Electrical and Computer Engineering

EE3613 Processor Systems Laboratory

LAB 6:

Atmel AVR Microcontroller Hardware Programming And ATmega324PB Xplained Pro

Fall 2019

OBJECTIVES:

- To learn how to use AVR microcontroller hardware, ATmega324PB Xplained Pro
- To learn how to program AVR microcontroller hardware and debug it using Atmel Studio7
- To learn how to read the output signal from AVR microcontroller hardware IO pin
- To learn how to use the oscilloscope to measure the signal

REFERENCES:

Mazidi and Naimi, "The AVR Microcontroller and Embedded Systems," 2nd Ed. Chapters 1 and 2.

MATERIALS:

Atmel Studio 7

MAP OF THIS LAB:

- Activity 1: Learn the basic structure of the AVR microcontroller board and peripherals that are connected to the controller board's ports
- Activity 2: Program and debug the board using Atmel Studio 7
- Activity 3: Use an oscilloscope to measure the output signal

LAB REPORT INSTRUCTIONS:

This lab consists of three activities. You must show the results of the port output (Activity 2 and 3) to the lab instructor for the lab grade during the lab time. **Any written lab report is not required.**

Total points for this lab: 50 points (Activity 2 - 20 points, Activity 3 – 30 points)

Check-out duration: In the Lab time – Tuesday (10/7/2019) 3-5pm, Thursday (10/9/2019) 9-11am, (after the time duration, you will get 10 points deduction from the total)

Activity 1. Explore the AVR microcontroller, ATmega324PB Xplained Pro

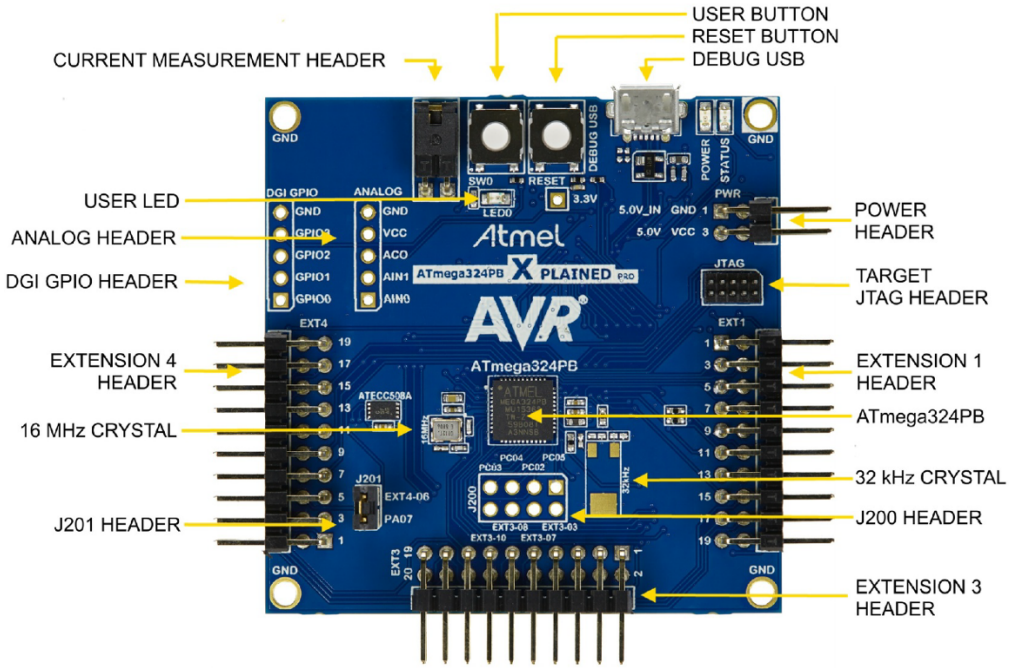


Figure 1: ATmega324PB Xplained Pro Board

ATmega324PB Xplained Pro Quick Start User Guide:

Steps to start exploring the Atmel Xplained Pro platform:

1. Atmel Studio should be installed on your computer.
2. Launch Atmel Studio software.
3. Connect the DEBUG USB port on the kit to the PC using provided USB cable (Standard micro USB cable).

When the Xplained Pro MCU kit will be connected to the computer for the first time, the operating system will install the driver software. The driver file supports both 32- and 64-bit versions of Microsoft® Windows® XP, Windows Vista®, Windows 7, Windows 8, Windows 10, and Windows Server 2012.

When the Xplained Pro MCU board is powered, the green power LED will be ON, and Atmel Studio will auto detect the specific Xplained Pro MCU- and supported extension board(s) that are connected. Atmel Studio will present relevant information such as datasheets and kit documentation. The kit landing page in Atmel Studio also has an option to launch Atmel Software Framework (ASF) example applications for the kit. The ATmega324PB device is programmed and debugged by the on-board Embedded Debugger and therefore no external programmer or debugger tool is required.

The ATmega324PB Xplained Pro board has:

- Atmel ATmega324PB microcontroller
- Embedded debugger (EDBG)

  - USB interface
  - Programming and debugging (target) through JTAG
  - Virtual COM-port interface to target via UART
  - Atmel Data Gateway Interface (DGI) to target via synchronous SPI or TWI
  - Four GPIOs connected to target for code instrumentation
- Digital I/O
  - Two mechanical buttons (user and reset button)
  - One user LED
  - Three extension headers
- Two possible power sources
  - External power
  - Embedded debugger USB
- 16MHz crystal
- 32kHz crystal footprint

Additional Support Information:

http://www.microchip.com/developmenttools/ProductDetails/ATMEGA324PB-XPRO

Extension Board Information:

The Atmel ATmega324PB Xplained Pro Extension Kit is a hardware platform made to evaluate the I/O peripherals of Atmel ATmega324PB development board. The kit offers a set of features that enables the ATmega324PB user to get started using the ATmega324PB peripherals right away and to get an understanding of how to integrate the device in their own design.

