Namal University

Mianwali

Department of Electrical Engineering

EE-252L: Introduction to Embedded Systems

Lab Manual: 03

Atmega328p AVR Branch & Call Instruction

Students Name	
Roll Number	
Submission Date	
Marks Obtained	

Instructors: Engr. Faizan e Mustafa / Dr. Hamza Zad Gul

Introduction

Conditional branch instructions are the set of instructions that is used to branch out of a loop. We will discuss these instructions for the AVR micro-controller.

To understand these instructions, first, we need to know about the registers in the AVR micro-controller.

Status Register (SReg):

- It is the flag register in the AVR micro-controller.
- It is a 8 – bit register. Only 6 of these 8 bits are called conditional flags. They are C, Z, N, V, S, H.
- The value of these bits indicates some conditions that result after the execution of an instruction.

1. C – Carry Flag:
   - This flag is set whenever there is a carryout from the D7 bit.
   - This flag bit is affected after an 8-bit addition or subtraction.
2. Z – Zero Flag:
   - The zero flag reflects the result of an arithmetic or logic operation.
   - If the result is zero, then Z = 1. Therefore, Z = 0 if the result is not zero.
3. N – Negative Flag:
   - Binary representation of signed numbers uses D7 as the sign bit.
   - The negative flag reflects the result of an arithmetic operation. If the D7 bit of the result is zero.
   - Then N = 0 and the result is positive. If the D7 bit is one, then N = 1 and the result is negative.
4. V – Overflow Flag:
   - This flag is set whenever the result of a signed number operation is too large.
   - Causing the high-order bit to overflow into the sign bit.
5. S – Sign Flag:
   - This flag is the result of Exclusive-ORing of N and V flags.
6. H – Half Carry Flag:
   - If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set; otherwise, it is cleared.
   - This flag bit is used by instructions that perform BCD (binary coded decimal) arithmetic.

The following table shows a few branch loop instructions:

Instruction	**Explanation**	**Flag Status**
BREQ	Branch if equal	Branch if Z = 1
BRNE	Branch if not equal	Branch if Z = 0
BRSH	Branch if same or higher	Branch if C = 0
BRLO	Branch if lower	Branch if C = 1
BRLT	Branch if less than (signed)	Branch if S = 1
BRGE	Branch if greater than or equal (signed)	Branch if S = 0
BRVS	Branch if Overflow flag set	Branch if V = 1
BRVC	Branch if Overflow flag clear	Branch if V = 0

Unconditional Branch Instructions:

The unconditional branch is a jump in which control is transferred unconditionally to the target address. In AVR, there are 3 unconditional branch instructions: JMP, RJMP, and IJMP. Using which instruction depends upon the target address.

1. **JMP (long jump) –**

JMP is an unconditional jump that can go to any memory location in the 4M (word) address space of the AVR. It is a 4-byte instruction in which 10 bits are used for the opcode, and the other 22 bits represent the 22-bit address of the target location.

2. **RJMP (relative jump) –**

In this 2-byte instruction, the first 4 bits are used for the opcode and the rest of the bits are used for the relative address of the target location. The relative address range of 000-$FFF is divided into forward and backward jumps, that is within -2048 to +2047 of memory relative to the address of the current program counter.

3. **IJMP (indirect jump) –**

It is a 2-byte instruction. When it executes, the program counter is loaded with the contents of the Z register, so it jumps to the address provided by the Z register. IJMP can jump within the lowest 64K words of the program memory.

CALL:

CALL is a control transfer instruction that is used to call a particular subroutine. A subroutine is a block of instructions that need to be performed frequently.

In AVR, there are 4 instructions for the call subroutine as following.

1. **CALL (call subroutine)**
2. RCALL (relative call subroutine)
3. ICALL (indirect call to Z)
4. EICALL (extended indirect call to Z)

CALL:

In this 4-byte instruction, 10 bits are used for the opcode and the other 22 bits are used for the address of the target subroutine just as in the JMP instruction. In this, 4M address space of 000000-$3FFFFF for AVR and can be used to call subroutines within the given range of address. To make sure that the AVR knows where to come back after the execution of the subroutine, the microcontroller automatically saves the address of the instruction just below the CALL instruction on the stack. After finishing the execution of the subroutine, the RET instruction transfers control back to the caller. Hence, every subroutine has a RET instruction at the end.

