Project Setup

I will try to explain the project I am working on a little more on a little more in depth. A couple months ago I asked somewhere on TI (not the support forums) if it was possible to output the values, I was reading from the TDC1000 to somewhere else. I asked if I could do this by configuring the GPIOs of the c2000 to outputs and write them onto an Arduino and see the values change in real time via serial monitor.

I started by hooking up two piezo ultrasonic transducers across from each other. One transducer was connected to TX1 for the transmitting signal and the other transducer was connected to RX2 for the receiving signal. There are three different Applications you can use for the TDC1000:

- **Application 1: Fluid Level**
  - Uses 1 transducer to send a signal back and forth. The time-of-flight is calculated to see how far away the top of the liquid is from the transducer.
  - Mounted on the bottom of the container
- **Application 2: Fluid Identification/ Concentration**
  - Can use either 1 or 2 transducers. The time-of-flight is calculated to identify what type of fluid is in the container.
  - Mounted on the sides of the container.
- **Application 3: Flow Meter**
  - Uses 2 transducers. Both transducers are excited and sends the difference of transmit time propagation.
  - Mounted non-intrusively side by side.

More details of all these applications can be found below:

https://www.ti.com/lit/an/snaa220a/snaa220a.pdf?ts=1598978916649&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FTDC1000-C2000EVM

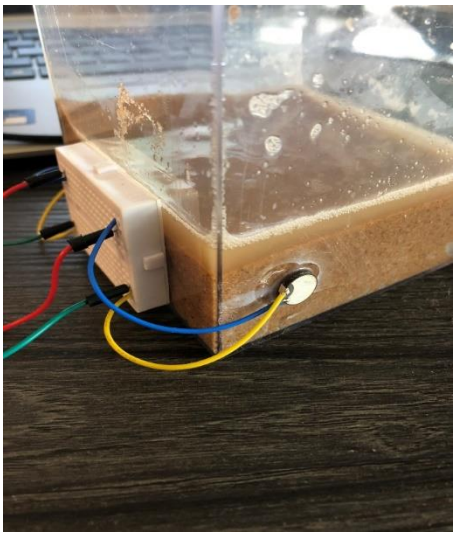

Figure 1 - transducer placement

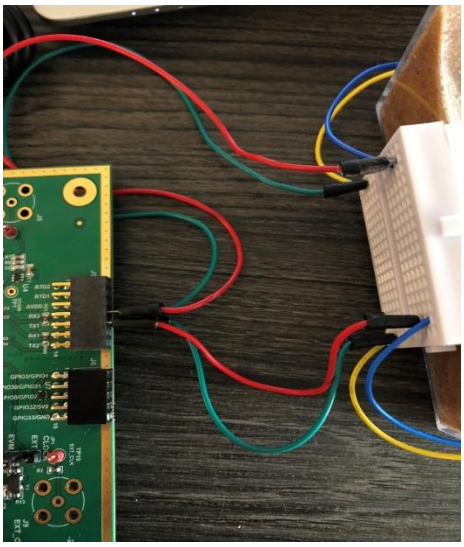

Figure 2 – wire setup

Figure 1 and 2 show how I have my setup working. I messed around with the GUI software switching between application 1, 2 and 3. In the TDC1000_C2000_EVM software where you can read the time of flight values, there are three different modes (TOF_MEAS_MODES) as shown in figure 3.

- Mode 0: Fluid level and identification. Transducer transmits then receives its own signal.
- Mode 1: Flow sensing. Transducer 1 transmits and transducer 2 receives. Requires manual channel switching
- Mode 2: Flow sensing (preferred). Transducer 1 transmits and transducer 2 receives. Allows automatic channel switching