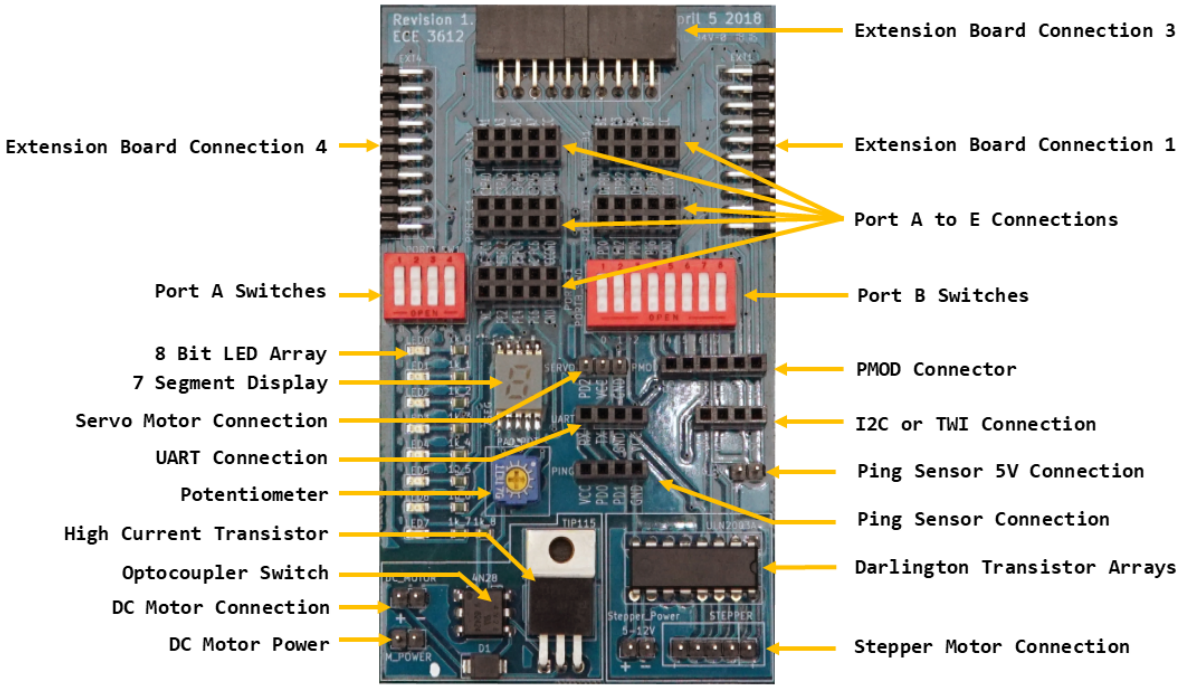


Figure 2: ATmega324PB Xplained Pro Extension Kit

The extension board comes with some common and basic components necessary for our lab assignments.

List of components:

PORT A

1. Analog Input (ADC0-7)
2. LED Output
3. Seven Segment Display

PORT B

1. DIP Switch
2. PMOD Interface

PORT C

1. I2C Interface

PORT D

1. UART Interface
2. Servo Motor
3. PMOD Interface
4. PING Sensor

PORT E

1. DC Motor
2. Stepper Motor

PORT ASSIGNMENTS:

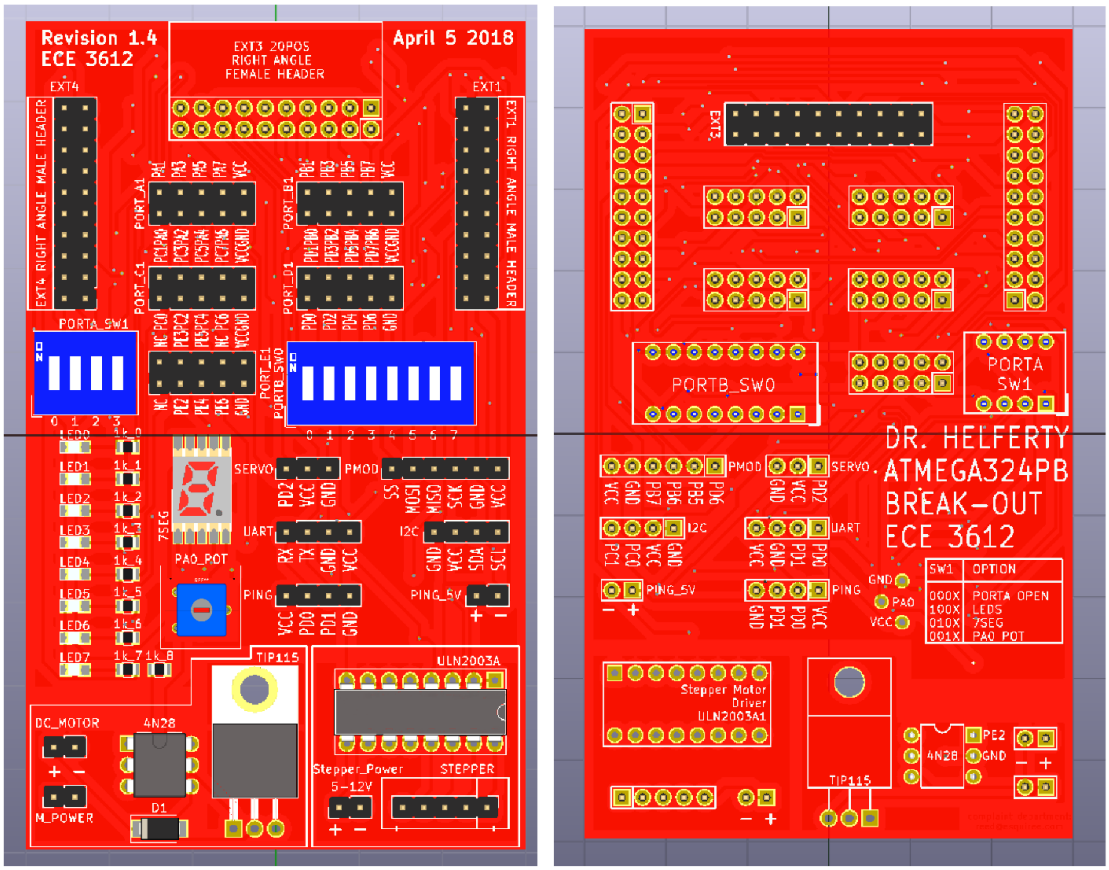
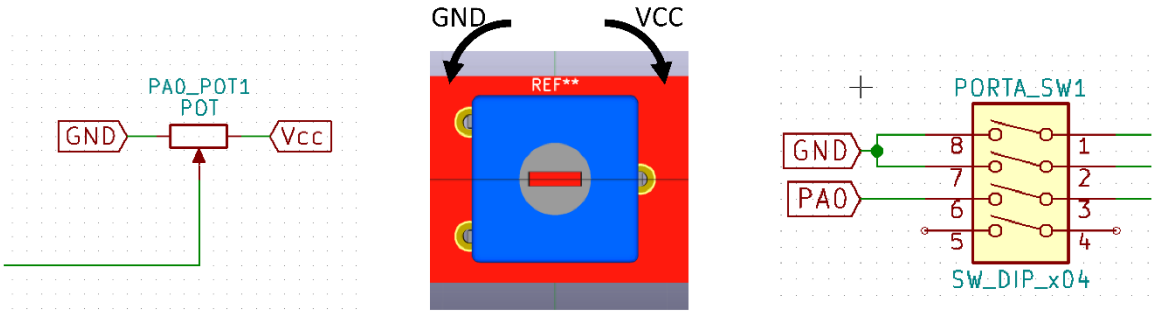


