22.s902 Advanced Nuclear Laboratory

Tiny Multi-Channel Analyzer (MCA)

04.20.2024

OVERVIEW

In this lab we will study the signal from a spectroscopic detector (e.g. a NaI(Tl) scintillator based detector), and build a peak detector which will produce a slow signal proportional to the peak height of the detector pulse. Here the peak height will be proportional to the energy deposited in the detector. Then, using a microcontroller unit (MCU), we will digitize this value via the ADC on the MCU, and transfer the value to a computer via a serial connection.

The electronic "drivetrain" that goes from the sensor (detector) to the computer is referred to as **data acquisition (DAQ).**

GOALS

1. Observe the operation of a spectroscopic detector
2. Build a peak detector
3. Instrument it with an MCU
4. Acquire the data by the computer
5. Plot the spectra, and understand the various features therein

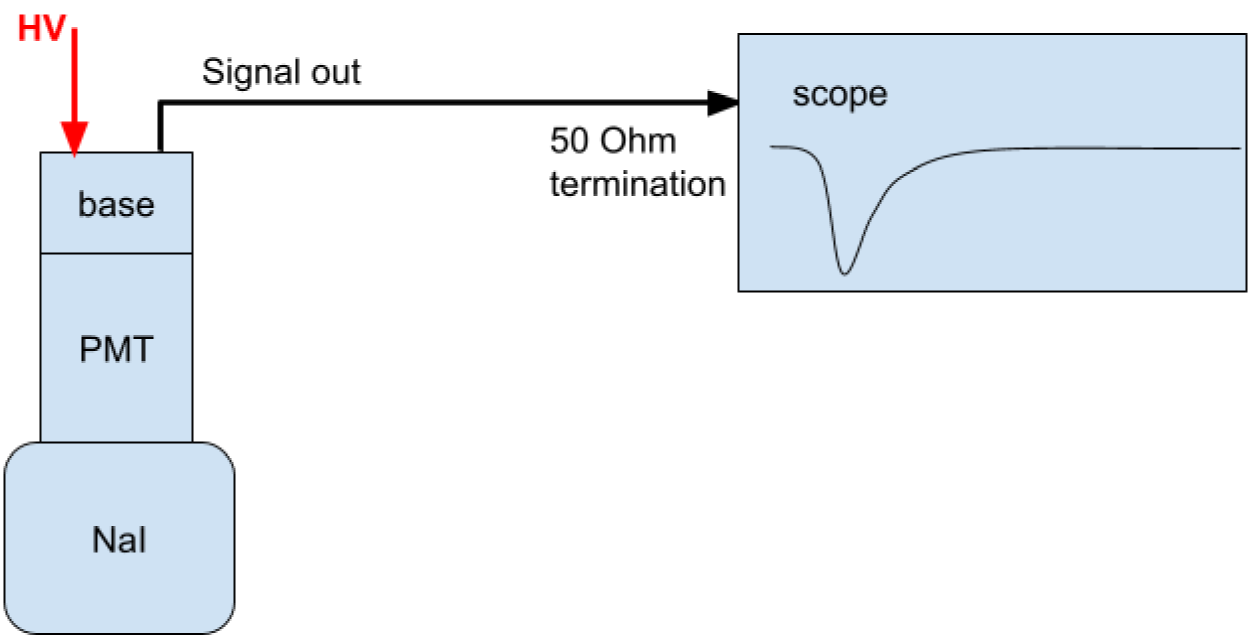
Signal from the NaI(Tl) detector

The thallium doped NaI inorganic crystal is one of the most common types of scintillators used in nuclear spectroscopy.

First, we will observe the signal from the NaI detector on a simple oscilloscope. You can think of the scope/oscilloscope as a very fast digitizer, which digitizes the analog signal (voltage) from the detector, and plots the results on an LCD screen.

Detector setup

With the help of the instructor, connect the NaI to a high voltage (HV) supply of appropriate polarity. Set the HV value and turn it on. The instructor will tell you the value of the voltage (~ 900-1500 V).