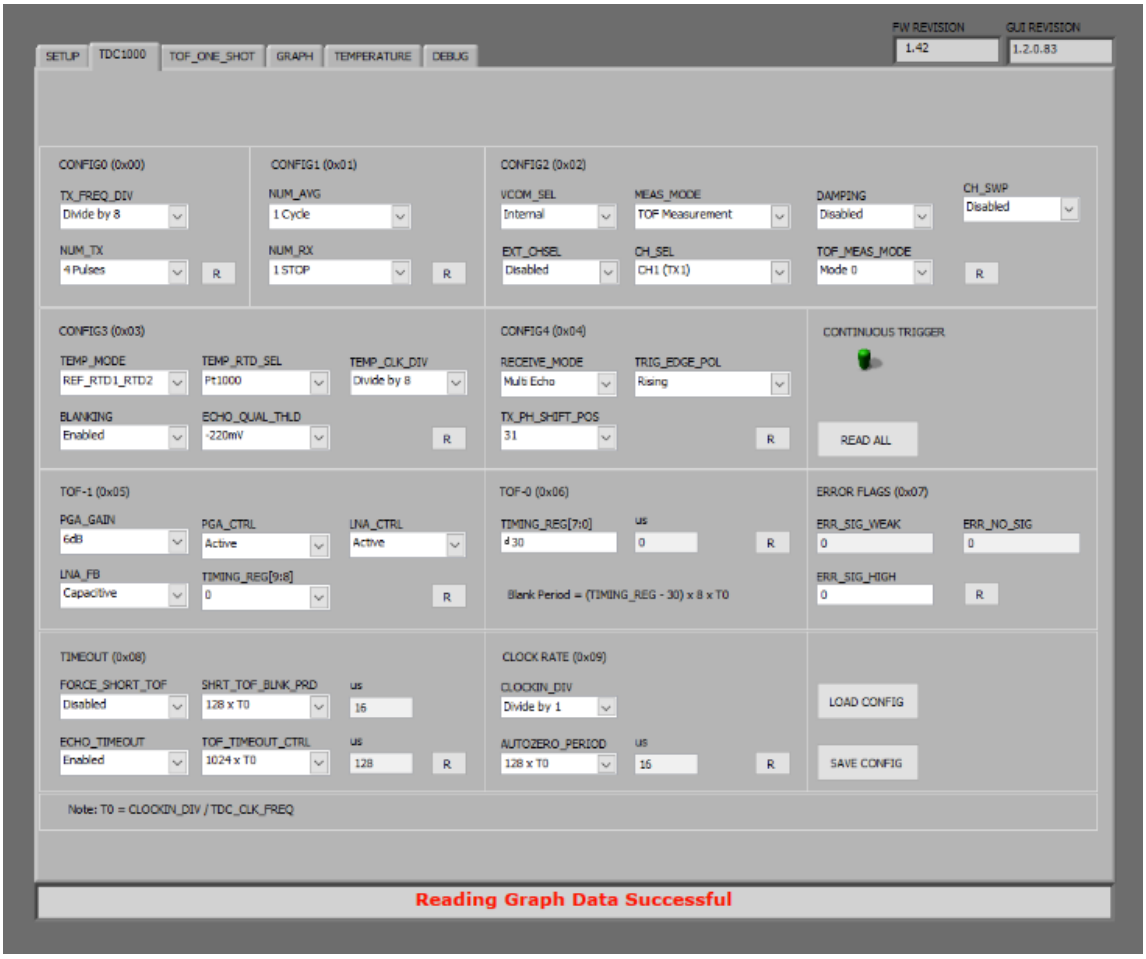

Figure 3 – TDC1000 Configuration

I then went to the graph to read the values of the Time of Flight. For the project, I wanted to read when there was a difference between air and sand + water. Hypothetically there should be a difference in the time of flight between air and sand + water due to it taking longer for the signal to travel through sand and water. This turned out to be true and could tell an obvious difference. Figure 4 shows what the graph is like with a steady TDC AVG value. It is not shown in the graph below but when sand and water are added, there is a sharp slope up. Figure 5 shows a separate occurrence of an excel graph for when I ran a test run going from air to sand + water.