Figure 3: ATmega324PB Xplained Pro Extension Kit (Front and Back)

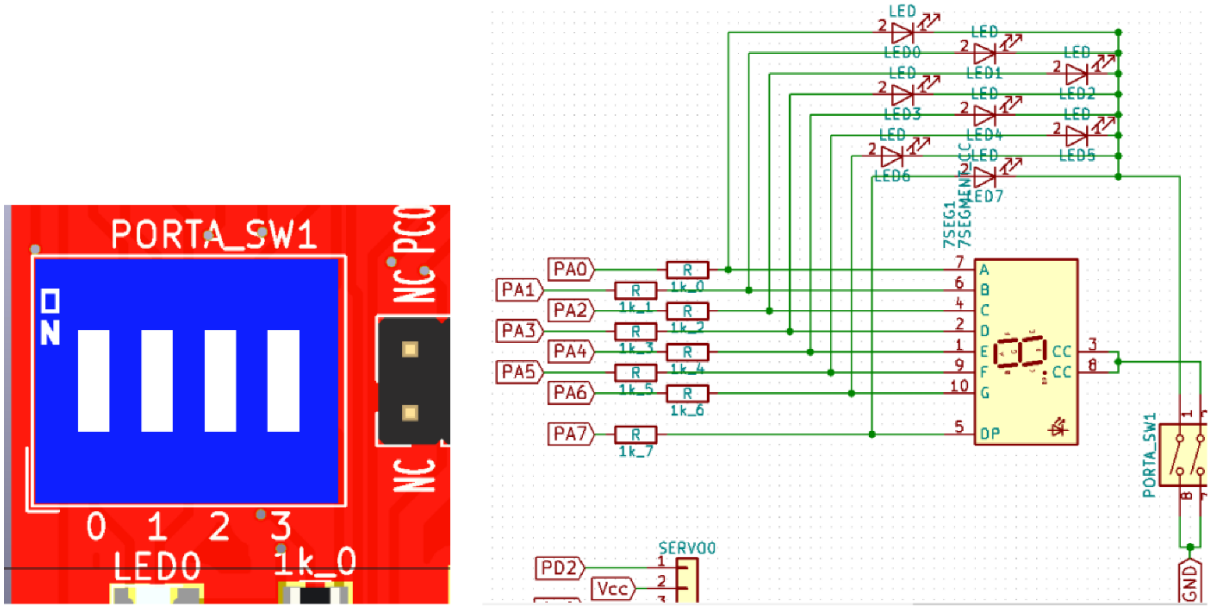
PORT A

PORT A has three purposes:

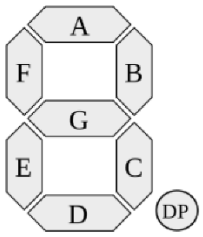
1. Analog Input. PA0(ADC0) is connected to the potentiometer through PORTA_SW1. To enable the potentiometer, 'PORTA_SW1' needs to be set properly. The first two switches need to be turned ON (labeled 0&1), this leaves PORTA floating. When these are ON, the analog input will be pulled to ground. Next, to use the potentiometer, turn on the third switch (labeled 2) as well.

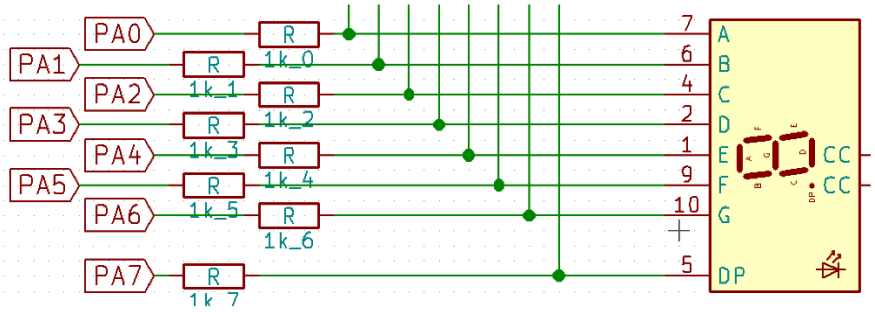


2. LED Output. LED0-LED7 are connected to PA0-PA7 through 1k Ohm current limiting resistors. To complete the circuit by giving the LEDs a path to ground, PORTA_SW1's first switch '0' must be enabled. If LED0 does not turn ON/OFF properly, turn off PORTA_SW1's switch '2'.



3. Seven Segment Display. The LEDs inside this KCS02 package use the same current limiting 1k Ohm resistors as the 8 individual LEDs. This is a common cathode (CC) device. To enable the SSD, set PORTA_SW1's switch 1 to ON. Disable PORTA_SW1's switch 2 if segment A is not working properly.

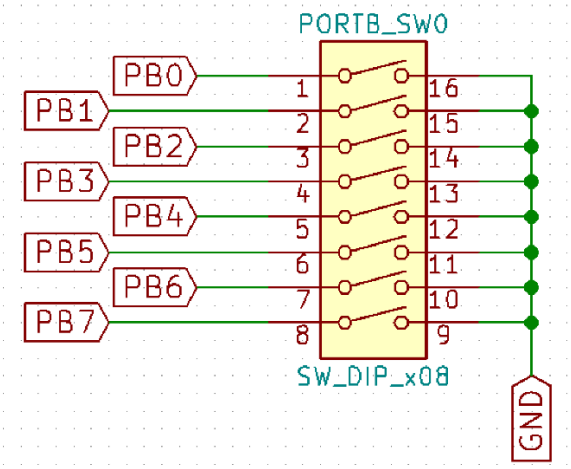




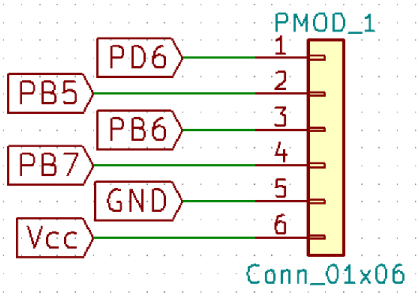
PORT B

PORT B has two purposes:

1. DIP Switch. This switch connects PB0-PB7 to ground when enabled. Remember to use the internal pull up resistors (through code) when using this input.

