University of Massachusetts Amherst

College of Engineering

Department of Electrical and Computer Engineering

ECE231, Introduction to Embedded Systems, Spring 2020

**Lab 1**

**From C to C to C; programmed in C: being a Musical Signal Generator**

*Purpose*

To learn the fundamentals of programming the ATmega328P 8-bit AVR microcontroller in the C programming language using the Atmel Studio 7 Integrated Development Environment (IDE) and the Atmel ICE debugger.

To learn how to use the ADC (analog-to-digital converter) on-chip AVR peripheral.

To learn how to generate a square wave of a certain frequency using C software delay loops. The user will select the desired musical note by turning the analog linear potentiometer.

*Background Material*

The background material for this lab is in lectures 3, 4 and 5. The lecture videos are posted on the Echo360 site. Access the Echo360 recorded lectures via the link on the Moodle site.

*Logistics, Equipment, Parts List and Schematic Circuit Diagram*

The Lab 1 demos will take place in the evenings Feb 12 and 13 (Wednesday and Thursday). You will work in pairs. Do not divide up the work. Do the work together. Both team members must fully participate in every aspect of the lab. Do not store your team’s parts box in M5. Take it with you. You are responsible for returning all the parts and equipment at the end of the semester. Track all Piazza messages and additions to the resources section.

(continued)

Part Part

Reference Description

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

BOB1 SparkFun TRRS 3.5mm Jack Breakout

Digi-key Part Number: 1568-1382-ND

C1 22 microfarad electrolytic capacitor (OBSERVE POLARITY!!)

Read chapter 12. capacitor in the

Encyclopedia of Electronic Components Volume 1 (of 3)

by Charles Platt (UMass Library e-book)

C2 22 microfarad electrolytic capacitor (OBSERVE POLARITY!!)

WARNING Regarding C1 and C2:

These are electrolytic capacitors (polarized). Be very careful when placing the electrolytic capacitor into the breadboard – it is polarized meaning that you it matters which way you put it in. The stripe with the (-) sign is the negative lead, meaning the other side is positive.

C1 is acting as a noise filter. The positive lead of C1 must be connected to power and the negative lead to ground. If you don’t do this your capacitor could blow up. That’s not something you want to happen – it's dangerous and the odor is horrible.

C2 is acting as a DC voltage blocking capacitor. (In order to protect your earbuds.) The positive lead of C2 must be connected to the PB0 pin on the 328P and the negative lead must be connected to the proper end of R3 (240 ohms). If you don’t do this your capacitor could blow up. That’s not something you want to happen – it's dangerous and the odor is horrible.

R1 10 kilo-ohm resistor

Digi-key Part Number: 10KH-ND‎

Read chapter 10. resistor in the

Encyclopedia of Electronic Components Volume 1 (of 3)

by Charles Platt (UMass Library e-book)

R2 10 kilo-ohm potentiometer

Digi-key Part Number: 3310R-125-103L-ND

Read chapter 11. potentiometer in the

Encyclopedia of Electronic Components Volume 1 (of 3)

by Charles Platt (UMass Library e-book)

R3 240 ohm resistor

R4 47 ohm resistor

ANOTHER WARNING:

Don’t inadvertently swap R2 (240 ohms) and R3 (47 ohms)! If you do it will result in too much voltage being supplied to your earbuds or headphones, potentially destroying them. Make sure you follow the schematic carefully so you don’t ruin your earbuds or headphones.

SW1 SPDT slide switch (single pole, double throw)

Digi-key Part Number: EG1903-ND

U1 ATmega328P microcontroller

Digi-key Part Number: ATMEGA328P-PU-ND

X1 20 MHz clock (Abracon ACH-20.000MHZ-EK)

Digi-key Part Number: 535-9173-5-ND

Atmel ICE debugger with Atmel ICE flat cable

Digi-Key Part Number: ATATMEL-ICE-ND

Two (2) A-Male to Micro B Charger Cables

Adafruit USB micro-B breakout board (to power your circuit)