Look at the signal on the scope screen, and estimate the following quantities:

- **The rise time of the pulse. Rise time is defined as the time it takes the pulse from its 10% value to its 90% value. Answer:**
- **The time constant of the exponential decay of the pulse.   Answer:**

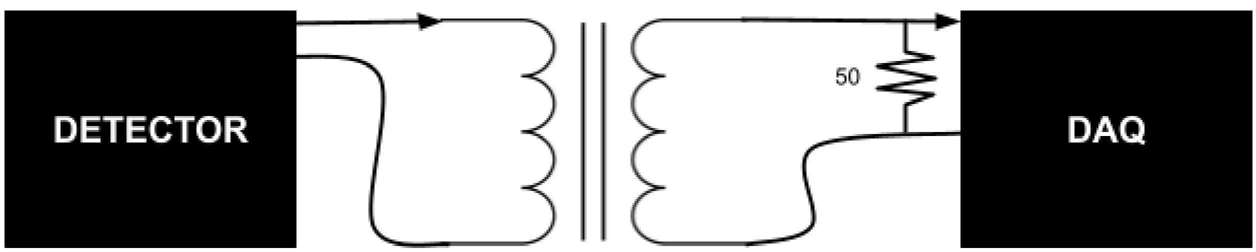
Inverting the polarity of the pulse

Now, for the purposes of the peak detector, we need a positive pulse. How do we invert the polarity of the pulse? This is a little tricky...you can change the signal/neutral wires of the cable, however that will introduce a number of problems due to the fact that the neutral connection is shorted to the ground of the HV supply.

To overcome this, we build a small transformer, which allows for isolation between the detector and the next stage of the DAQ. The instructors will either provide you such a transformer, or will show you how to build one. The transformer consists of three parts:

- A ferrite ring
- A wire, which wraps ~5-10 times around the ferrite ring
- A second wire which does the same

The first coil of wire is connected directly to the signal output from the NaI. The second one is sent to the next stage of the DAQ.



On the DAQ side we pick the wire endings such that the previously negative pulse shows up as a pulse with a positive polarity.

Once this has been set up, look at the signal on the scope, and confirm that you are seeing a positive pulse [if you still see a negative pulse switch the wires on the DAQ side)]. Qualitatively speaking, how are the detector pulses different from what you had seen in the previous step?

- **What is the time constant of the pulse decay? Answer:**
- **How does the undershoot compare to the main pulse height? Answer:**

Peak Detector

Now that we have a positive pulse, it's time to set up our (analog) peak detector circuit!

Here we use the simplest of peak detector designs, as described in Horowitz, Section 4.5.1, Figure 4.58 (**A**) – see next page.

Note the choice of the components:

- 0.47nF capacitor...this is the holding capacitor. It's tempting to make it large, as it can "hold more." But by the same token it will take longer to charge. We are dealing with *very* fast signals, so we can't afford any delays. So we gotta stay small on the capacitance.
- Small capacitor means that it will leak out its charge fast. This is why we need a very specialized op amp – OPA357. This op amp has a few critical characteristics:
    - Small input bias current. The input bias current tells you how fast your capacitor will 'bleed' out its charge into the op amp. So the input bias current needs to be << fA.
    - Fast slew rate. Slew rate tells you how many volts the output can change in a us. Since the signal is fast, the slew rate needs to be fast as well. We need at least 10V/us.
    - Large bandwidth. This is necessary for the NaI pulse, which has a rise of ~10ns. This requires >> 10 MHz.
- Schottky diode. Why not a simple PN? PN's are too slow, and they "eat" 0.6 V in a diode voltage. Schottky diodes are very fast, and have a diode voltage of only 0.3 V.