CALL instruction, RET instruction and the role of stack:

When the CALL instruction is executed, the address of the instruction below the CALL instruction is pushed onto the stack. When the execution of that subroutine is finished and RET is executed, the address of the instruction below the CALL instruction is loaded in the program counter and it is executed.

Objectives

In this lab, student will learn to configure programmer and Microchip studio to program atmega328p. They will also

learn to program control LEDs using Branch and Call instructions

Course Learning Outcomes

CLO1: Practice the correct use of programming constructs of assembly language
CLO2: Conduct hardware interfacing of AVR microprocessors.

Software

- Microchip studio

Hardware

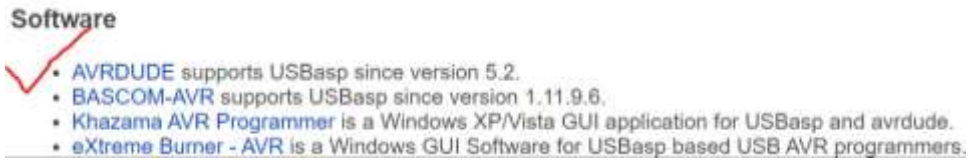
- Atmega 328p
- USBasp programmer
- Atmega328p USBasp programmer Board
- Breadboard
- Connecting wires
- LEDs
- Resistors
- Capacitors
- Crystal oscillator

Instructions

- You must submit the required files before the given deadline in **PDF format**.
- Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom.
- Multiple such incidents will result in disciplinary action being taken.

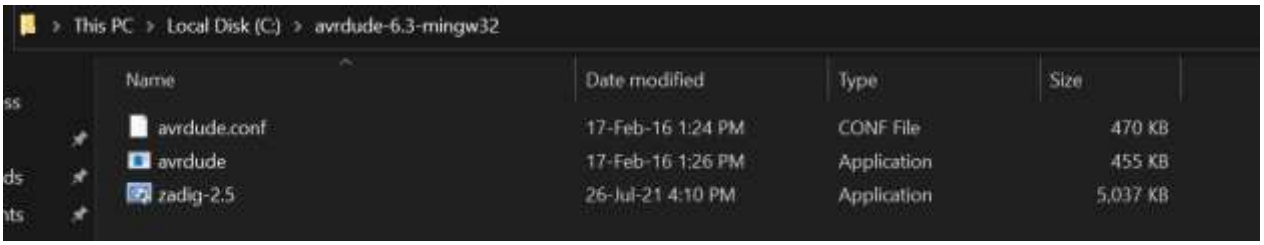
Lab Task 1: Installing the AVRDUDE and USBasp drivers

1. Visit the USBasp official website [USBasp - USB programmer for Atmel AVR controllers - fischl.de (https://www.fischl.de/usbasp/)](https://www.fischl.de/usbasp/)
2. Scroll down, find the software section, and click the avrdude software.



3. Download the latest version of software zip file.

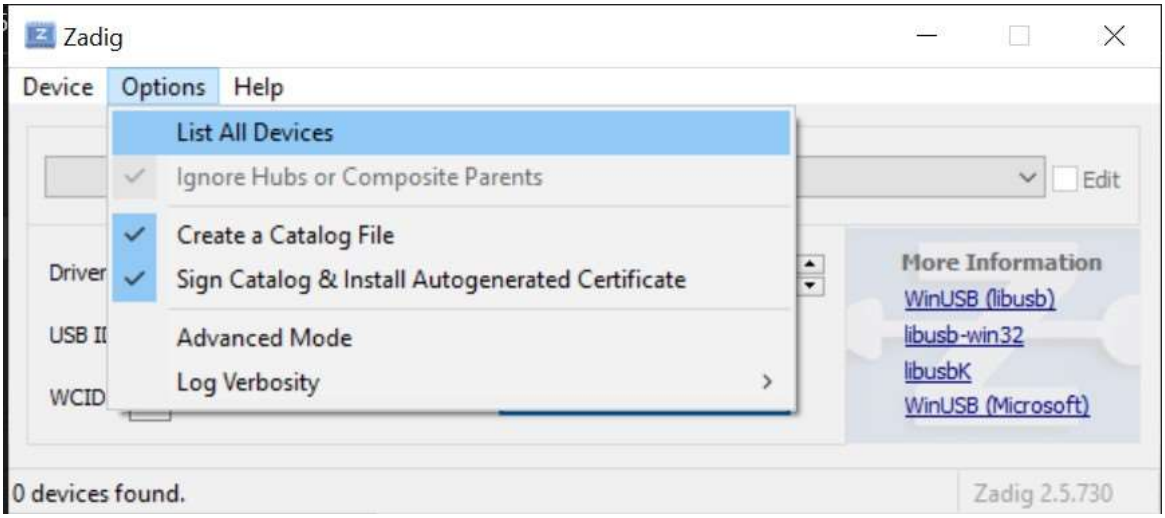
4. Extract the zip file, make a folder of same name in C drive and copy these files to it.