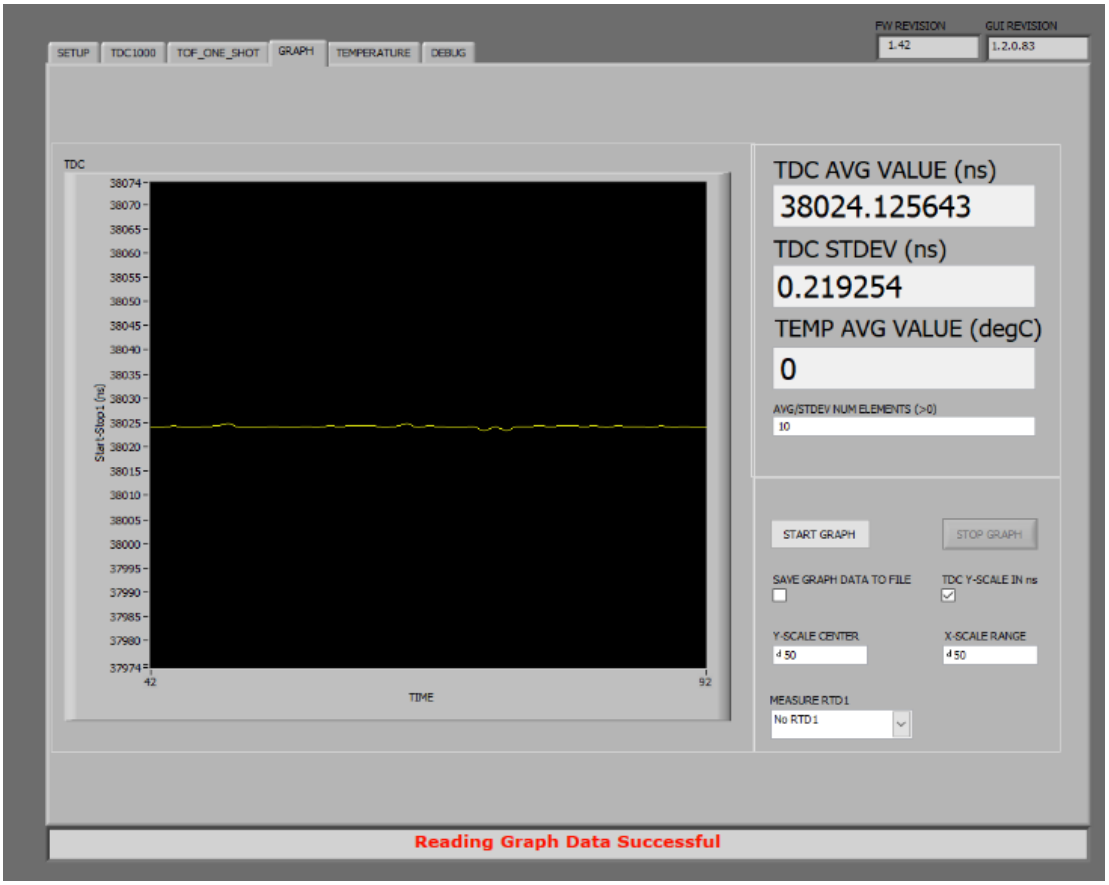

Figure 4 – TDC1000 graph no change in medium

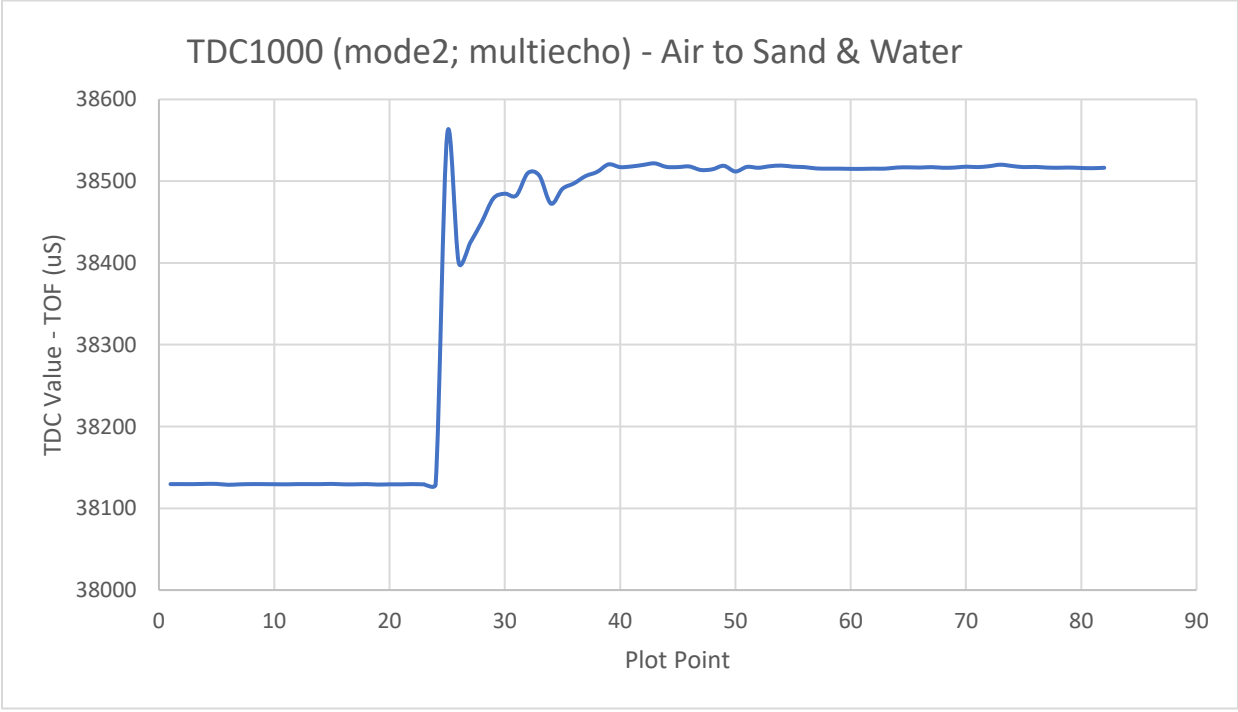

Figure 5 – TDC1000 excel test run air to sand & water

The only way to get these values in excel is to start the graph, let it run and then end the graph. The values are then saved to an excel graph with each individual time sample. I then put the values into a graph to make it look cleaner hence figure 5.

Going on from here, I wanted to be able to read this TDC AVG Value that the graph is reading in real time outside of the GUI software and excel. This led me to explore the TSM320f28035 microcontroller on board with the thought that I could read the TDC AVG Value through that microcontroller via connecting to the GPIO pins with J6 as shown in the red box in figure 6. I asked TI a while back if this was possible and they said it was.