Art of Electronics Third Edition

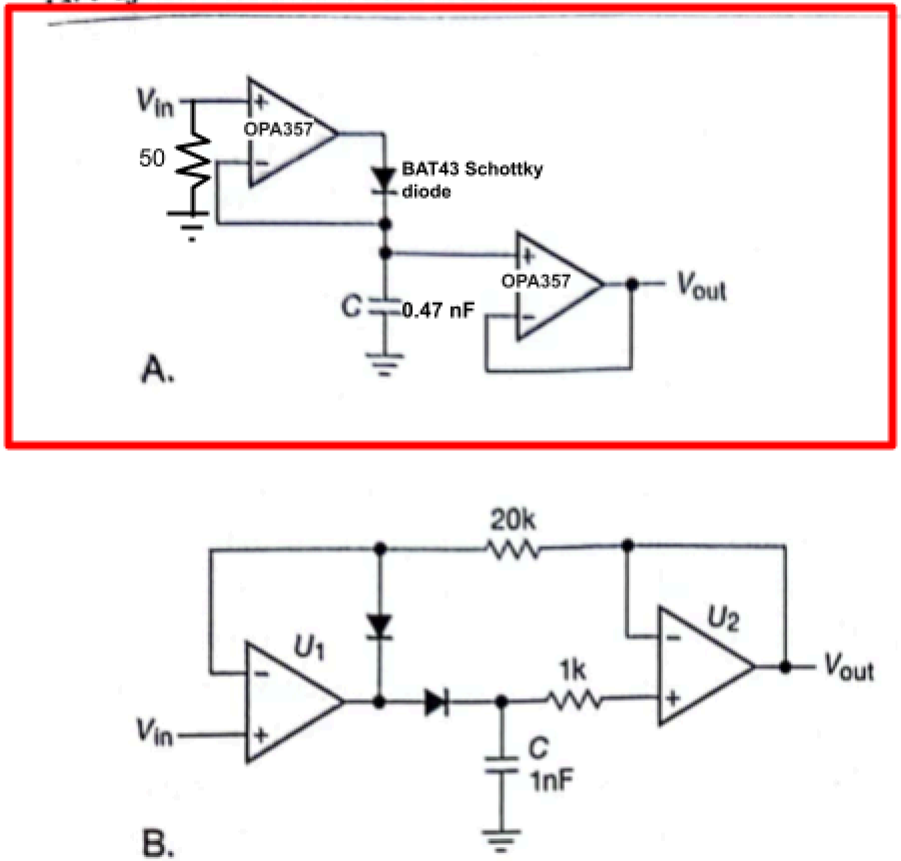


Figure 4.58. A. Op-amp peak detector (more accurately, a "peak tracker"). B. Improved peak tracker responds to short peaks, because the input op-amp does not have to slew from negative saturation.

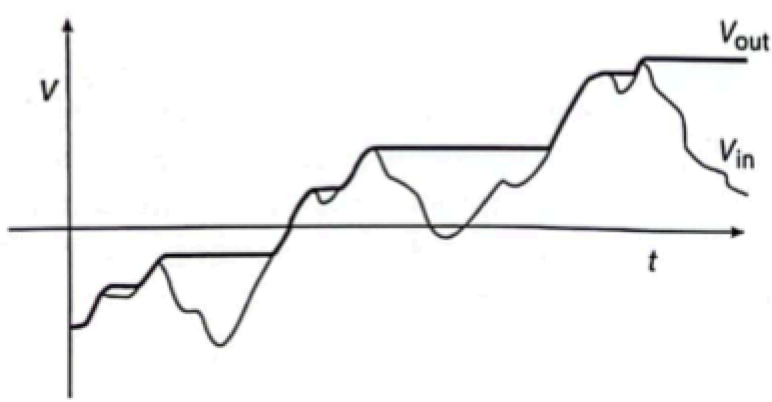


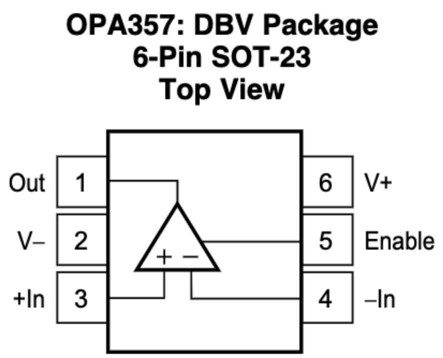
Figure 4.59. Peak detector output waveform.

Note that we have added a 50 Ohm resistor to the input to the peak detector. Why are we doing this? The reason is because the coaxial BNC cable that brings the signal to your peak detector has an electromagnetic impedance of 50 Ohms. By terminating this signal with 50 Ohms we ensure that there is an *impedance match*. This prevents reflections and ensures fidelity in the signal.

