


## Lab 2: Logic Analyzers and Monitoring Digital Signals.

Objective: Familiarization with the Digilent Logic Analyzer.

## Description of Lab 2:

Often the number of signals that we will need to monitor becomes too numerous, and they vary too quickly for us to use a logic probe to monitor them. In this situation we will need to use a device known as a Logic Analyzer (LA). This device is an electronic instrument that will monitor the digital value on a set of logic lines and record/display them on a set time basis. The lab is equipped with Digilent Analog Discovery (<https://www.digilentinc.com/data/products/analog-discovery/AnalogDiscovery.pdf>) which is a combination analog oscilloscope and logic analyzer. We will be using the LA portion of this system, which is set up to monitor digital lines.



On the computers in the lab, you will see an icon that looks like . If you click on this icon it will bring up a window like that shown in Figure 2-1. Note for this lab we will want to select the Analyzer button, under the Digital portion. This will bring up the window seen in Figure 2-2.

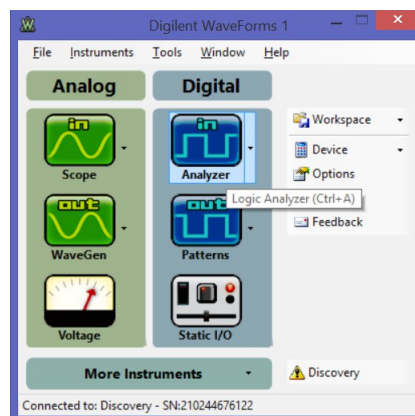


Figure 2-1. Digilent Start Up Window

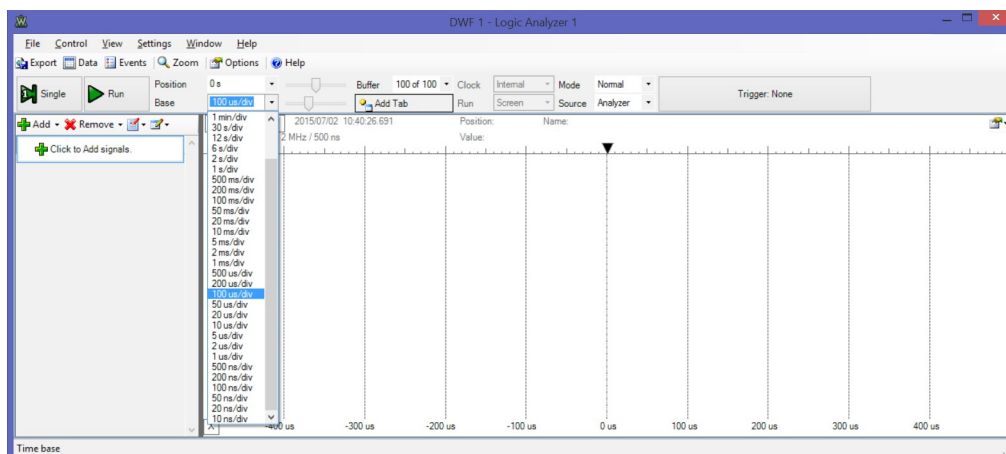


Figure 2-2. LA Start Up Window.

On this window there are three things that need to be set every time we use this application, and not necessarily in this order. The time base will be something that needs to change, based on the character of the signals that we will be monitoring. Note in this lab we will be looking at signals that change on a 10 and 20 millisecond intervals, so a time base of 10 ms/div.

Also important are the input signals, note it is entirely up to the user to choose which lines are attached to your circuit, but once connected they need to be selected from the list. Figure 2-3 shows an image of the LA, where the numbers along the bottom of the device identify the line number of each wire. Note the wires are color coded to aid in tracing them.

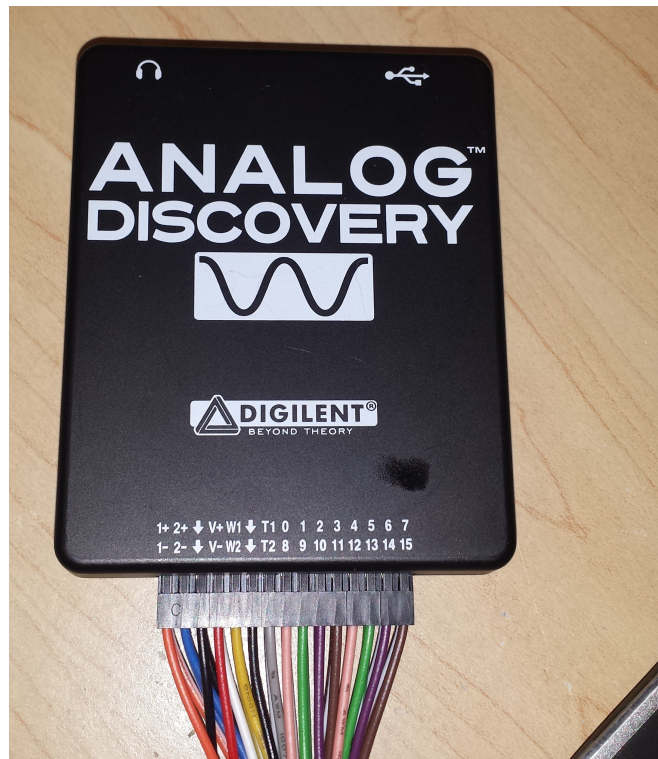


Figure 2-3. LA Hardware.