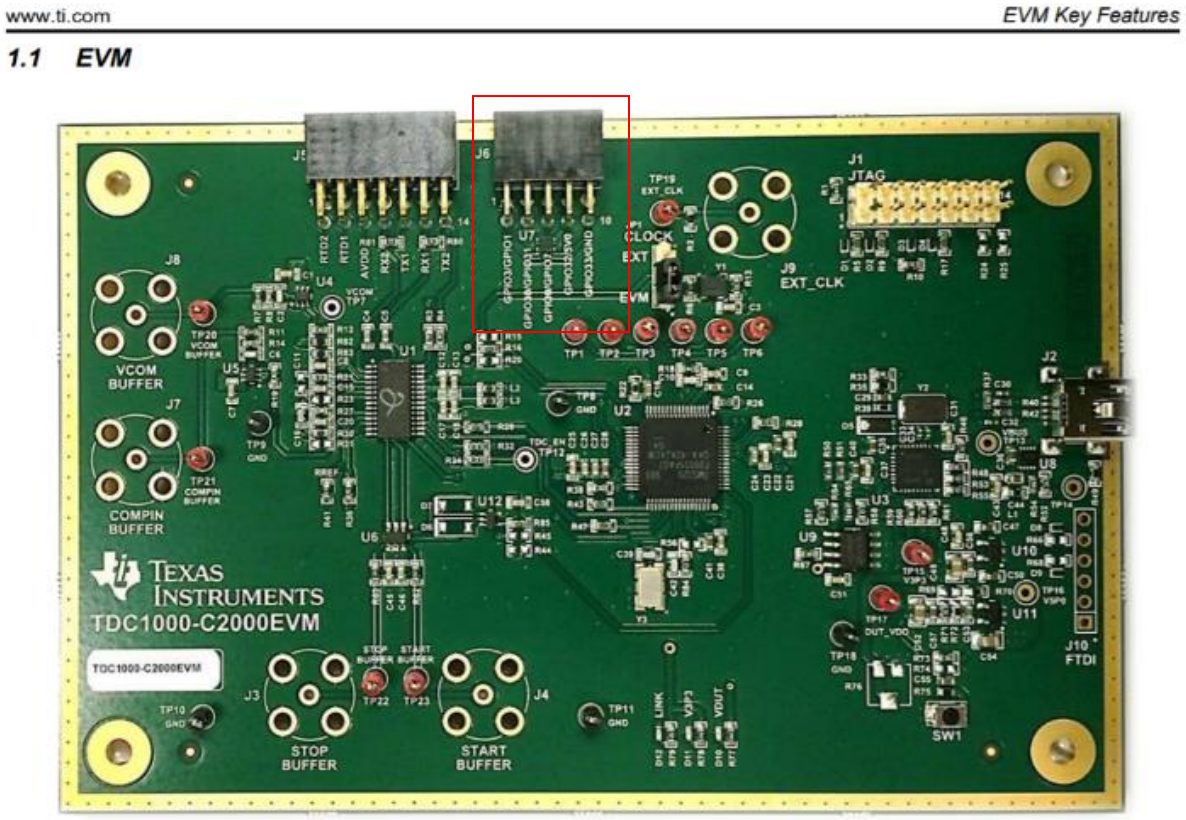

Figure 6 – TDC1000-C200EVM Evaluation Board

From here I started looking up the different GPIOs available to connect to through J6. My assumption was that the GPIOs could be set to outputs and that it could write the TDC AVG Value somehow. The GPIOs that you can connect to via J6 are GPIO0, GPIO1, GPIO3, GPIO30, GPIO31, GPIO32, and GPIO33. I then went to the TMS320F2803x Microcontroller datasheet to evaluate the different descriptions that these GPIOs could be se to.

https://www.ti.com/lit/ds/sprs584n/sprs584n.pdf?ts=1598988872881&ref_url=https%253A%252F%252Fwww.google.com%252F

Below shows what type of communication the output GPIO0 and GPIO1 can be set to.

GPIO AND PERIPHERAL SIGNALS [2]					
GPIO0				I/O/Z	General-purpose input/output 0
EPWM1A	69	56	49	O	Enhanced PWM1 Output A and HRPWM channel
–				–	–
–				–	–
GPIO1				I/O/Z	General-purpose input/output 1
EPWM1B	68	55	48	O	Enhanced PWM1 Output B
–					–
COMP1OUT				O	Direct output of Comparator 1

Figure 7 – GPIO Configuration

In my last post with you, I talked about channel A and B but meant to put Output A and B. I was a little confused on what the differences between A and B were, but I think you explained a little bit in your last response. What I am a little confused about is if I am setting these outputs correctly. For GPIO0, I want the output to be EPWM1A - *enhanced PWM1 Output A and HRPWM channel*.

Below are some snippets of the code of how I thought the setup should be to do this. I am not sure if this makes sense because some of the GPIOs have multiple outputs and if you set one to an output with multiple outputs then how is it going to know which one? **This leads me to the question for GPIO1 – Do I need to instead of write GPIO1, write EPWM1B somehere in the code? The same goes for EPWM1A (GPIO0) and all the other GPIOs.**