Now that you have had the chance to think about the setup, let's put together the actual circuit. Using a small breadboard (not the Elvis boards), assemble the circuit. A few cautions:

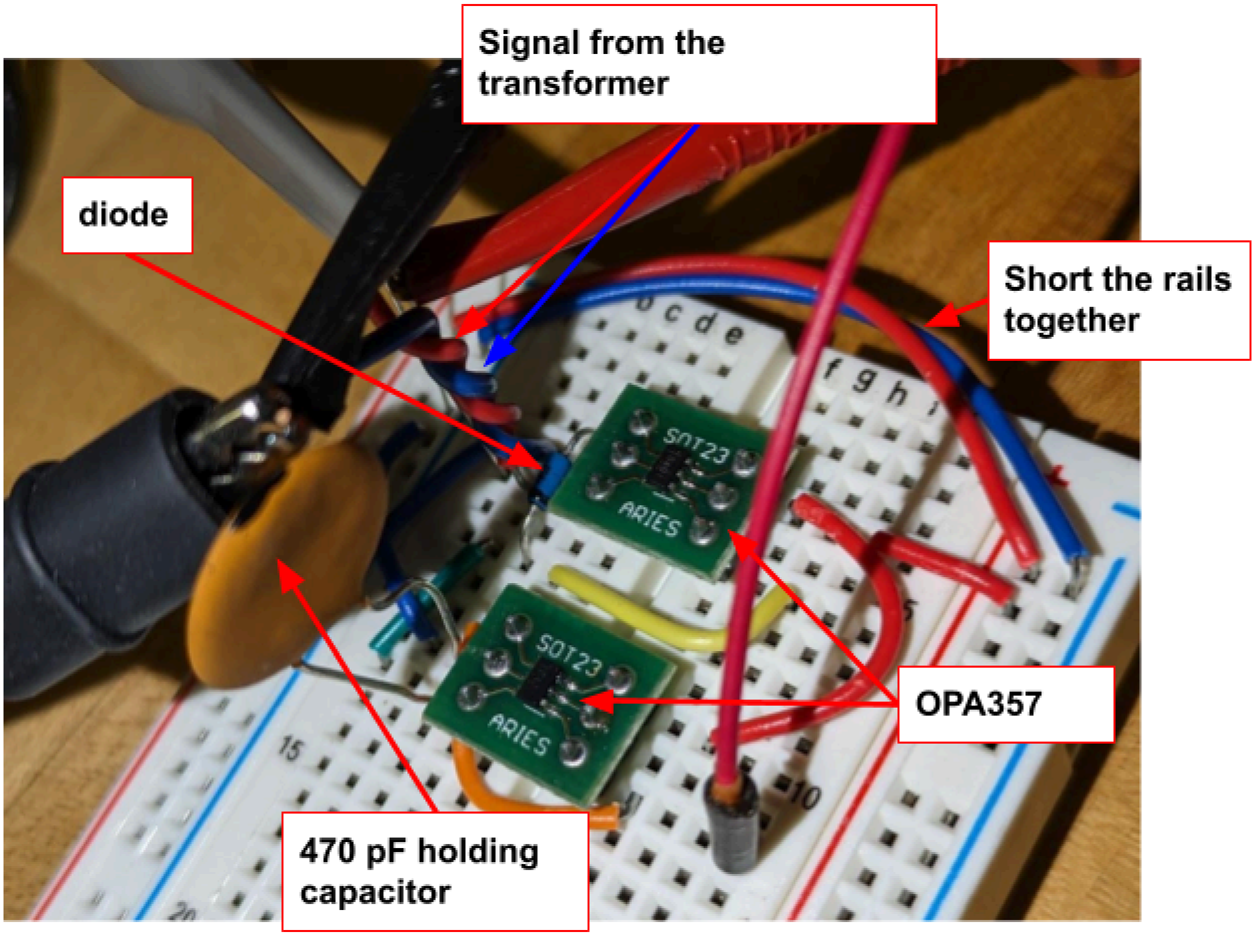
- Make your wires as flat and short as possible!
- Large jumper wires work like antennas – they radiate your fast signal (which is in the ~RF range), and pick up noise from the power tools of the workers next door.
- They also add parasitic capacitances and inductances.
- Use color coded wires as much as possible, this will make your circuit easier to read and debug. For example:
    - **Red** for power
    - **Blue** for ground
    - **Green**, yellow, and other colors for signal interconnects.