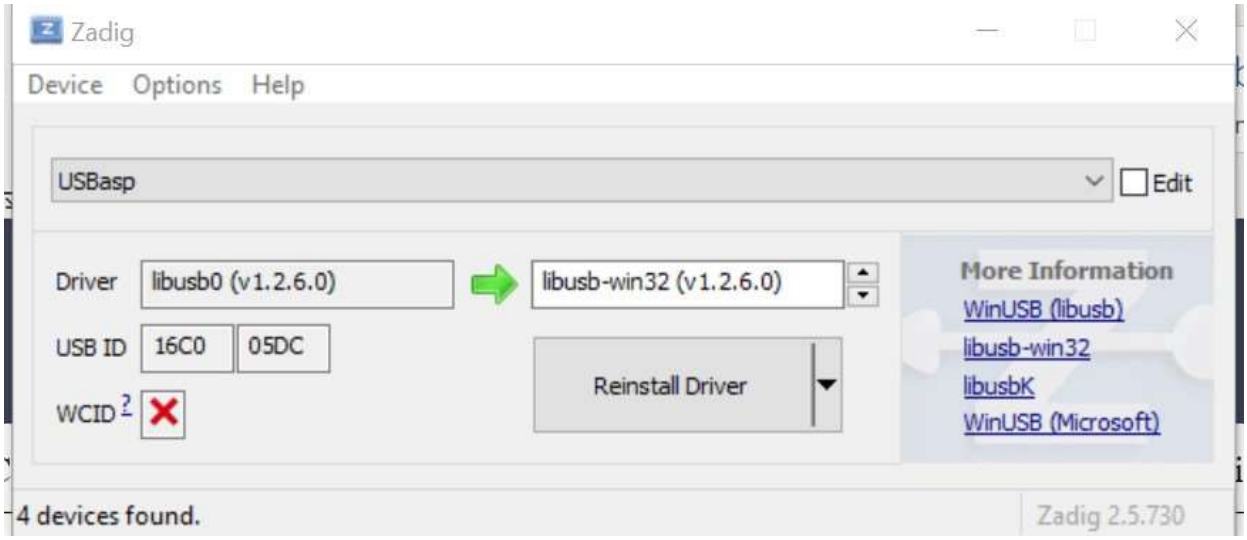
5. Now got to home page of USBasp website and this time click on driver installation