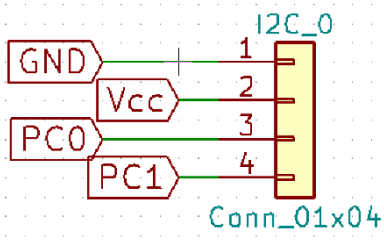


2. PMOD. The ATmega324PB uses PB5-PB7 for the SPI communications. When using this header, disable the corresponding switches on PORT B to leave them floating.



PORT C

1. I2C Interface. PC0 and PC1 are used for hardware I2C communications in the ATmega 324PB. VCC is 3.3V. Note that, the extension module given to you has wrong I2C pin indicators. SDA is SCL and SCL is actually SDA. It will be fixed in future edition.

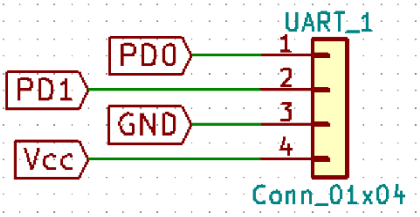


2. PC2-PC5 are used for JTAG communications and cannot be accessed by default.

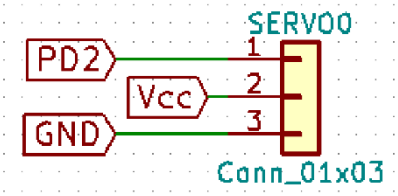
PORT D

Port D has four purposes:

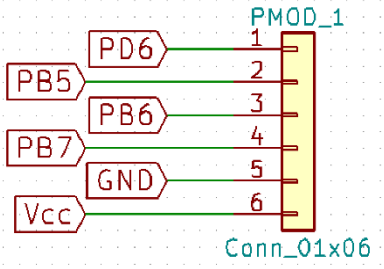
1. UART Interface. PD0 and PD1 are used for hardware UART communications in the ATmega 324PB. VCC is 3.3V.



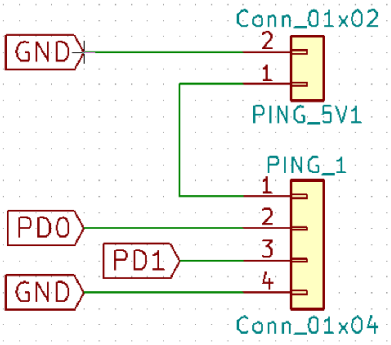
2. Servo Motor. PD2 is used to control the servo through PWM. VCC is 3.3V.

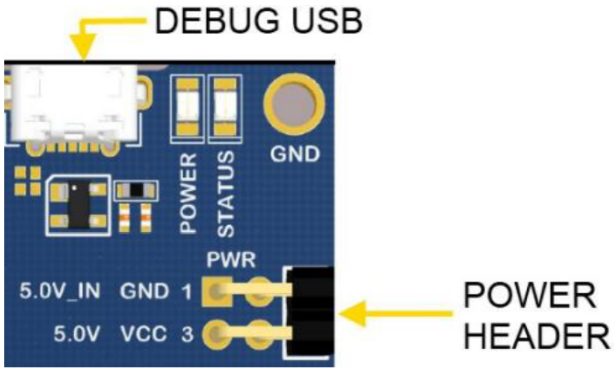


3. PMOD. PD6 is used as the slave select line for the hardware PMOD communications in the ATmega 324PB.



4. PING Sensor. The ping sensor requires 5V power. A female to female jumper wire can be used to get 5V from the USB input to the XPLAINED board.

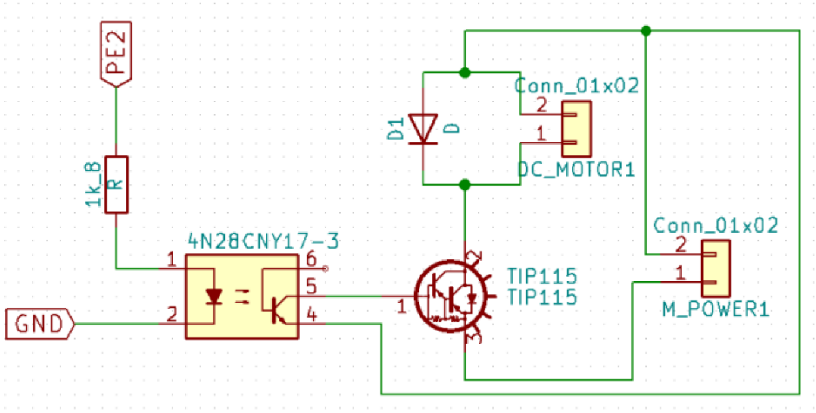




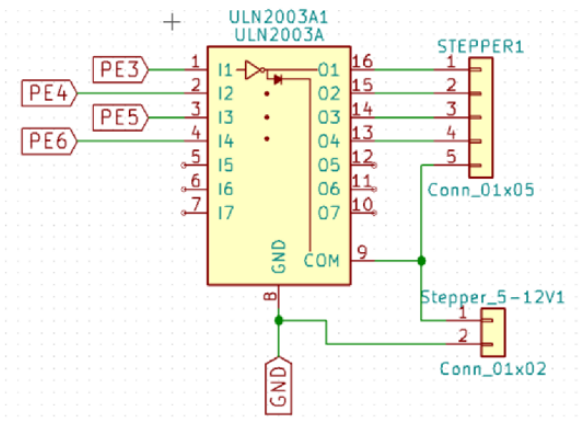
PORT E

Port E has two purposes:

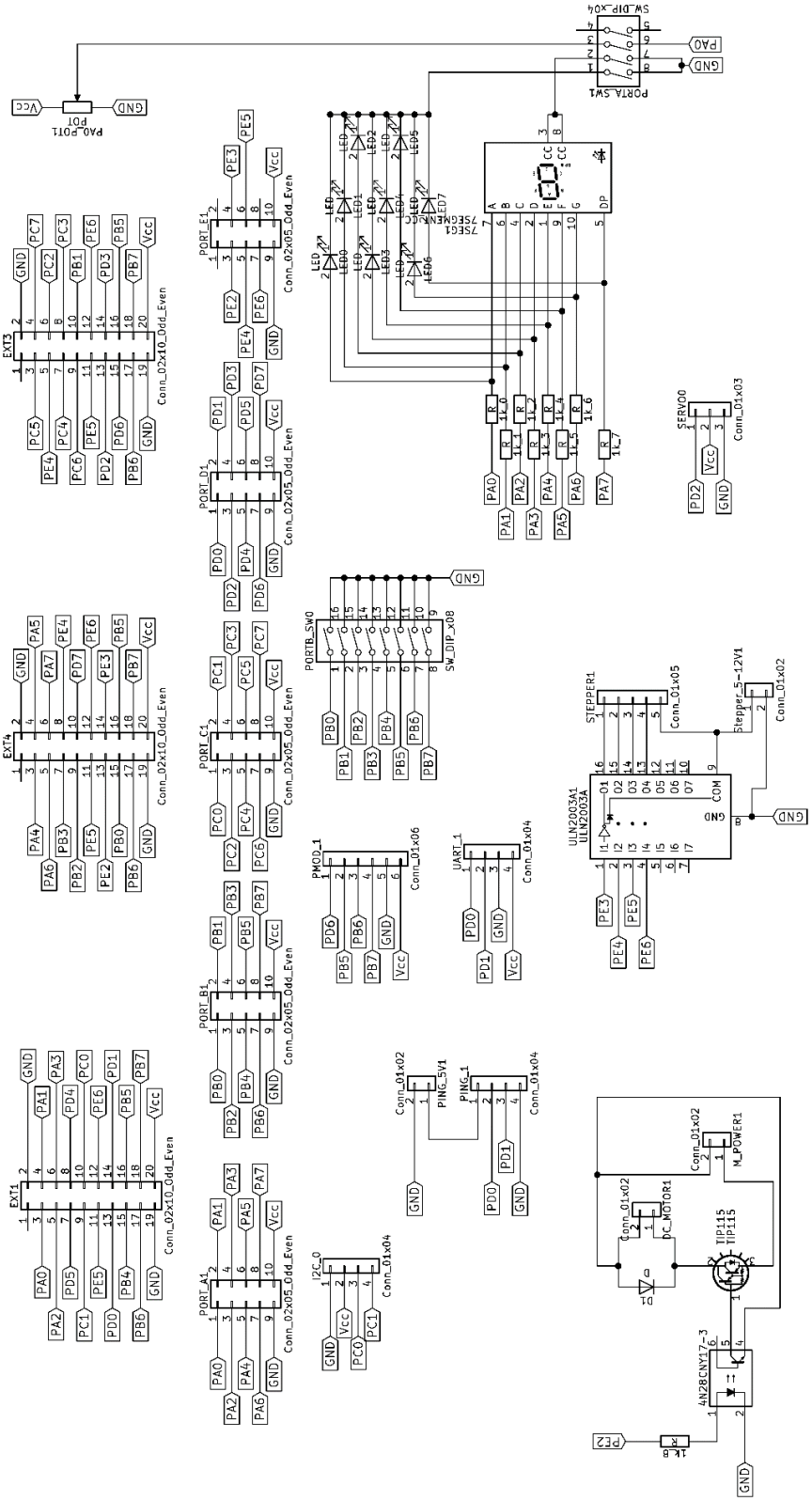
1. DC Motor. PE2 is connected to a 4N28 opto-isolator. This IC prevents noise and feedback from being introduced. A PNP Darlington transistor is used to amplify the signal. An external power source is required.



2. Stepper Motor. PE3-PE6 are connected to the ULN2003A IC. This IC amplifies the signals. An External power source is needed. 5V is the minimum and 12V is the maximum.



Full Schematic:



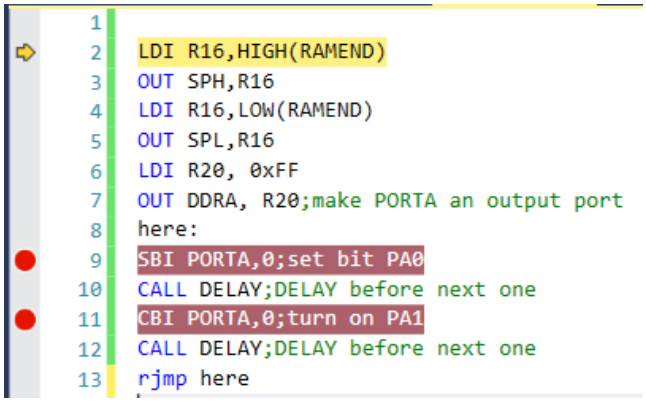
Activity 2: Program and debug the AVR hardware – Atmega324PB XPro board using Atmel Studio 7

2.1 Simulation mode

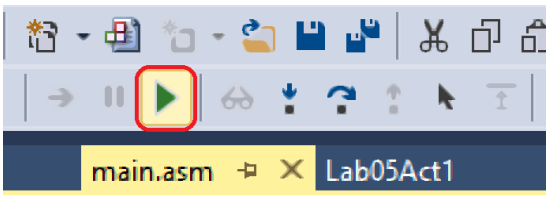
Step 1. Create a new project and run the following code in your code editor

```
//LED 0 Blinking
LDI R16,HIGH(RAMEND)
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16

LDI R20, 0xFF
OUT DDRA, R20    ;make PORTA an output port

HERE:
SBI PORTA,0      ;set bit PA0
CALL DELAY       ;DELAY before next one

CBI PORTA,0      ;turn on PA1
CALL DELAY       ;DELAY before next one
RJMP HERE

DELAY:    LDI r20,32
          L1: LDI R21, 100
          L2: LDI R22, 150
          L3: NOP
              NOP
              DEC R22
              BRNE L3
              DEC R21
              BRNE L2
              DEC R20
              BRNE L1

           RET
```

Step 2. Run the debugging mode with the simulator, put two breakpoints as shown below:



Step 3. click the run button to reach the first breakpoint



Step 4. click the run button to reach the first breakpoint, then check the stopwatch (the frequency must be set to 16MHz)

```
1
2   LDI R16,HIGH(RAMEND)
3   OUT SPH,R16
4   LDI R16,LOW(RAMEND)
5   OUT SPL,R16
6   LDI R20, 0xFF
7   OUT DDRA, R20;make PORTA an output port
8   here:
9   SBI PORTA,0;set bit PA0
10  CALL DELAY;DELAY before next one
11  CBI PORTA,0;turn on PA1
12  CALL DELAY;DELAY before next one
13  rjmp here
```