To understand what the op amp pinouts look like, here's a schematic from the spec sheet:



You can ignore the "Enable" pin. Just hook up V+ to 5V rail, and V- to the ground.

Here's what a setup could look like, as a reference:



The red and black grabbers are coming from the transformer. The alligator and the gray grabber are going to CH0 of the scope. The red jumper on the right is going to the CH1 on the scope.

Set the scope to trigger on CH0 – which should be the signal straight from the transformer that inverts the NaI pulse. Also enable CH1 – which should be the output of the peak detector.

Power – you need 5V for your circuit. Connect the serial adapter (provided by the instructors) to your laptop, and connect the adapter's Vcc and Ground wires to the + and - rails of the breadboard.

Plot (either by hand, or just embed a photo below) what you see, for a time scale of a few us. We want to see

a) The main pulse
b) The peak detector output

Now, let's see how long the capacitor can hold the charge. Still observing the output of the peak detector in CH1, change the scale of the scope to ~ms scale. **At what value of time does the output of the peak detector reach half its original value. Answer:**

For this you might need to put the scope in a "single trigger" mode. Most oscilloscopes have such functionality.

Comparator and Capacitor Reset

Before we can send the signal to the digitizing stage, there are two additional things we need to do. First, we need to compare our peak detector output to a voltage threshold and produce a LOGIC HIGH when that threshold is overcome. How do we do that? Well, comparators are our friends! Then, once the MCU (more about that later) has digitized the output from the peak detector it needs to reset the capacitor, so that we can detect the next pulse. How? By using a LOGIC HIGH sent to a MOSFET which works as a shorting switch on the capacitor. Let's go one by one.

MOSFET Reset

Right below your capacitor add a fast MOSFET, such that its drain (top) is connected to the capacitor, while its source (bottom) is shorted to ground. The middle pin, i.e. the gate, will receive a LOGIC HIGH from the MCU (more about it later).

