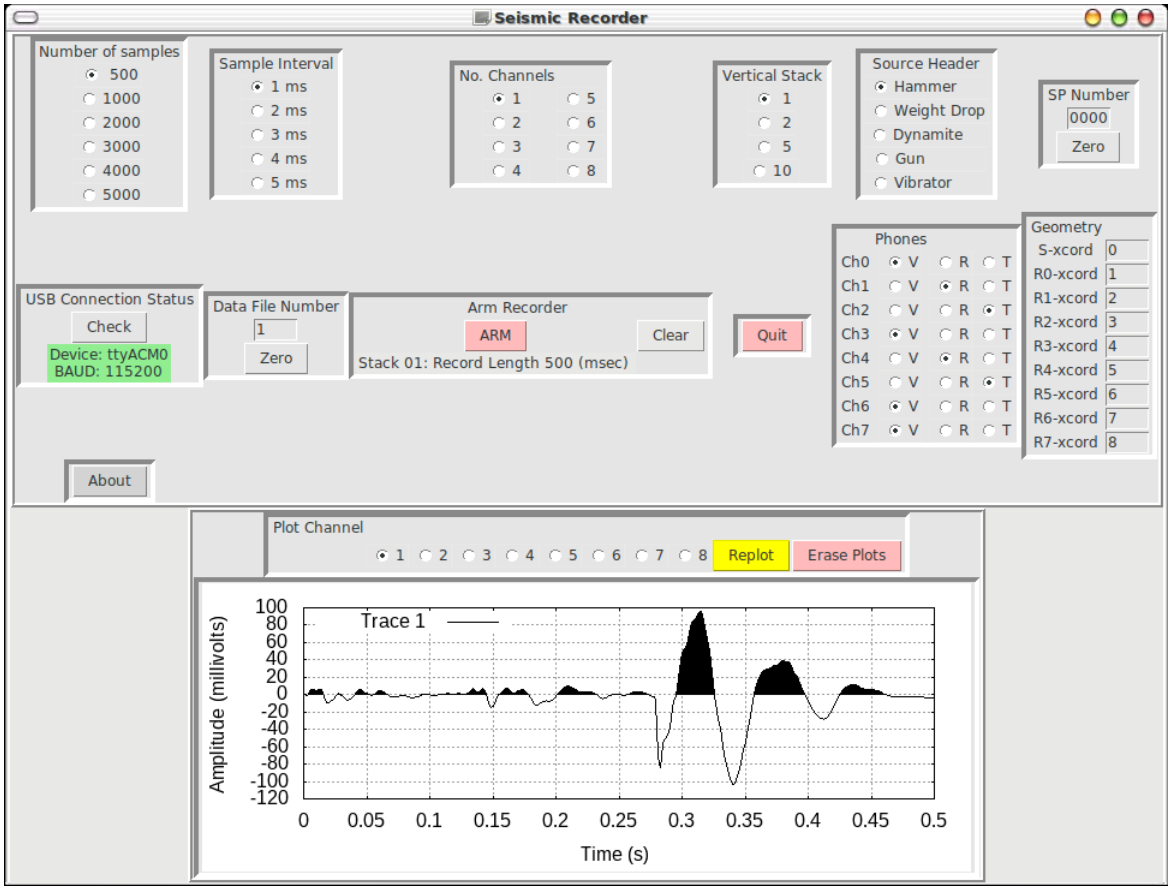
Build an Arduino-Mayhew Seismic Recorder

Dr. P. Michaels, PE
<pmsolid@cableone.net>
<paulmichaels@boisestate.edu>

September 15, 2020
Version 1.0.1



Michaels Engineering Geophysics
Boise, Idaho

Contents

List of Figures

1 Introduction

While there are a number of excellent engineering seismographs on the market, they are expensive. These instruments are designed to satisfy a number of survey techniques which often include refraction, reflection, surface wave, and bore-hole surveys. The required complexity of the design results in costs measured in thousands of dollars per channel. This project is designed to satisfy a narrow range of applications, surface wave and bore-hole surveys. Design goals include reducing complexity and cost. The result is a project suitable for students and those getting started in engineering geophysics. See section 6 for refinements of the original version of this document.