Once the wires are connected, the lines need to be selected. This is accomplished by clicking on the green + on the left of the LA window. Clicking on add signals, brings up the window in Figure 2-4. Hopefully this window is self explanatory.

The last thing that needs to be set is the trigger. As the LA is monitoring the signals it can't just be displaying all the data, that would just be confusing. Rather it needs to watch the signals and only display them when a certain event occurs, such as a signal changing or a signal being high or low. Once you have the signals selected you will see a column on the window marked as Trigger. By right clicking on this column by the signal, a set of options will be displayed. Among the options are rising edge, falling edge, change, high, low and don't care (X).

In the end, the window should look like that in Figure 2-5.

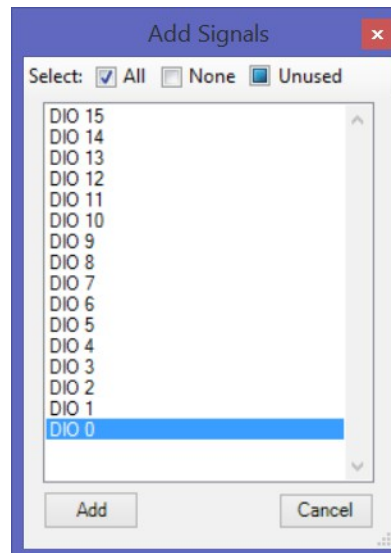


Figure 2-4. Add Signals Window.

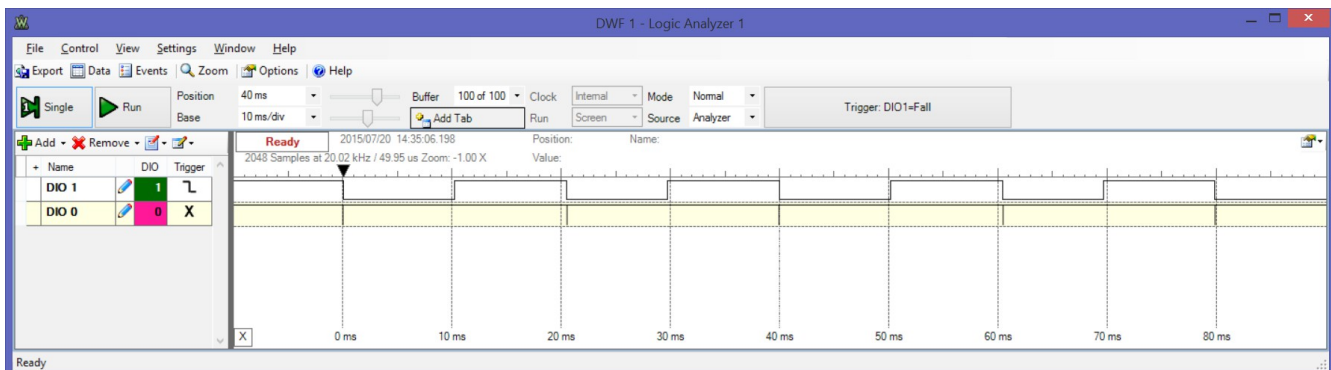


Figure 2-5. LA in Final Set Up.

Although we have tried to be complete in describing the process needed to set up the LA, there are many details that would just be tedious to write. However a set of in-class demonstrations will be done and hopefully prepare everyone to use the system in the laboratory.

## Lab Assignment:

Prelab: Rewrite the second program from lab 1 to have the two pins toggle at 20 and 10 milliseconds. Then upload it to the website.

## Lab 2:

- 1) Upload the program from the prelab to the arduino and attach the LA to the two pins and capture the output. Note you will need to set the time divisions such that you can measure the 20 and 10 millisecond pulses. ***Be sure to check this off with you instructor and capture the display, including it in your report.***
- 2) Change the program such that the 10 milliseconds pin will be pulsed low and then back high, as quickly as possible every 20 millisecond rate. As a first attempt, simply use the digitalWrite function to set the pin low and then high. Capture the display and include it in your report.
- 3) Repeat part two, except this time pulse the pin by directly access the port using the variable PORTB. Code for this would look something like.

```
PORTB &= ~0x10; // Force bit 2, Pin 12, low  
PORTB |= 0x10; // Force bit 2, pin 12, high
```

Questions: How accurately does your measured rate for pulses on the two pins match the desired for the program?

How different is the timing on the pulses from parts 2 and 3? Which is faster and why?