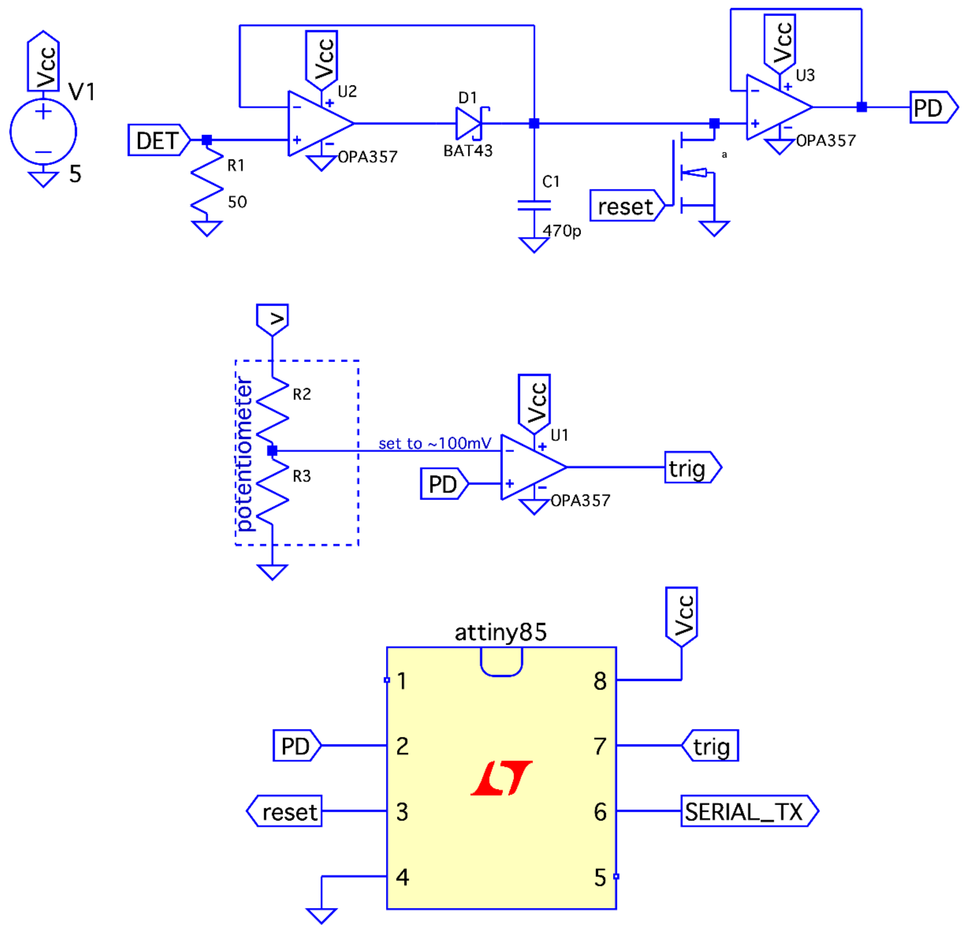
Comparator

Add an additional OPA357 below the assembled components on the breadboard. Send the output of the peak detector to the positive input. Next to the op amp put a potentiometer, with its external pins connected to 5V and ground, and its middle pin (sweeper) hooked up to the negative input of the op amp. The output of the op amp now works as a comparator – it will need to be sent to the ADC pin of the MCU (more about it later).

Now, connect a voltmeter to the negative input of the op amp, and change your potentiometer until you see a voltage of 0.1 V.

The circuit so far

The circuit schematic should look like the following (plus the MCU, which is in the next step).