1.0.1 Design Components

The components of this project include the following:

- **Arduino microprocessor board**. While one might also consider a Raspberry Pi for this project, the multi-process feature of the Raspberry Pi makes millisecond sample interval accuracy unattainable. The Arduino does just one task without competition from other interrupts. The Arduino Uno cost was about $20.
- **Mayhew Extended ADC Shield**. The analog to digital converter (ADC) shield snaps directly on top of the Arduino board and provides 4 differential or 8 common ground channels. This project provides the 8 channel instance. The Mayhew shield comes in a number of precisions (12 bit, 14 bit, and 16 bit). This project uses the 16 bit shield. One can use this project's software to vertically stack to higher higher precisions, up to 32 bit if you have the time. The shield cost was about $40.
- **Two Pole Anti-Alias Filter, K-Gain of 10**. A separate printed circuit (PC) board is wired with point-to-point soldering. Four TLC272 Operational Amplifier chips are wired as eight Sallen-Key low pass Butterworth filters. The cut-off frequency is 100 Hz (-6 dB_v below the low frequency gain of 10). The TLC272 Op Amps are very low power, and run off the Arduino's 5 Volt supply. A virtual signal ground at 2.5 Volts is formed by a resistor voltage divider shunted with capacitors to prevent parasitic oscillations from forming. The cost for the Op Amps, resistors, capacitors, PC board would be about $30. You might have a lot of these parts in your junk box. See section 6 for refinements.
- **Inexpensive Lap Top Computer**. The above hardware are powered by a laptop via a USB cable. The lap top computer provides an LCD display, control software, serial communications software (USB), temporary memory, and solid state hard drive to save the recorded data in SEG-2 format (Pullan, 1990). The computer runs under the Linux operating system. As data are sampled by the Mayhew shield, the Arduino sends the samples as 16 bit binary integers on the fly, stored in the computer RAM memory (32 bits). When vertical stacking, each recording is summed in RAM using integer arithmetic. This increase the precision of the resulting sum. The SEG-2 format data are written as integers. The Dell laptop can run for up to 8 or more hours on a single charge while powering the acquisition hardware (solid state hard drives draw far less current than those that spin). The Dell Inspiron came with Windows 10. This was replaced with Ubuntu 18.04.1 LTS operating system running a Mate window manager. The processor is Intel Celeron(R) CPU N3060 at 1.60 GHz, 2 cores in a single physical processor, 2 threads. The graphics are 1366x768 pixels. RAM memory is 2G bytes with about 1GB free to save data on the fly. About 11 GB of disk are used for the OS, leaving about 15 GB to store data. The laptop computer cost was about $200.
- **Water Resistant Case**. Exclusive of the computer, the electronics are housed inside a Seahorse 300 case with O-ring sealed hinged lid ($21). A 16 gage piece of sheet metal was cut and drilled from a 12x12 inch piece (about $11 at Home Depot). The sheet metal provides a panel, screwed into the inside of the Seahorse case. After cutting holes for bulkhead connectors, switches, LED indicator, the panel was painted with 2 coats of Rust-Oleum 2X Hunt Club Green Satin spray paint ($4).
- **Software**. Software on the laptop includes:
  - **seisR.tk**. Tcl/Tk program that provides user interface.
  - **Status.cpp**. C++ program that checks USB connection and sets serial parameters. Called by seisR.tk when status button clicked.
  - **SEIS16v2.cpp**. C++ program that receives captured data and writes files in SEG-2 format . Called by seisR.tk when Arduino receives trigger (pin 2 drops to ground).

1.0.2 Compromises

On the plus side, commercial engineering seismographic recorders will have more channels, faster sample intervals, greater precision in the ADC, variable gain and filters. These features permit the tool to serve a wider range of applications, recording conditions, and energy sources. On the negative side, commercial recorders will cost more and require more power to operate. This means 12 Volt car batteries or some gel cell external batteries. They also will run using some form of Microsoft software (DOS or Windows).

A project like this is both educational and inexpensive. The decision to transmit data on the fly to the laptop RAM both limits the sample rate and introduces overhead that needs to be compensated for (as opposed to adding RAM to the Arduino). This is compensated for in the calibration (see section 4).

2 Design: Anti-Alias Filter, K-Gain

When digitizing an analog signal, there is the risk that high frequencies may become aliased. One limits that risk by filtering the analog signal with a low pass filter. The Nyquist frequency is the highest frequency that can be sampled without becoming aliased and falsely appearing to be a lower frequency. The Nyquist frequency is given as

$$F_{nyq} = \frac{1}{2 \cdot \Delta T} \quad , \tag{1}$$

where ΔT is the sample interval. This project permits the user to select sample intervals in the 5 to 1 ms range ($.001 \leq \Delta T \leq .005$ seconds). The largest sample interval poses the greatest risk. In this case, that would be 100 Hz. A rule of thumb is often to use 1/2 of the Nyquist. However, since most of my planned sources and sample interval would be less risky than this worst case, I decided on a cut-off frequency of 100 Hz. Most planned hammer-soil sources radiate energy at less than 100 Hz, and the sample interval most likely to be used would be 2 ms (250 Hz Nyquist). Of course, other noise might exist in the environment, so one is never safe. It is a matter of risk management. Avoid working in noisy environments and employ common sense.

