TECH 3233
Lab #5
Analog to Digital Converters

Ver 2.0

In this experiment we will use the built in ADC Converter to read a Potentiometer display ADC out, and the Voltage on Input pin.

Note: Make sure your read the ENTIRE LAB HANDOUT before starting the project Important information on setting the project to use printf is included that must be done while starting the project)

The registers associated with the ADC in the Atmel 328P are as follows:

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS[1:0]: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 24-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 24-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal V_{ref} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V Voltage Reference with external capacitor at AREF pin

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "ADCL and ADCH – The ADC Data Register" on page 259.

- **Bit 4 – Reserved**

This bit is an unused bit in the ATmega48A/PA/88A/PA/168A/PA/328/P, and will always read as zero.

- **Bits 3:0 – MUX[3:0]: Analog Channel Selection Bits**

The value of these bits selects which analog inputs are connected to the ADC. See Table 24-4 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

For channels ADC0-ADC7 make MUX3..MUX0 equal to the channel number. Beyond the value 0b0111 (chn 7) they are as follows:

1000	ADC8[(1)]
1001	(reserved)
1010	(reserved)
1011	(reserved)
1100	(reserved)
1101	(reserved)
1110	1.1V (V_{BG})
1111	0V (GND)

Note: 1. For Temperature Sensor.

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADATE: ADC Auto Trigger Enable**

When this bit is written to one, Auto Triggering of the ADC is enabled. The ADC will start a conversion on a positive edge of the selected trigger signal. The trigger source is selected by setting the ADC Trigger Select bits, ADTS in ADCSRB.

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the Data Registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a Read-Modify-Write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS[2:0]: ADC Prescaler Select Bits**

These bits determine the division factor between the system clock frequency and the input clock to the ADC.

Table 24-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

The ADC 10bit answer will be found in ADCL and ADCH registers (since 10 bits cannot fit in one 8 bit register). The ADLAR register defines the alignment (MSB in ADCH Bit 7, when ADLAR =1 OR MSB in ADCH BIT 1 when ADLAR=0) as shown below:

	ADCH								ADCL							
ADLAR=0	-	-	-	-	-	-	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
ADLAR=1	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	-	-	-	-	-	-

Note that in Atmel Studio, the above ADCH and ADCL are combined in a **16bit register named ADC** (not in Atmel documentation, found via the iom328p.h file)

There are two other registers associated with the ADC. They are ADCSRB which controls when the ADC conversion is started (not needed for this lab – leave as free running), and DIDR0 which turns of the ability to use a bit as a digital input pin as well as ADC. See Atmel documentation for more info on these registers.

Lastly, although not mentioned in the textbook, you do need to turn off the Power Reduction Mode for the ADC (reg PRR bit PRADC = 0). If left in power reduction mode, the ADC registers will not update and the ADC will not operate.

PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0	
(0x64)	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSART0	PRADC	PRR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

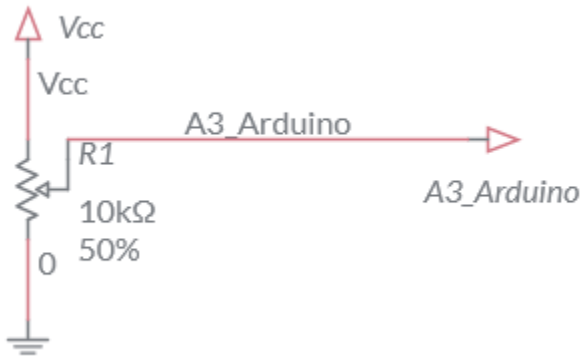
Atmel studio seems to write a 1 to all the bits in PRR, so you must write a 0 to the bits associated with the hardware you wish to use. REMEMBER THIS FOR FUTURE LABS!

Write to this register RIGHT AFTER the atmel_start_init(); line. Writing before it will have no effect, to other hardware control registers before turning on the hardware with the PRR register will have NO effect!

