

# NWS Capstone Supplementary

## Information

Ethan Wagner

Fall 2025

# Table of Contents

Disclaimer .....	3
Preface.....	4
The Post Charge Regulator .....	4
Overview .....	4
The Bleed Off Current .....	4
Fault .....	4
Triggers .....	5
Trigger Timings.....	6
Interpulse Period .....	6
Discharge Trigger Ambiguity .....	6
Software Clarification.....	7
Keypad .....	7
Menu .....	7

# Disclaimer

This document was written by a capstone student **NOT** a Radar Operations Center engineer.

Confirm all important information with capstone sponsors.

## **Preface**

I am writing this document because it would have been very helpful to me when I started this project. In it, I'll try to cover the basic information that you'll need to get started on your project and get a better understanding of the Post Charge Regulator and the related systems. Since I worked on the software, the information in it may be biased toward software.

## **The Post Charge Regulator**

### **Overview**

The Post Charge Regulator (PCR) is a module in the radar transmitter that regulates the voltage of another module called the Pulse Forming Network (PFN). This regulation is necessary because the charge accumulated in the PFN is inconsistent between charging cycles. The PCR regulates the PFN voltage by bleeding charge out of the PFN. This charge is called the bleed off current.

### **The Bleed Off Current**

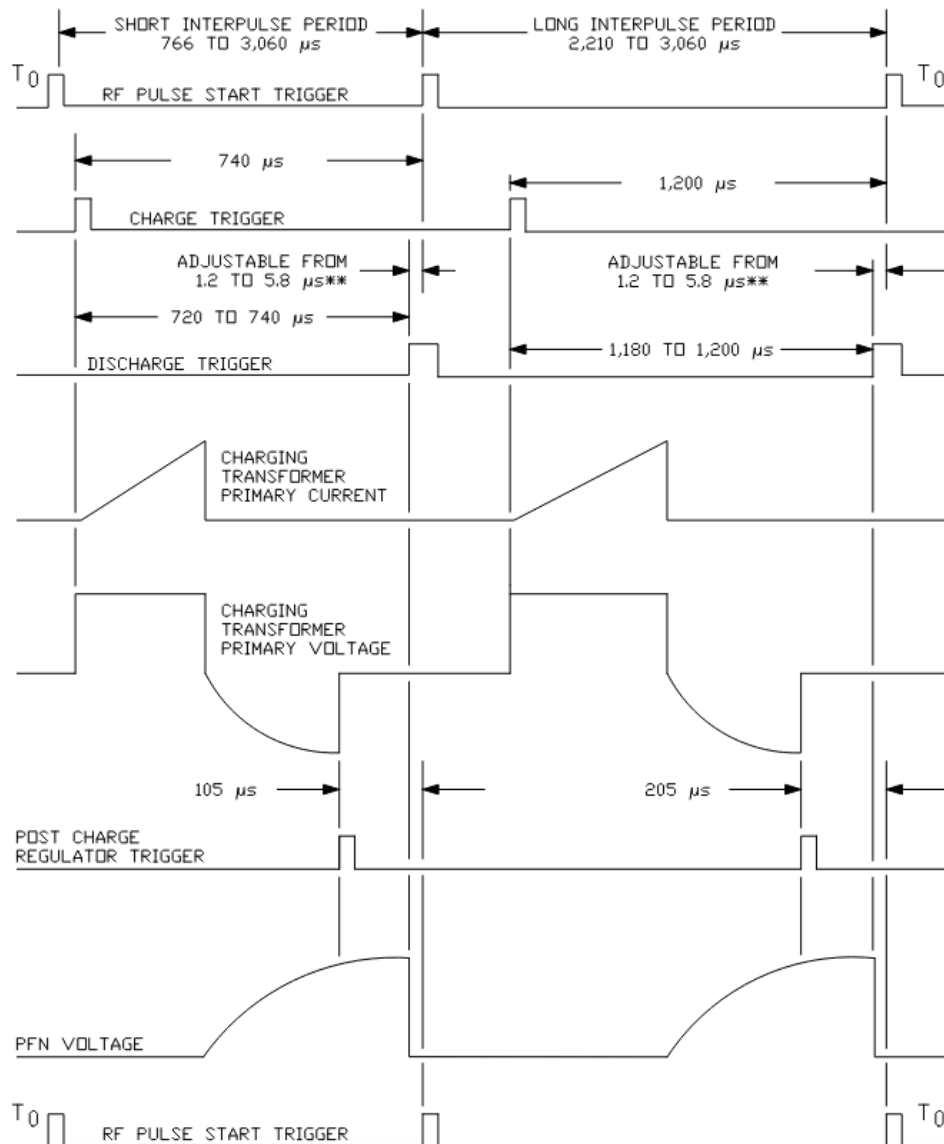
The bleed off current is very important in this project. To understand whether the PCR is doing its job, technicians need to have a measurement of this current. The bleed off current can be measured on the pin in the D-SUB connector labeled Current Monitor, which is pin 10. Here, the current can be measured as a voltage that is proportional to the current. The relationship I was given is 10mA/V, so for a voltage of 1.23, the bleed of current would be 12.3mA. You should verify the correctness of this value before using it.

