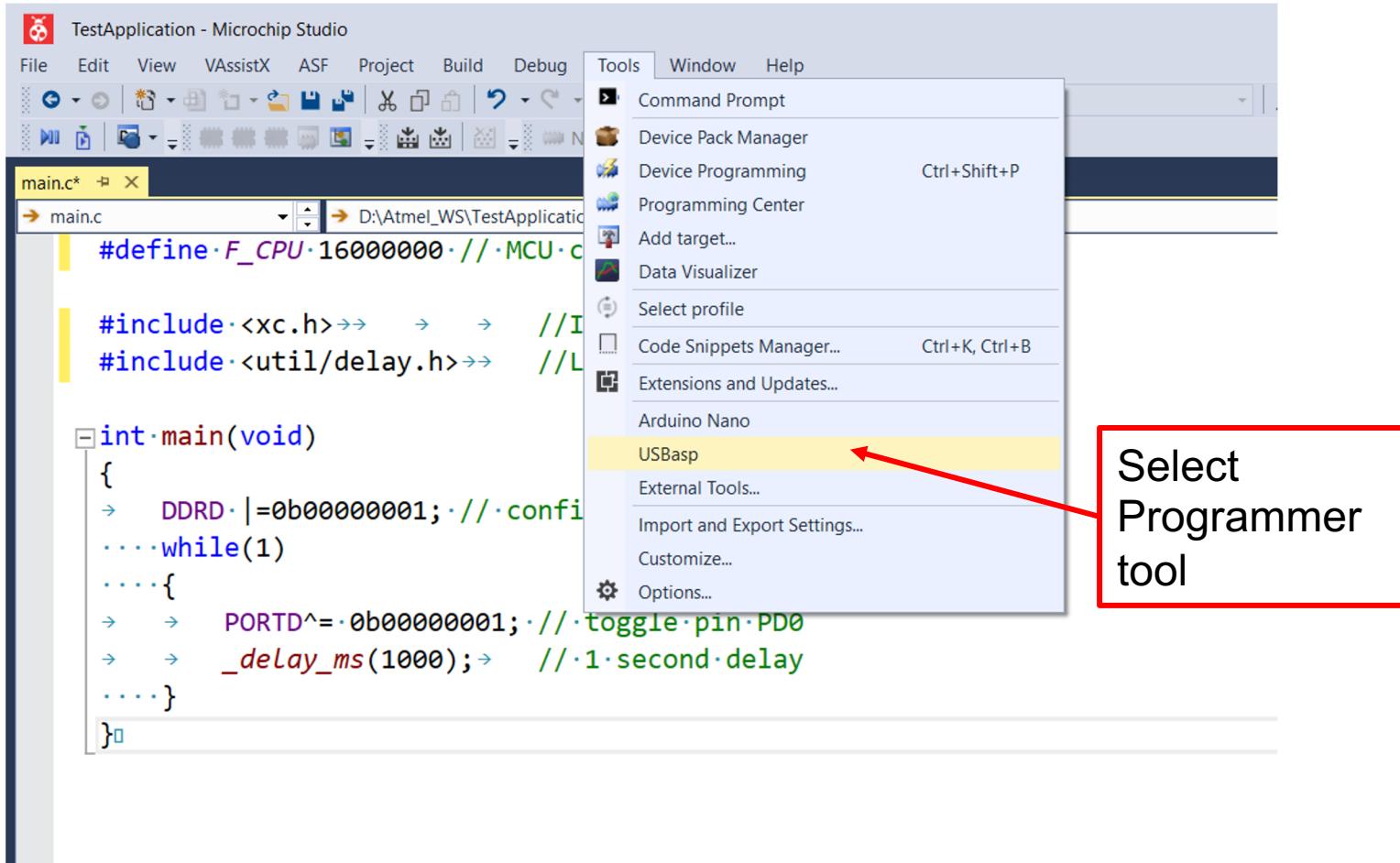
The screenshot shows the 'Output' window of Atmel Studio 7. The title bar says 'Output'. The dropdown menu shows 'Build' is selected. Below the menu are several icons: a magnifying glass, a double arrow, a red X, and a refresh symbol. The main text area displays the following build log:

```
Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "TestApplication.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != ''').
Target "Build" in file "D:\Programs\AtmelStudio7\7.0\Vs\Avr.common.targets" from project "D:\Atmel_WS\TestApplication\TestApplication.cproj"
Done building target "Build" in project "TestApplication.cproj".
Done building project "TestApplication.cproj".

Build succeeded.
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped =====
```

Download Code to MCU

- To download the code, select the external tool created for USBasp:



Download Code to MCU

- Finally, check the output window to verify that the MCU has been programmed successfully.
- Now the LED connected to PD0 should be blinking as instructed in the code.

```
Output
Show output from: USBasp
Reading | #####| 100% 0.14s
avrdude.exe: verifying ...
avrdude.exe: 146 bytes of flash verified
avrdude.exe: safemode: Fuses OK (E:FF, H:D9, L:62)
avrdude.exe done.  Thank you.
```