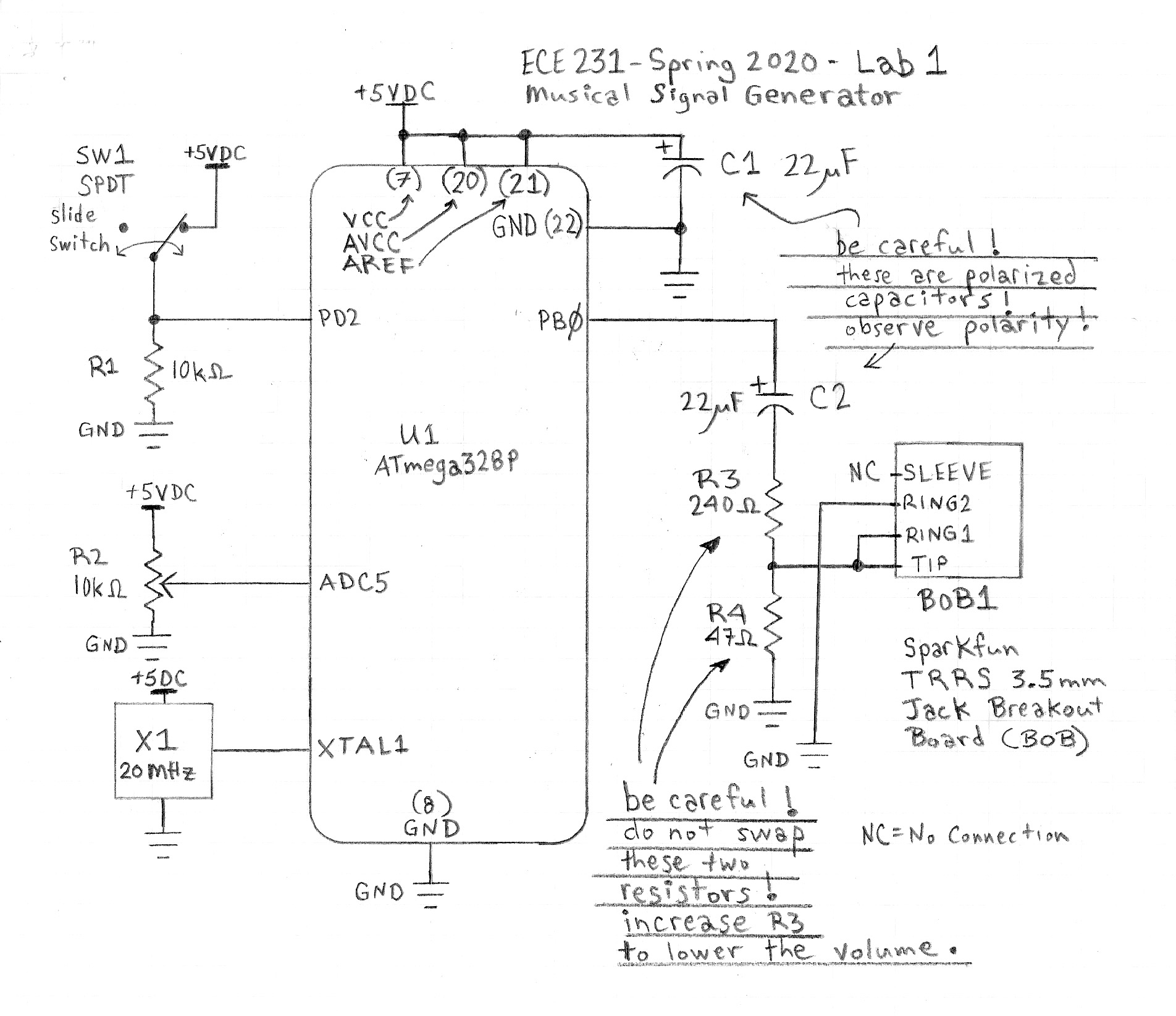
Digi-key Part Number: 1528-1383-ND

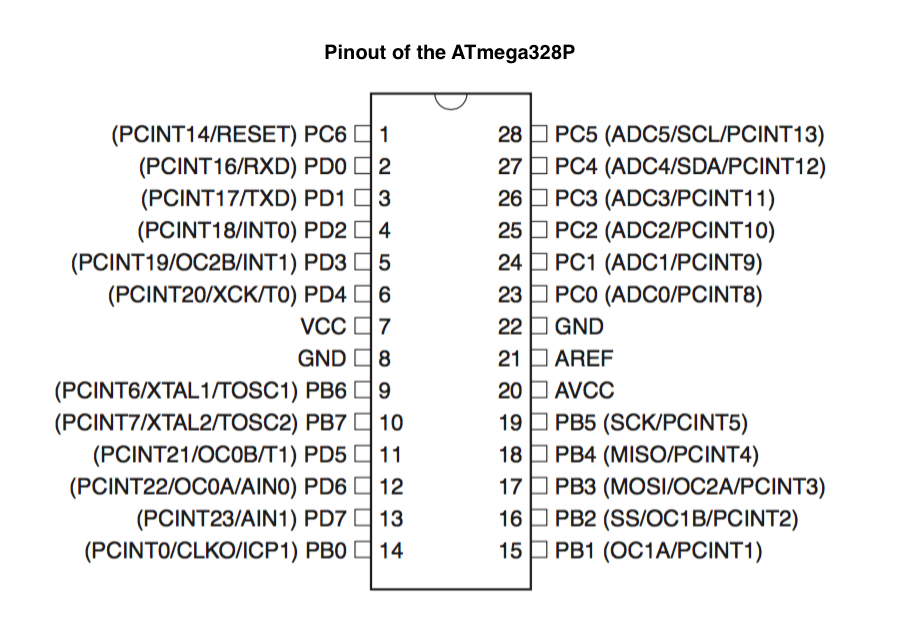
Solderless Breadboard Terminal Strip (No Frame)

Digi-Key Part Number: 1528-2143-ND

n.b. This is a parts list and not a Bill of Materials (BOM) because a BOM is "a list of the raw materials, sub-assemblies, intermediate assemblies, sub-components, parts, and the quantities of each needed to manufacture an end product." We are not manufacturing so we use the parts list term.

(continued)





*Lab Exercises*

Build the circuit described in the schematic circuit diagram above, with one exception: You are not required to build the switch circuit. You are not required to incorporate the switch in your solution. However, if you choose to add some cool optional functionality via the switch circuit, you must not modify the given switch circuit.

Add a POWER ON LED to your board. Not on the schematic. Get appropriate LED and current limiting resistor in M5. If needed, ask for help with the circuit and the calculation the resistor value.

Set the external clock fuse. Refer to the document

2020\_02\_Atmel\_ICE\_Debugger\_Instructions\_ver02.docx

on Piazza > Resources.

Your main task is to write a program which does the following:

You will write many small test programs. When you have a successful test program record that success in the comments of the program and save it. Do not simple keep adding to your code after you get one part working. Save successful test programs separately. We may ask to see the test program during your demo. You can check the ADC results and any variable within Atmel Studio 7 (AS7) when it is in debugger mode. Read about watch windows and breakpoints. Breakpoint actions can send variable values to the Output Window in AS7. (Refer to the AS7 User Guide in the Resources section.)