A few other notes:

1. Since we are using an Arduino board with a 16MHz clock on board, and the successive approximation ADC cannot be clocked faster than 200K, we can only use the /128 pre-scaler for the clock.
2. We will use AV_{cc} for our Reference Voltage (this is 5v on the Arduino board)

To wire the Potentiometer (Pot), we wire it as a voltage divider (with the wiper arm of the Pot to the ADC pin (AD3) and the other wires go to +5V and GND



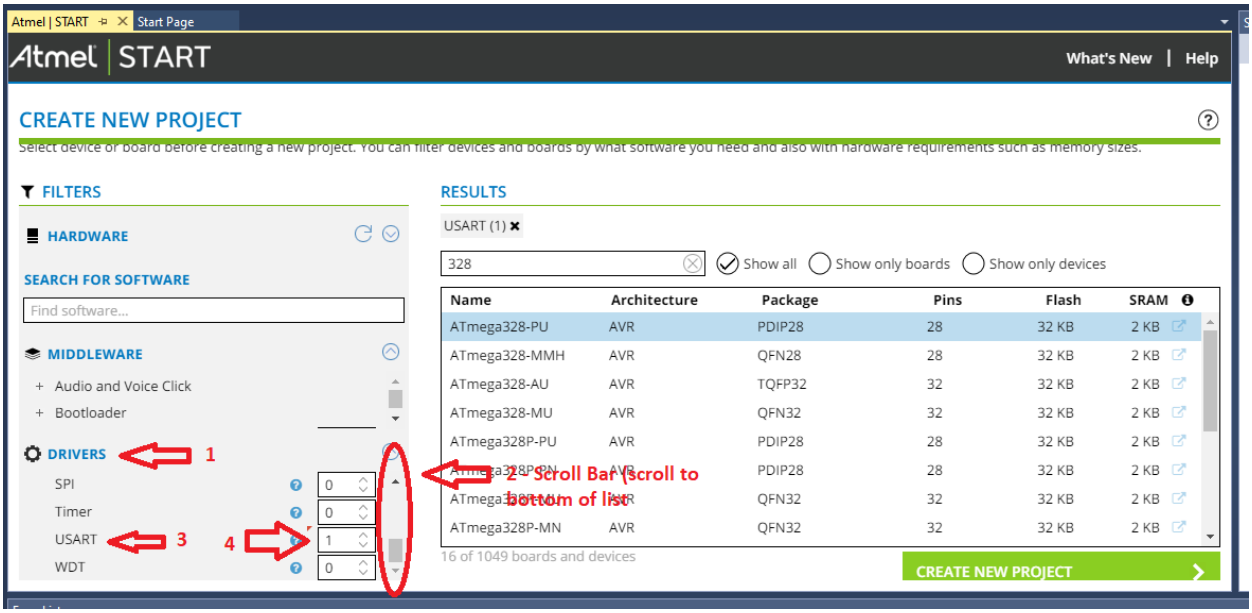
Write a program that will read an ADC input connected to the circuit (above) and display ADC out, and Voltage out every half second.

You should be able to figure out the formula for calculating the Voltage out (given the number of bits of the ADC and the V_{ref} = 5V).

The output should be in comma delimited format (#,#<cr><lf>).

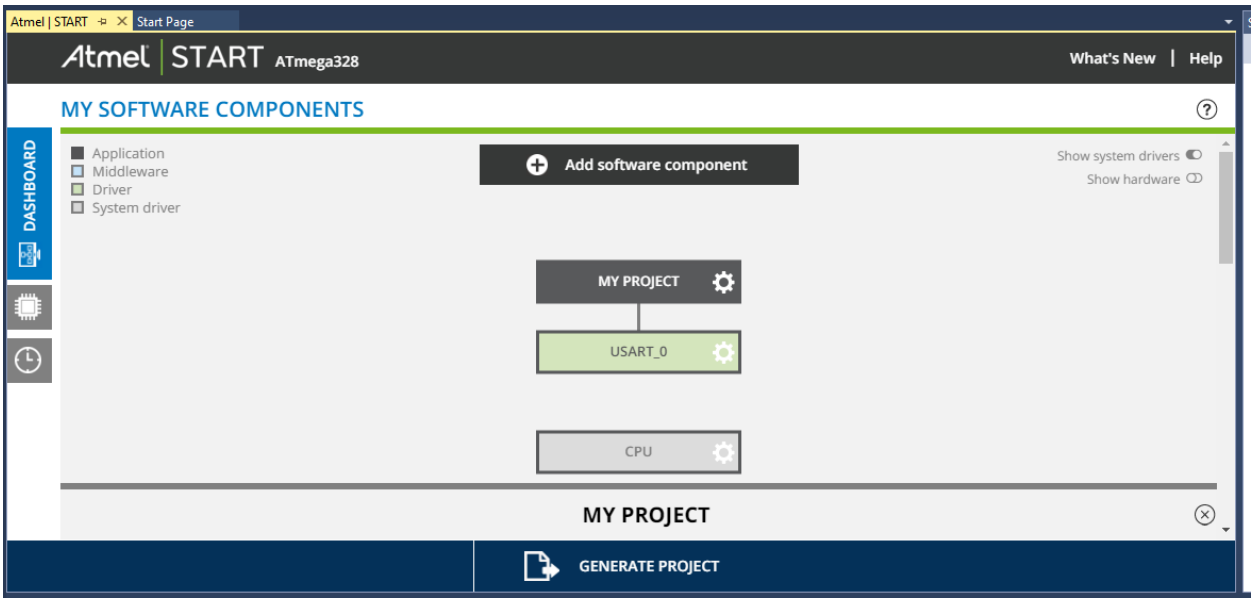
Since we want to print out the ADC value and Voltage, we will be using PRINTF for the first time.