6. Download the driver installation tool and copy it in the above mentioned folder.

7. Connect USBasp to your computer, run driver installer and select the following options.

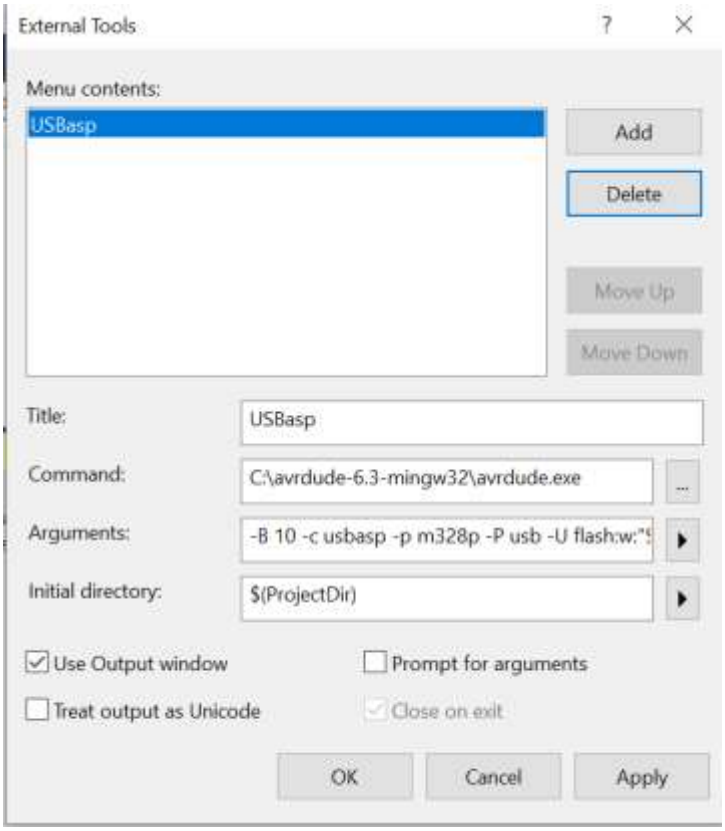


8. After that select USBasp from dropdown and select **libusb-win32 (v1.2.6.0)** driver and clickinstall or replace button.



Lab Task 2: Setting up the Environment in Microchip studio

1. Open the microchip studio and go to tools →External Tools.
2. In external tools window click add and give following commands.

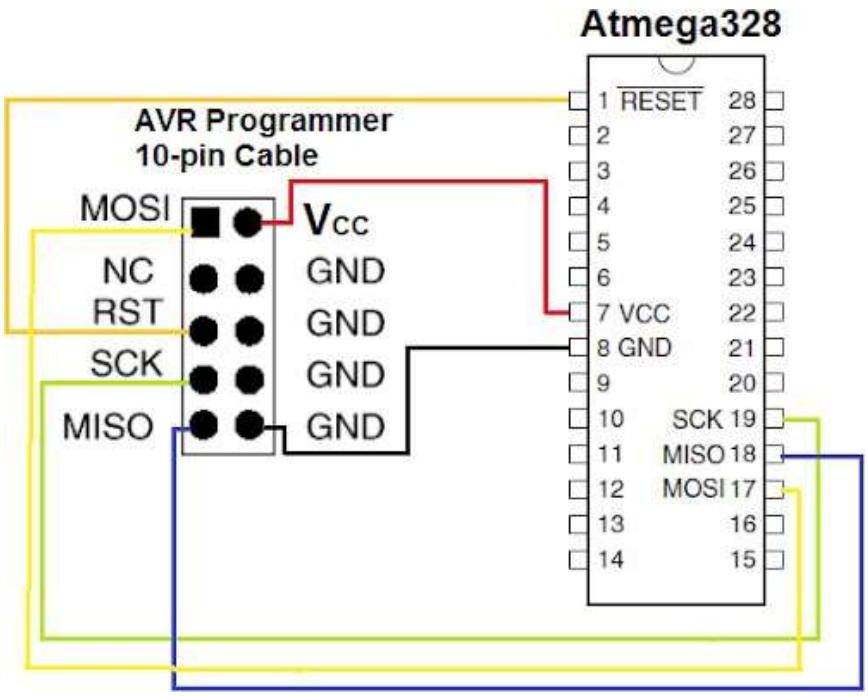


Label	**Values**
Title	USBasp or Any name you like
Commands	C:\avrdude-6.3-mingw32\avrdude.exe
Arguments	-B 10 -c usbasp -p m328p -P usb -U flash:w:"$(OutDir)Debug\$(TargetName).hex"
Initial directory	$(ProjectDir)