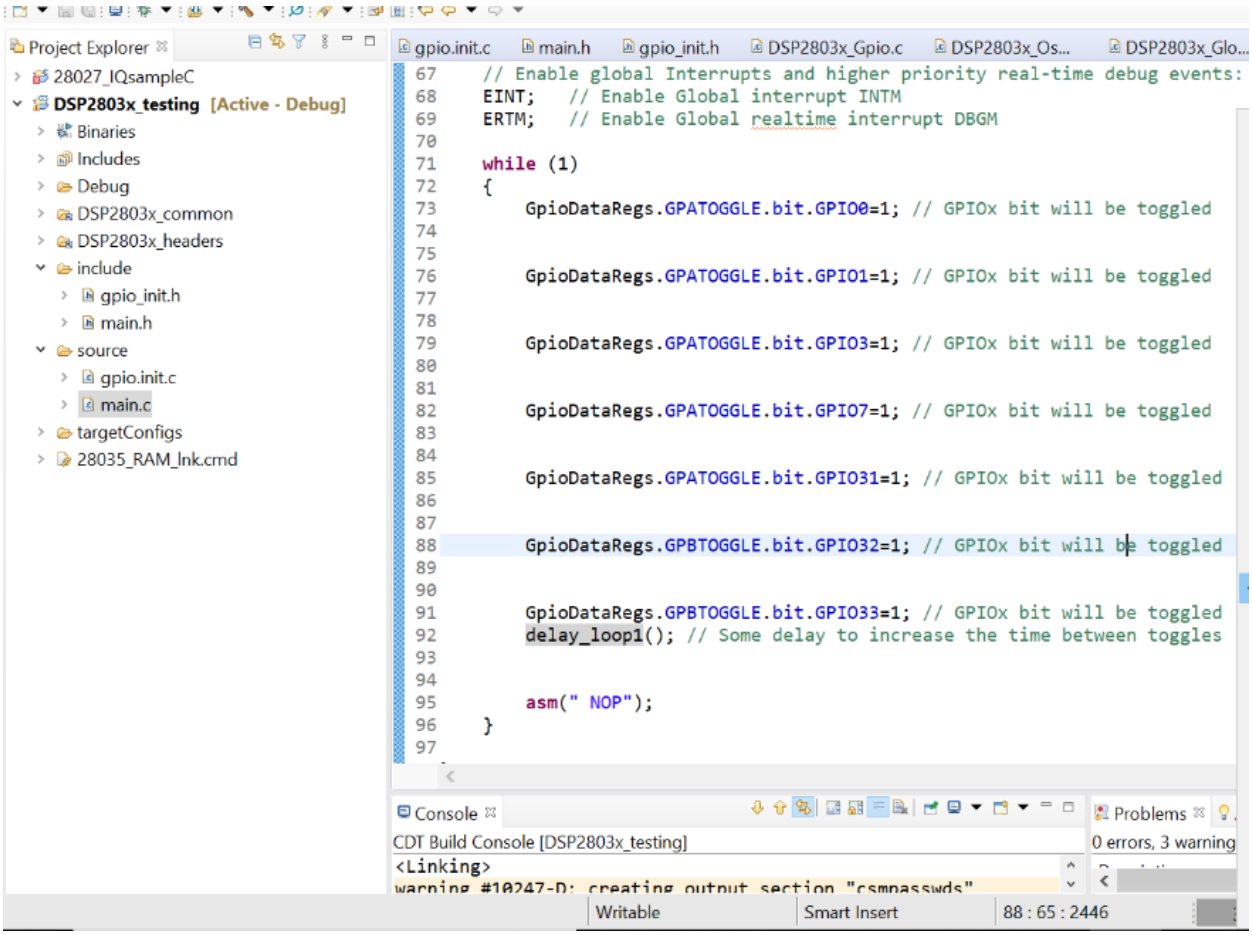

Figure 8 – Code Composer main code

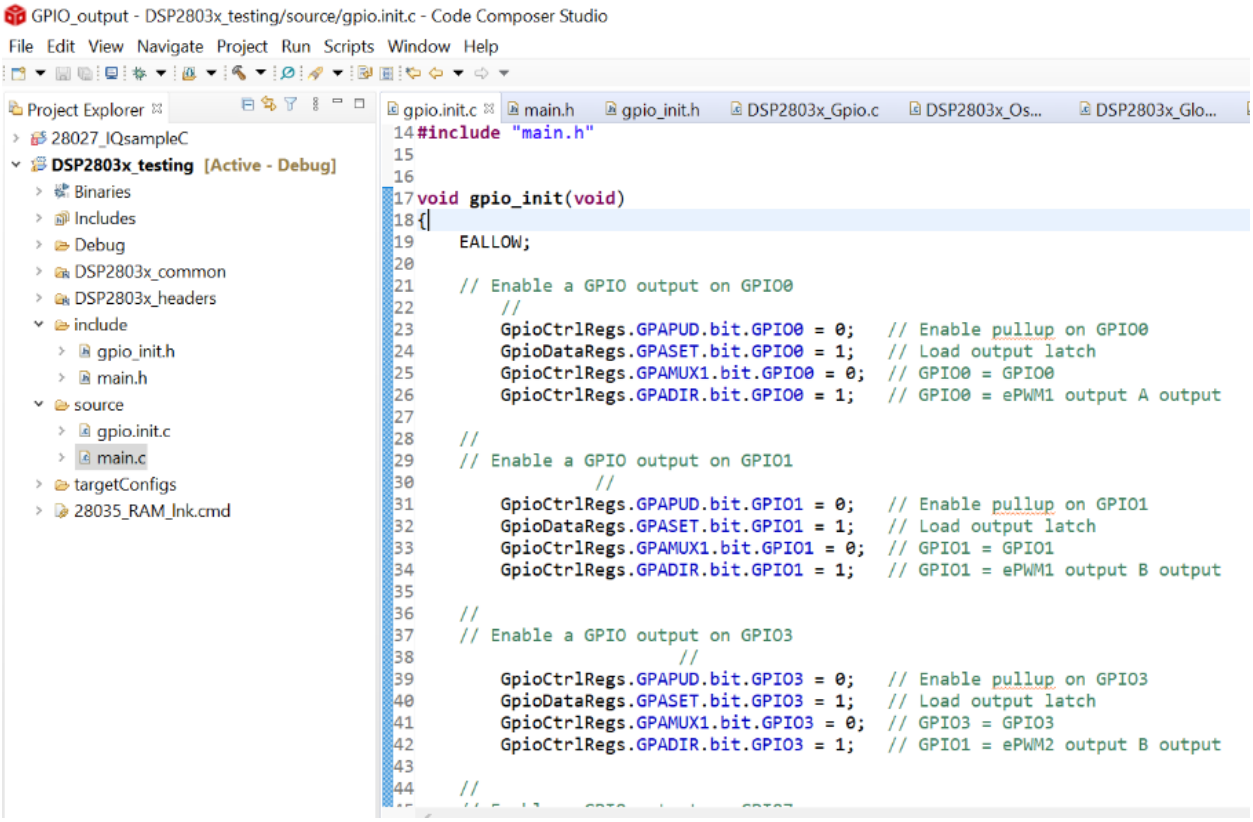

Figure 9 – Code Composer GPIO initialization code

As of now, I have the J6 (the one that connects to GPIO pins) connected to the Arduino. The Arduino has input pins that can read PWM signals. The Arduino software also has a serial monitor capability to where it can print values in real time. I start to read values once I debug and run the code in code composer. I am mainly using the Arduino to test and read these values in real time since I do not know how to do that in code composer. The values don't change when I remove the transducer from TX1, so it doesn't seem to be that I am reading values of the TDC AVG (TOF) or TX1 in that matter. The values do go to zero on the other hand when I comment out the code for any of the GPIOs. I am reading something, but I am not sure what these values mean and if they're related to the TDC1000.

The goal is to be able to write that value out from the GPIO pins. I understand that the TDC1000 and the C2000 are two independent systems. This leads me to a couple questions:

1. After explaining my project a little more in depth, is what I am trying to do still possible?
    a. How do I get the TDC1000 and the C2000 to communicate to each other to output the TDC AVG (TOF) value onto a GPIO pin via PWM or another form of communication?
2. For the GPIO pins, do I need to change the name in the code somewhere so it knows to be that specific output?
    a. For example; GPIO1 has output options of EPWM1B and COMP1OUT. How does it know which specific output to be set to?