To set up ATMEL studio to use printf, start the project as you did in LAB3 procedure Step 1, but in addition to selecting the ATmega328-PU, you will need to add the USART. To do this: Find Drivers (1 below) and use the scroll bar (2) to find USART(3), in the box next to it (4), put in a value of 1.

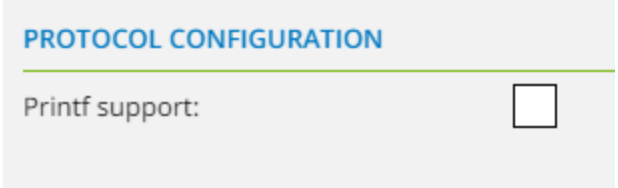


Then hit “Create Project”.

You should have:



Change the CPU clock speed to 16MHz as you have in the past, but now also click on USART_0 and scroll down to find:



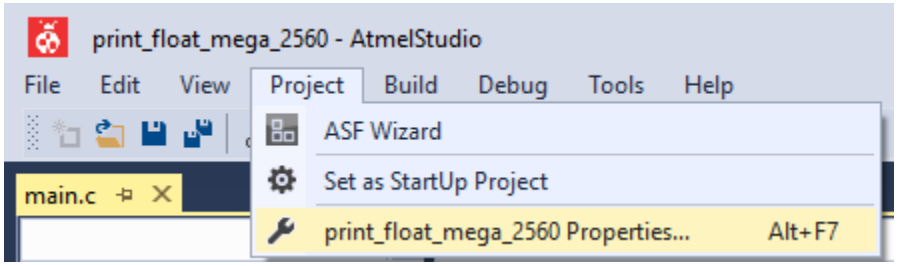
And put a check in the box (no other changes are required). Now hit “Generate Project”.

This will allow you to use printf in your program.

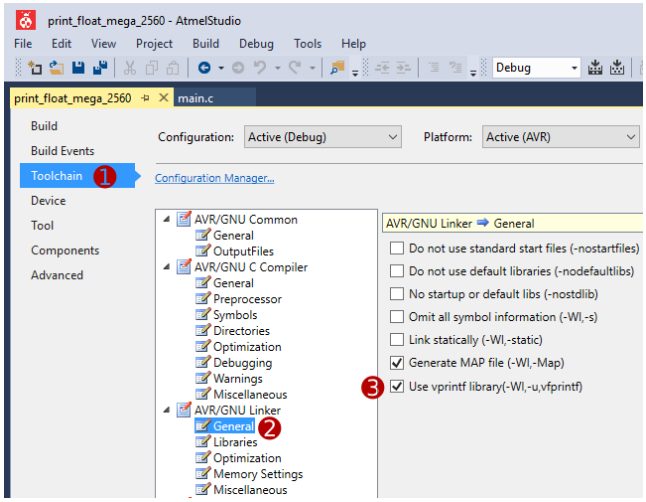
Since the voltage will be a **floating point number**, you MUST tell the compiler to enable the floating point printf (this is NOT included by default to save memory space).