Processor Status

Name	Value
Program Counter	0x00000006
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	6
Frequency	16.000 MHz
Stop Watch	0.38 µs
Registers	

Step 5. Click the run button to reach the second breakpoint. You should wait a little bit to complete the simulation, then check the stopwatch

```
1
2   LDI R16,HIGH(RAMEND)
3   OUT SPH,R16
4   LDI R16,LOW(RAMEND)
5   OUT SPL,R16
6   LDI R20, 0xFF
7   OUT DDRA, R20;make PORTA an output port
8   here:
9   SBI PORTA,0;set bit PA0
10  CALL DELAY;DELAY before next one
11  CBI PORTA,0;turn on PA1
12  CALL DELAY;DELAY before next one
13  rjmp here
```

Processor Status

Name	Value
Program Counter	0x00000009
Stack Pointer	0x08FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	2409712
Frequency	16.000 MHz
Stop Watch	150,607.00 µs
Registers	

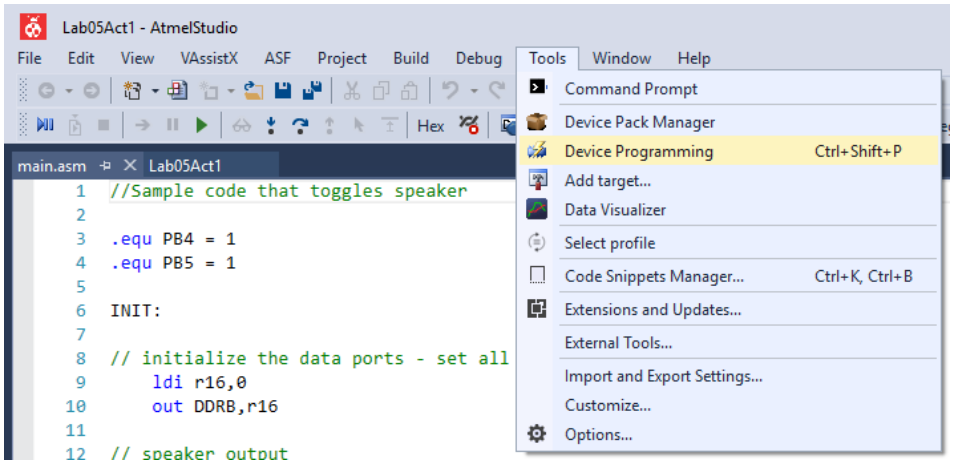
Step 6. Play with the numbers in the delay loop to make the time delay as 1 second.

```
14
15  DELAY:
16   ldi r20,32
17  L1: LDI R21, 100
18  L2: LDI R22, 150
19  L3: NOP
20  NOP
21  DEC R22
22  BRNE L3
23  DEC R21
24  BRNE L2
25  DEC R20
26  BRNE L1
27  RET
```

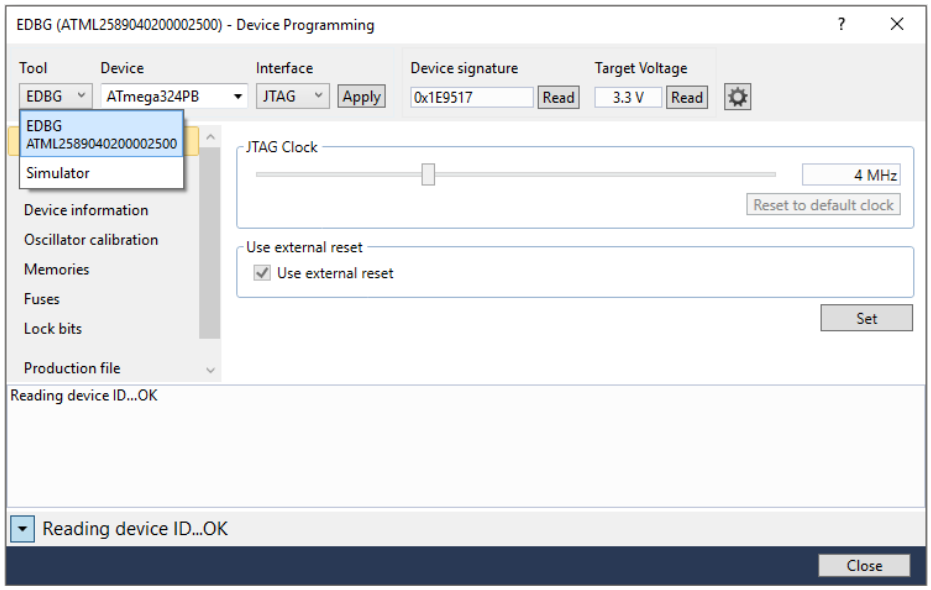
2.2 Hardware

Step 1. Run the code from the activity 2.1 (simulation) in your code editor. The code controls an LED on your board. Turn on PORTA_SW1 (push the switch '0' to close position) and turn off other switches.

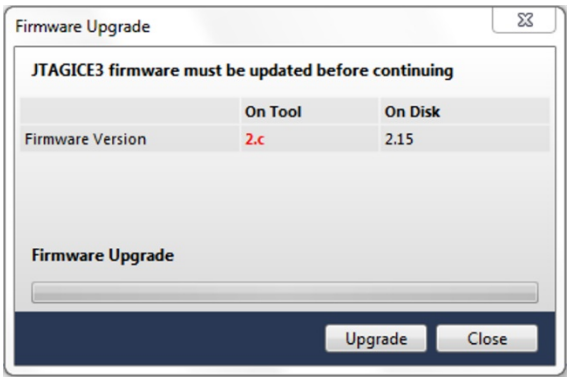
Step 2. Go to **Tools → Device Programming**



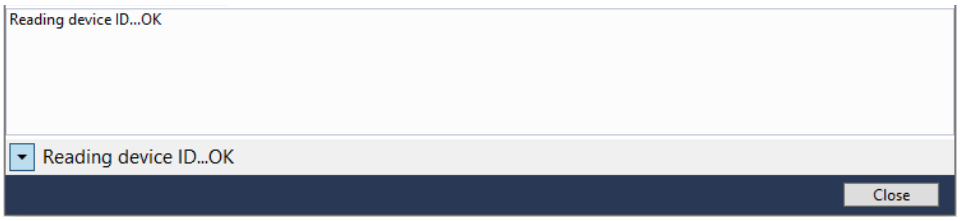
Step 3. Select **EDBG** from upper left corner, click on **[Apply]** beside JTAG.