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

(continued)

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

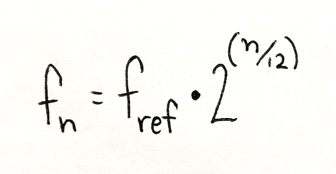
///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Your final program will, at any given time, generate a square wave with a frequency of one of 25 distinct musical notes. The note will be determined by the user who presents an analog voltage at ADC5 in the 0 to 5 volt range, inclusive. The 25 musical notes will be as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Octave Number | Ad hoc Lab 1 Note Number | Analog Voltage Range (Volts) | ADC result range (hexadecimal) | Musical Note | Frequency (approx.) (Hertz) | Period (Micro-seconds) | Half-Period (Micro-seconds) |
| 5 | 25 | 4.801 - 5.000 |  | C5 | 523.3 | 1,910 | 955 |
| 4 | 24 |  |  | B4 |  |  |  |
| 4 | 23 |  |  | A#4 or B flat 4 |  |  |  |
| 4 | 22 |  |  | A4 (aka A220) |  |  |  |
| 4 | 21 |  |  | G#4 or A flat 4 |  |  |  |
| 4 | 20 |  |  | G4 |  |  |  |
| 4 | 19 |  |  | F#4 or G flat 4 |  |  |  |
| 4 | 18 |  |  | F4 |  |  |  |
| 4 | 17 |  |  | E4 |  |  |  |
| 4 | 16 |  |  | D#4 or E flat 4 |  |  |  |
| 4 | 15 |  |  | D4 |  |  |  |
| 4 | 14 |  |  | C#4 or D flat 4 |  |  |  |
| 4 | 13 |  |  | C4 (Middle C) | 261.6 | 3,822 | 1911 |
| 3 | 12 |  |  | B3 |  |  |  |
| 3 | 11 |  |  | A#3 or B flat 3 |  |  |  |
| 3 | 10 |  |  | A3 (aka A220) | 220.0 | 4,546 | 2,273 |
| 3 | 9 |  |  | G# or A flat 3 |  |  |  |
| 3 | 8 |  |  | G3 |  |  |  |
| 3 | 7 |  |  | F#3 or G flat 3 |  |  |  |
| 3 | 6 |  |  | F3 |  |  |  |
| 3 | 5 |  |  | E3 |  |  |  |
| 3 | 4 | 0.601 - 0.800 | 0x007B - 0x00A3 | D#3 or E flat 3 |  |  |  |
| 3 | 3 | 0.401 - 0.600 | 0x0052 - 0x007A | D3 |  |  |  |
| 3 | 2 | 0.201 - 0.400 | 0x0029 - 0x0051 | C sharp (#)3 or D flat 3 |  |  |  |
| 3 | 1 | 0.000 - 0.200 | 0x0000 - 0x0028 | C3 |  |  |  |