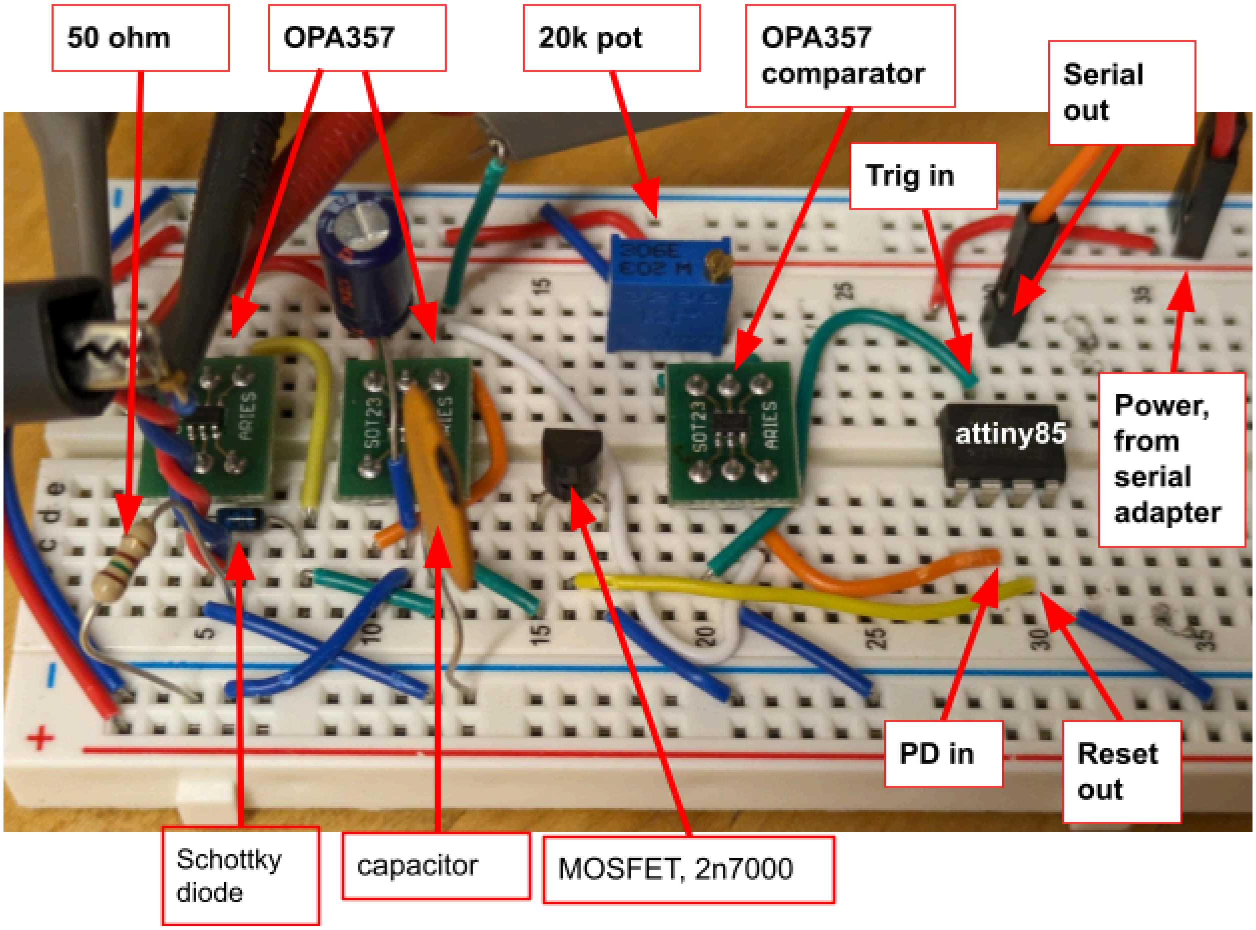


← **Peak Detector (PD)**

← **Comparator**

← **MCU** (the brain)

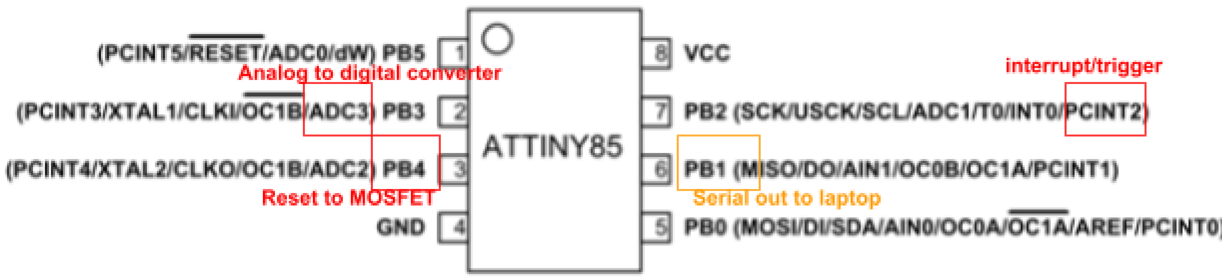
Here's a photo of what the full assembly could looks like:


50 ohm
OPA357
20k pot
OPA357 comparator
Serial out
Trig in
attiny85
Power, from serial adapter
PD in
Reset out
Schottky diode
capacitor
MOSFET, 2n7000

Digitizing the output

Finally we have the peak detector working. However we need numbers, and not just pretty traces on a scope's screen. This is where a digital microcontroller unit (MCU) comes in handy. Think of an MCU as a tiny computer that does few things really well.

For this course we will use a cheap, $1.6 ATTINY85 MCU. Below is the pinout of ATTINY85:



The MCU is one little busy guy (or gal) – it does a lot of things!! Let's go through them one by one:

- **Pin 7**, marked as PB2, is going to be programmed to accept a trigger input. Why? Well, someone has to tell the MCU *when* to sample the output of the peak detector, right? It can't be sampling all the time, right? More about this later...
- **Pin 2**, marked as PB3, is going to be programmed to do the actual voltage sampling via its *Analog to Digital Converter* (ADC). ATTINY85's ADC is 10 bit, which means that it produces numbers between 0 and 2^10=1024 that are proportional to the voltage, with 0 being 0V, and 2^10 being 5V. So, for example, if you get the number 835 then the corresponding voltage is 5V*835/2^10 = 4.08 V. **Question – what is the ADC resolution (i.e. the smallest step in voltage that the ADC can resolve) in voltage? Answer:**

- **Pin 3** is going to reset the holding capacitor. Why? Well, once you are done with the sampling, you want to reset the capacitor so the whole process can be repeated. More about this later...
- Finally, once the MCU does its sampling and resetting and all, it needs to send the data to your computer for you to acquire and analyze. But how? This is done via the Serial protocol. **Pin 6** will be programmed to send out the numbers to the Serial adapter, which will then transfer the information to your CPU. You can acquire this signal on your laptop either via a serial monitor software, or via a simple python script that we will provide you.