### **Fault**

The PCR has an output called the fault. It is a logical value and it can be read from the pin labeled FLTOUT on the D-SUB connector, which is pin 5. This output must be enabled from the pin labeled FLTENIN, pin 1. This can just be tied to power so that the fault is always enabled. If a fault is detected, you must assert PCR RESET, pin 2, to clear the fault.

# Triggers

Triggers are logical signals that control the timing of important events withing the transmitter. In the test fixture, the triggers that are sent out are first split into differential pairs. These are the Charge, Discharge, and PCR triggers. There is another trigger, the RF Start trigger, which is an input to the system. This trigger dictates the timing of the rest of the triggers, which you can see in the diagram below.



This diagram is part of the modulator timing relationships diagram. On this diagram, there is a small note at the bottom that says, “Triggers are inverted for clarity.” How this clarifies anything is beyond me, but the consequence of this is that the triggers that are shown as active high are actually active low and the rising edges are actually the falling edges.

## Trigger Timings

Notice that the time values that specify when the triggers occur are measured relative to the falling edge of the *next* RF Start trigger. Because our system is causal and does not know the future, it has to guess when the next RF start trigger will come. It does this by taking one over the Pulse Repetition Frequency (PRF), which gives us the Pulse Repetition Interval (PRI). This gives us the amount of time between the two RF start triggers. From that, we can simply subtract the time shown on the diagram to get the amount of time after a pulse a given trigger should occur.

## Interpulse Period

The interpulse period changes the timings of the triggers. When setting trigger timings, the interpulse period is the first thing you should consider. Short pulse is shown on the left side of the diagram, and long pulse is shown on the right.

## Discharge Trigger Ambiguity

In the diagram, the discharge trigger has a spacing labeled “Adjustable from 1.2 to 5.8  $\mu\text{s}$ ,” which I found very confusing. This has something to do with the processing delay of the IFDR. The reasoning isn’t that important for this project, but the values are. The actual values at the time of writing are 2.24 $\mu\text{s}$  for short pulse and 4.48 $\mu\text{s}$  for long pulse

## Software Clarification

The software is commented to an extreme degree, but it still could be unclear what my intentions were with the various functions.

### Keypad

The Keypad.h library included in the software is supposed to be able to read from a keypad without blocking anything else in the code. It also includes debouncing. When you call `keypad.getKey()`, the library returns the character from the keymap matrix at the location of the key that was pressed. So if the key in the *i*th row and *j*th column were pressed, the function returns the character in `keymap[i][j]`. After getting this value, it should be passed to `keyToString()`, which switches the character to a string and also adds the function name for function buttons. After this, if you're actively outputting the characters being typed, pass the strings to `concatenateKeys()`, which will combine the strings into one string.

### Menu

The menu class is like a state register and state transition logic for menus. The syntax looks a little odd, but it's pretty simple. Each state is a member of the datatype `ID`, and the state transitions are called with the `enter()` and `back()` functions. To see which menu you're in, just call the `id()` function. If you want to change the names of the menus, just change the enum list at the top of the class.