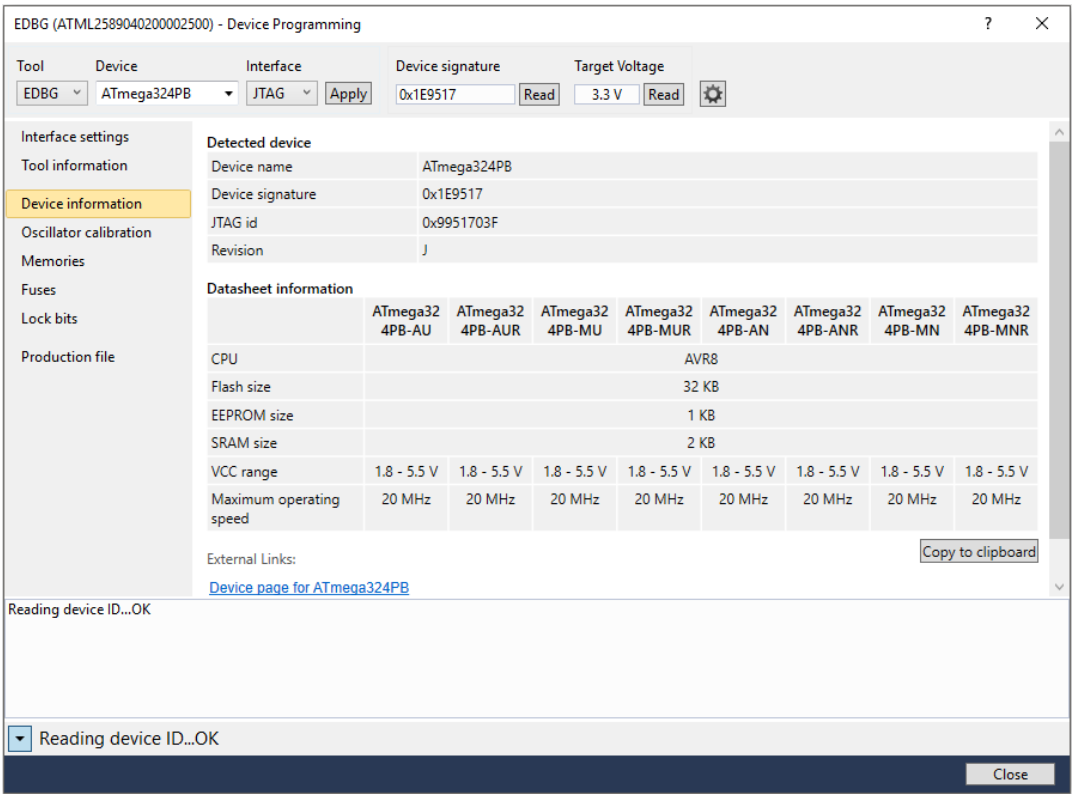
Step 4. In some cases, you will get the next window asking for an upgrade. Click on **Upgrade**.



Step 5. The log at the bottom of the window should provide you with an **OK** status.



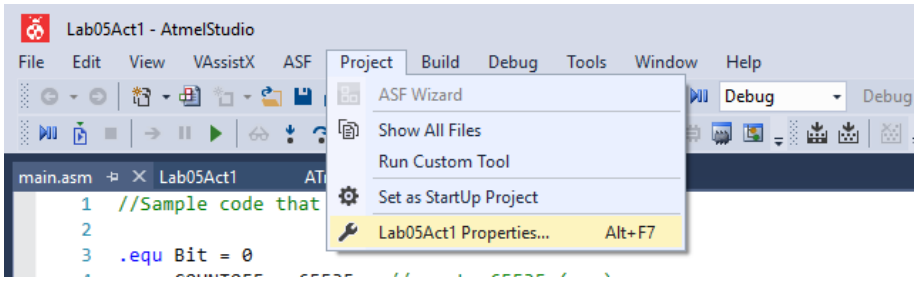
Step 6. Then click on **Device Information** and click on **Read** button. You will then get this status window and in the lower left corner it will have “Reading device ID…OK”.



Step 7. You have successfully connected the EDBG module for In-Circuit Programming and Debugging of the ATmega324PB Xplained Pro Board.

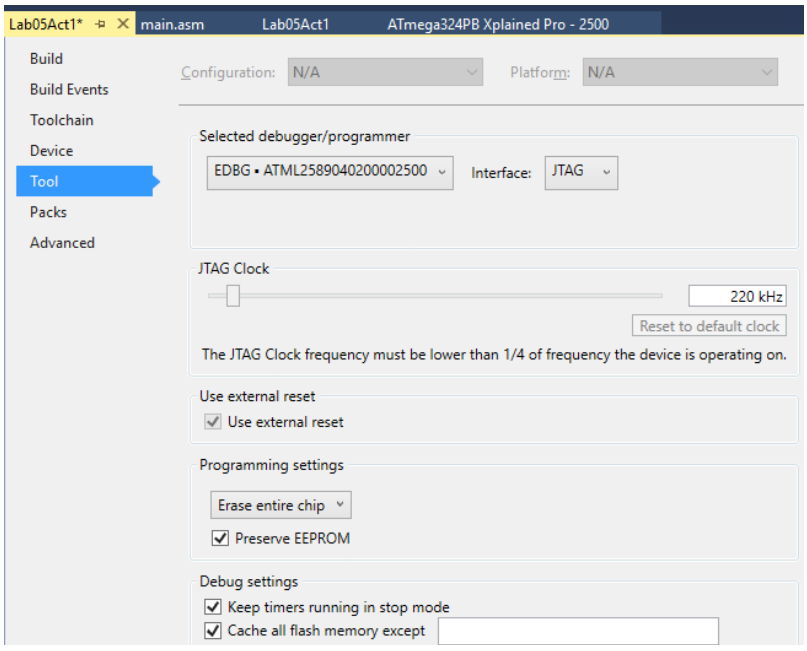
The techniques here are similar to those described in the Simulation Debug Views, except now the debugging occurs in the hardware. If you are not familiar with the simulation process steps, review them first.