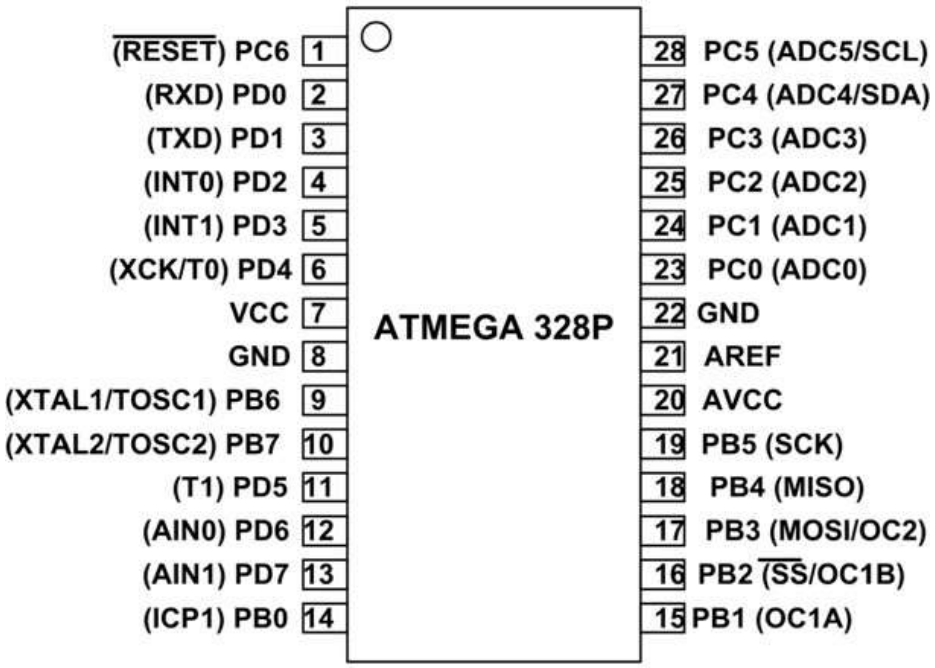
Lab Task 3: Blinking the LED

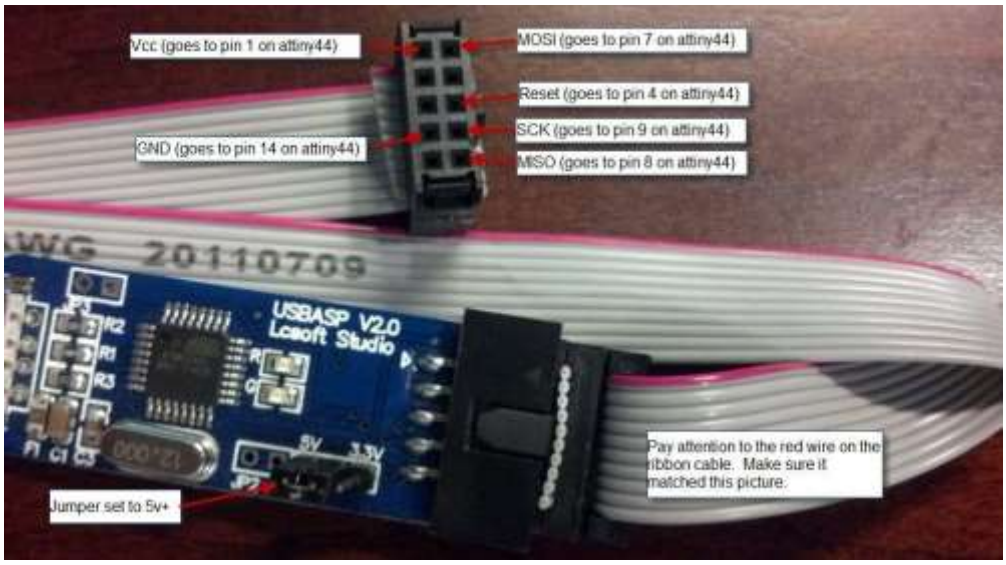
1. Use your knowledge in previous lab to write a code for blink LED