To add floating point capabilities you need to do the following (from https://startingelectronics.org/articles/atmel-AVR-8-bit/print-float-atmel-studio-7/)

> In Atmel Studio 7 on the top menu, click **Project → <project name> Properties...** to bring up the properties page for the currently open project. The image below shows the menu in Atmel Studio 7 for a project named **print_float_mega_2560**.

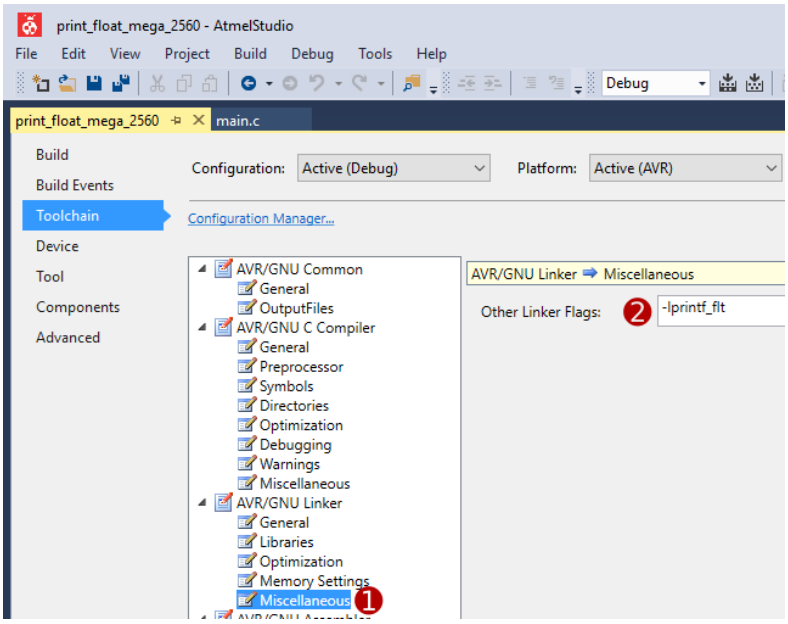


> Click **Toolchain** in the page at the left of the project properties page and then **General** under the **AVR/GNU Linker** item as shown in the image below. Finally check the **Use vprintf library(-Wl,-u,vfprintf)**.



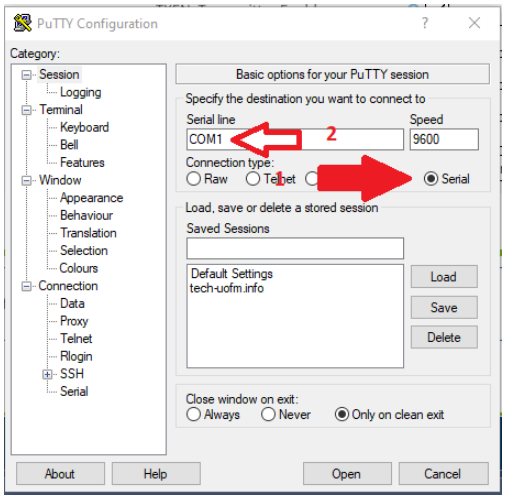
Now click **Miscellaneous** under the **AVR/GNU Linker** item and add the following in the **Other Linker Flags** box as shown in the following image.

```
-lprintf_flt
```



Save the changes to the linker options (Ctrl + S) and then rebuild the project. Projects that use sprintf and printf type functions should now be able to print floating point numbers to strings or standard output.

Since no “Screen” is attached to the Arduino, we use the same USB / COM port used to send programs to the board to send messages to a dumb terminal program on the PC. To open this, use the windows search bar to search for “putty” and click on the program found. You should get the following window:



Select Serial (1) and then change COM1 to whatever Com port the Arduino is on (same port as the setup for sending code).

NOTE: If you need to change your program and resend it to the Arduino, you will need to CLOSE PUTTY before you send, or you will get an error that says something about the port in use when you try to send the code to the board)

Once you have collected data for a while, copy and paste all data (right click on Putty's title bar and select "Copy all to clipboard", paste the data into Notepad (or notepad++) and save as a .csv file. Insert that into the root directory of your project file before zipping (this will act as your demo) and submit your fully commented code via electronic submission for full credit.