Step 8. After you have debugged your program using the Studio 7 AVR Simulator, attach your hardware to your PC or laptop. Make sure you have exited from the Debug mode. Open the Project Properties window: **Project → 'ProjectName' Properties…**

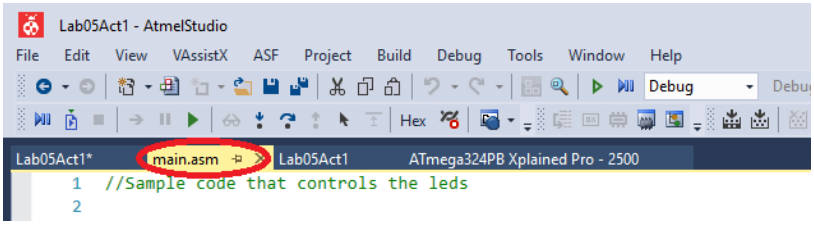


Step 9. Select the **Device Information** tab on the left. Then **Read**. It should provide microcontroller's information. If so, your connection is working.

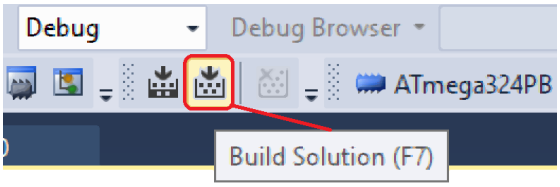
Step 10. Select **Tools** from the left side tab. The screen should show following information:



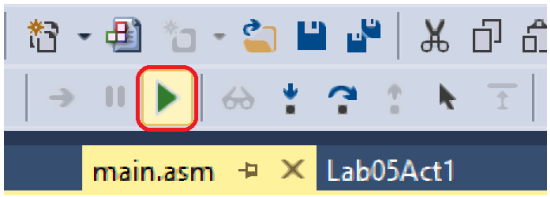
Step 11. Now click on the tab shown below to go to your code.



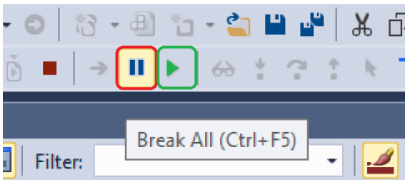
Step 12. Compile your code using the **Build Solution** button.



Step 13. Download the compiled program into your board. Your LED should blink really fast.



Step 14. **Start** and **Pause** the running code on the board by toggling between these two buttons.



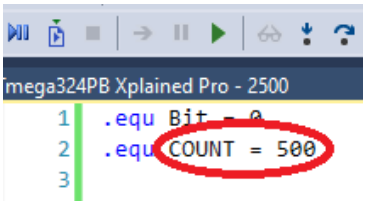
Assignment: Show your hardware results for the code of 0.15 second delay and 1 second delay of the blinking LED task to the instructor or TA to check out. (20 points)

Activity 3: Measure the output signal using the oscilloscope

We are going to run the following code and measure the time and frequency in the signal from the Bit 0 of the PortA. The following code will generate 50% of duty cycle pulse train based on the count. We are going to observe the signal and modify them to change the output's signals' time and frequency.

```
//PortA bit0 50% duty cycle output pulse train
.equ Bit=0
.equ COUNT=2000  //up to 65535 max

INIT:
//Initialize the output port A and set as output
ldi r16,0xff
out DDRA,r16

LAB6:                   //counter setup for on and off time
ldi r16,COUNT%256       //r16, r17 off time counter
ldi r17,COUNT/256
ldi r18,COUNT%256       //r18, r19 on time counter
ldi r19,COUNT/256

OFF_LOOP:               //off signal loop
subi r16,1              //decrement off time count [Subtract Immediate]
sbci r17,0              //upper byte [Subtract Immediate with Carry]
brne OFF_LOOP           //loop until zero [Branch if Not Equal]
sbi PORTA,0             //set to '0' [Set Bit in I/O Register]

ON_LOOP:                //on signal loop
subi r18,1              //decrement on time count [Subtract Immediate]
sbci r19,0              //upper byte [Subtract Immediate with Carry]
brne ON_LOOP            //loop until zero [Branch if Not Equal]
cbi PORTA,0             //set to '0' [Clear Bit in I/O Register]

rjmp LAB6               //loop forever
```

We will connect PC Oscilloscope to Port A Bit 0 and monitor the pulse width and frequency output while modifying the count value and reuploading the code.



Assignment: Fill-up the following table based on the oscilloscope reading. Then, show the corresponding oscilloscope views of the result to your instructor (or TA) to check out (30 points).

Count	Period	Frequency
1		
500		
1000		
20000		
60000		

To observe the output, an oscilloscope probe should be connected to the PORT_A1, PA0 pin (and scope's ground pin to board's ground pin).

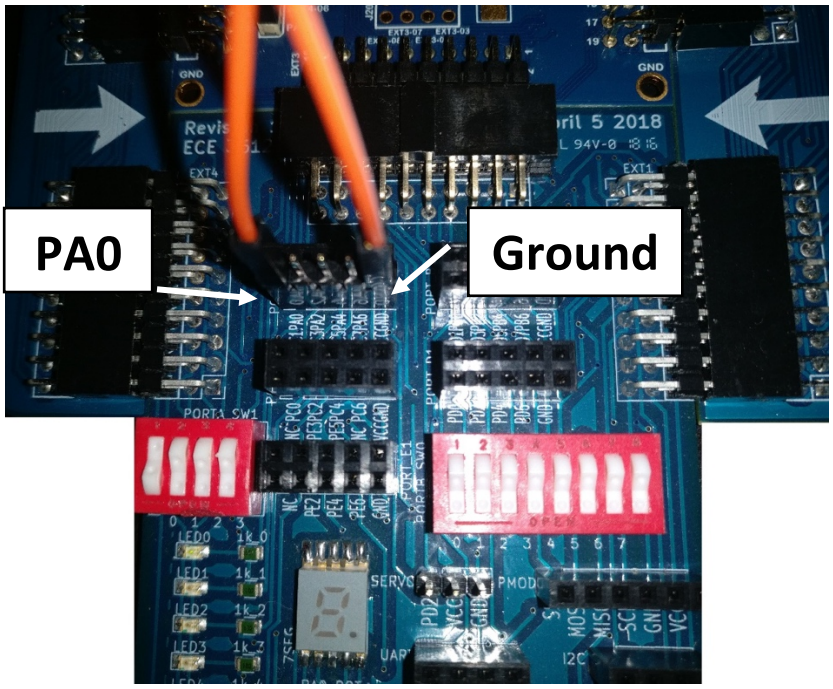


Figure: Oscilloscope connection to Port A bit 0, or PA0