2. Copy this code to your project.
3. For uploading the code to atmega328p build the following circuit.

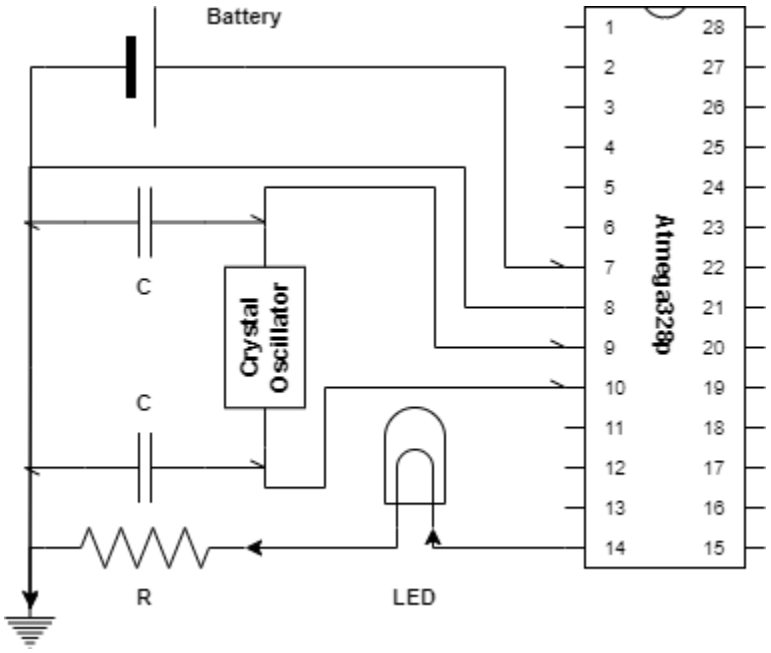


Hint: Consult the following pinout diagrams to make circuit. Also connect a 16MHz crystaloscillator at pin9 and 10 of atmega328p with two 22pF cap.





4. Now go to Microchip studio →Tools → USBasp and it will program the atmega328p.
5. This will upload the code to chip.
6. If you are using programming board provided, then remove the chip from it and make following circuit on a breadboard.



7. Write down your code and calculation for 1 second delay (Connect LED to PORTB0). Explain each command briefly with comments.

8. Write a program in which any number is stored in register R21 between 0 - 16. Divide this number by 4. Use PortB to show the quotient and PortD to show remainder through by turning ON LEDs. Use knowledge gained in previous lab to build the circuit and implement it on breadboard. Draw your circuit (label all the components in your circuit) and write your code below with comments.

Answer the following questions:

1. When the CALL instruction is used how does the microprocessor know to where it should return after completion of the subroutine?

2. What is the meaning of BRANCH PENALTY?

3. What is the different between CONDITIONAL and UNCONDITIONAL jump in AVR?

Introduction to Embedded System Lab Rubrics

- **Method of Evaluation** Viva Conducted during lab and lab reports submitted by students

Assessment tool/ weightage/ (CLO, PLO)	**Excellent (10 - 9)**	**Good (8 – 7)**	**Satisfactory (6 – 4)**	**Unsatisfactory (3 – 1)**	**Poor 0**	**Marks Obtained**
Programming (CLO1, PLO5)	Correct Code. Easy to understand with proper comments	Correct Code but without proper indentation or comments	Slightly incorrect code with proper comments	Incorrect code with improper format and no comments	Code not submitted	
Circuit Design (CLO2: PLO3)	Circuit is simulated/implemented correctly without any errors	Circuit is simulated but implemented with minor errors	Circuit is simulated & implementation both have errors	Circuit is simulated & implemented however some components are missing/incorrect value	Circuit is simulated/implemented does not work	
Individual/ Teamwork (CLO3:PLO9)	The student/s worked effectively throughout lab to perform the assigned tasks	The student/s performed all the assigned lab tasks however one member took lead	The student/s completed all tasks however failed to work effectively	The student/s attempted all the tasks however the one member did most of the work	The student/s did not work together/at all	
Lab Report (CLO4:PLO10)	The student was able to effectively answer all questions regarding performed tasks and report provides all information without mistakes	The student was able to effectively answer all questions regarding performed tasks however the report has minor mistakes	The student was able to answer most questions regarding performed tasks and information in report is not communicated effectively	The student was able to answer some questions regarding performed tasks and report is confusing and misleading	The student was not able to answer questions regarding performed tasks and report information is incorrect/irrelevant	
Total						