The western musical scale consists of 12 semitones (notes) per octave. The frequency of any given notes is doubled from octave to the next. Middle C has a frequency of approximately 261.6 Hz. The next highest C (C5) has a frequency that is double that of Middle C.

Use the following formula to calculate the remainder of the frequencies and periods in the table above.



Where is equal to the number of semitones your desired note is from your reference note.

For example, in the table above we see that Middle C is three semitones beyond A220, therefore if we use A220 as our reference frequency (220 Hz) then the frequency of Middle C can be calculated to be approximately 261.6 Hz (220 x 2^(3/12)).

Please note that we are dividing up the 0 - 5 volt range into 25 regions, one for each semitone. Therefore, each voltage region is 200 mV wide.

You must be able to explain why each ADC result region above consists of 41 discrete values (or 0x0029).

Complete all of the entries in the table above.

(continued)

The image and link below should help you get your head around these concepts.



An 88-key piano, with the octaves numbered and Middle C (cyan) and A4 (yellow) highlighted.

(source: [Wikipedia article on the A440 (pitch standard)](https://en.wikipedia.org/wiki/A440_(pitch_standard)))

**It is recommended that you organize your final program as follows:**

**- initialize all appropriate peripherals**

**- enter an infinite loop:**

**- convert user analog voltage present on ADC5 pin**

**- determine into which of the 25 voltage regions this voltage falls**

**- determine the associated half-period for the selected note/region**

**- force PB0 HIGH for one half-period**

**- force PB0 LOW for one half-period**

**(repeat)**

(Note that in this scheme you are only generating one full cycle of a note for each time you convert the analog voltage and look up the half-period. There are far more elegant solutions available but they are all optional at this point. This primitive solution is the only one required in this lab. We will discuss the other solutions later. (They involve hardware timer/counters such as the Real-time Counter and Timer1.))

Test every portion of your solution separately and save these test programs:

Test program 1: the LED blinker: Add a blinker LED to your board. This is not on the schematic. Get appropriate LED and current limiting resistor in M5. If needed, ask for help with the circuit and the calculation the resistor value. Drive the blinker LED from PB0. Write an LED blink program: generate a square wave (approximately 5 Hz) on PB0 via software delay loops. (Force PB0 HIGH for a half-period and then force PB0 LOW for a half-period. Repeat forever.) Remove this LED circuit before writing the next test program.

Test program 2: Write a Middle C square wave generator program: generate a Middle C square wave (approximately 261.6 Hz) on PB0 via software delay loops. (Force PB0 HIGH for a half-period and then force PB0 LOW for a half-period. Repeat forever.) Verify timing on an oscilloscope. ((You must be able to explain why, mathematically, 261.6 Hz is the frequency of the Middle C note.)

Test program 3: Write an ADC test program: convert analog voltage once and store it in a variable. It is a 10-bit ADC. The results will be from the minimum (0x0000) to the maximum (0x03FF). (You must be able to explain why that is the maximum.)

Test program 4: Write a half-period lookup program. Determine the proper half-period for any given result from the ADC. Don't anything with this half-period value yet. Just store it. Verify that it matches what you expect.

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

This document was revised on 6 Feb 2020.

Copyright © 2020 Baird Soules