But what do we mean with “trigger?”

Trigger

How do we trigger the MCU? How do we tell it that now is the time to sample the ADC? Naively speaking one could send the peak detector’s output to the MCU, tell the MCU to sample over and over again, and tell it to trigger once the voltage is above some threshold. However our MCU is small and weak, it can’t do that – not well at least.

To trigger fast, the MCU needs a LARGE logic pulse, i.e >4V, sent to one of its *external interrupts*. An *interrupt* tells the device (within one clock cycle) to interrupt EVERYTHING that it’s doing and do “something special” NOW. Where you tell it what “something special” means. This way it knows that, yes, it’s time to trigger. Once it gets such an interrupt the MCU is lightning-fast – it can trigger in a ~microsecond. But how do we get there?

This is where comparators come in, as discussed previously. In addition to sending the peak detector voltage to the MCU, we will fork it and also send it to another OPA357 op amp configured in a comparator mode. The threshold can be set with a voltage divider, e.g. at 0.1 V. The choice of the voltage is important: too low and you will trigger on noise; too large and you’ll miss out on valuable pulses. 0.1 V has been determined empirically, based on our op amp’s characteristics.

Programming the MCU

To program the MCU you’ll need, well, a microcontroller programmer. The instructors will have one handy, and will program the attiny85 for you. You can observe the arduino code of the github link here. Go ahead and take a look.

Observing the operation of the MCU

We don’t know what’s going on inside the MCU, but we can observe how its resetting the capacitor. Hook up CH1 of the scope to the output of the reset pin, and observe the reset. If you consult the code above, you’ll see that the reset is happening immediately after the ADC of the MCU is done sampling the peak detector voltage. Questions:

- How long does it take for attiny85 to sample the voltage from the peak detector? Answer:
- How long is the reset? Answer:

Transferring data to your laptop

Finally the MCU has sampled the ADC. It also uses the serial connection to ship the ADC values to whoever is listening on the other side of the serial bus. For this, you need to hook up the TXD (transmit) pin of your USB-serial adapter (provided by the instructors). Then, once the MCU is running, on your laptop you need to run the python script.

The script is also available on github. Download it and read the instructions in its header.

Once the MCU is running and the serial adapter is plugged in, on your laptop side find out the serial device. On a Mac/Linux it will be in your /dev directory. For example, on my Mac it looks like something similar to `/dev/cu.usbmodem142401`. Once you identified the device, you can run the python script in the command line:

```
> ./logger_printl_generic.py /dev/cu.usbmodem142401
```

If everything is working fine, this will spit a bunch of numbers on the screen. You are interested in the first column.

To save it to file, and say save 10000 counts, you need to do

```
> ./logger_printl_generic.py /dev/cu.usbmodem142401 -o myfile.txt -p 10000
```

The code will “hang”, but you’ll notice that myfile.txt is being filled. Take a look at it. It should look very similar to what you saw earlier.

Taking radiation data with NaI

Finally we have a fully working DAQ – time to take some real data!!

Background

Take data for 10000 counts. Read the data into ROOT, and histogram the ADC. You should be able to see the peak 40K peak at 1460 keV. **Show the histogram below (and remember: axes labeled!!), marking up the 40K peak**.

What is the ADC value of the main line? Answer:

Can you see the muons? What are their energies (use 1460 keV line for a quick calibration)? Approximate answer:

137Cs data

Now place a 137Cs source next to NaI, and acquire 10000 counts. **Show a histogram of the spectrum**. Can you see the 662 keV peak?

What is the ADC value of the main line? Answer:

60Co data

Replace 137Cs with a 60Co source. Look up the two lines that make up 60Co's gamma emission. Can you see them? **Show**:

What is the ADC value of the main line (note: you might see the two lines "lumped" together due to the limited resolution of our DAQ)? Answer:

Linearity

Is our ADC linear? If so, the 1460 keV, 662 keV, and the 60Co lines' ADC should be proportional to the energies. **Do the calculation, using those to support your conclusions**.

Long Background (time remaining)

Run the DAQ for at least 100000 counts. This will take ~hr. Can you see the 2.6 MeV line fro the thorium/uranium decay chain?