2.1 Sallen and Key (VCVS Low Pass)

I decided on a 2-pole Butterworth filter of the Voltage Controlled Voltage Source (VCVS) topology. The Laplace transform is given by

$$\frac{V_2}{V_1} = \frac{K b_o}{s^2 + b_1 s + b_o} \quad , \tag{2}$$

where the left hand side of equation 2 is the ratio of the output to input voltage, K is the gain, s is the Laplace transform variable, and the constants b_o, b_1 determine the type of filter (Butterworth, Chebyshev, etc.). For a 2-pole Butterworth filter, $b_0 = 1$ and $b_1 = \sqrt{2}$. The topology of the filter is shown in Figure 1.

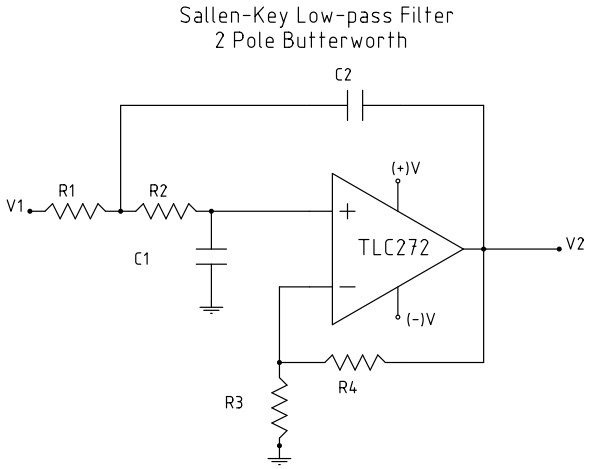


Figure 1: Low-pass Butterworth filter. Gain controlled by R3 and R4. If $R3 = \infty$ and $R4 = 0$, then unity gain.

2.2 Design Procedure VCVS Filter

In this discussion, "G" will be conductance of resistors "R". Thus, $G_1 = 1/R_1$. We will use a normalization procedure with a cut-off frequency of $\omega = 1\ rad/sec$ (Tedeschi, 1979).

1. We start with $C_2 = 1F$.
2. To keep it simple, make $C_1 = C_2$.
3. We choose gain, $K = 10$.
4. We calculate conductance G_1 as,
$$G_1 = \frac{b_1 + \sqrt{{b_1}^2 - 4.0 \cdot b_0 \cdot (C_1 + 1 - K)}}{2}. \tag{3}$$
5. We calculate the conductance G_2 as,
$$G_2 = \frac{C_1 \cdot b_o}{G_1}. \tag{4}$$
6. We calculate the resistor values $R1 = 1/G_1$ and $R_2 = 1/G_2$.
7. We compute the frequency normalization factor $U = 2\pi \cdot f_c$, where f_c = 100 Hz, our desired cut-off frequency.
8. We compute a starting Impedance Scale Factor, $ISF = f_c/(20\pi)$. This will be adjusted by trial and error until we get resistors and capacitors that fall in the range of convenient values. That is,
   - For resistors, $R_{convenient} = R_{norm} \cdot ISF$
   - For capacitors, $C_{convenient} = C_{norm}/(ISF \cdot U)$.
9. For this project, the ISF was found to be $ISF = \frac{f_c \cdot 100000}{(20\pi) \cdot 4.7}$.
10. The final values are
    $R_1 = 9.3K\Omega$
    $R_2 = 122.7K\Omega$
    $R_3 = 146.7K\Omega$
    $R_4 = 1.320M\Omega$
    $C_1 = C_2 = .047\mu F$
    Cut-off = 100 Hz, Gain = 10.0 (20dBv)
11. Since these are not all standard values, we round off to the following:
    $R_1 = 9.1K\Omega$
    $R_2 = 120K\Omega$
    $R_3 = 150K\Omega$
    $R_4 = 1.2M\Omega$
    $C_1 = C_2 = .047\mu F$
    Cut-off = 102.5 Hz, Gain = 9.0 (19dBv)

2.3 Connecting the Ant-alias with Gain to the Mayhew-Arduino

The op-amps are two to a chip. Figure 2 shows the how the first chip connects to the Mayhew extended ADC shield which is mounted on top of the Arduino. Note that there are two grounds. Pin 4 of the op-amp is the chassis ground, and the virtual ground for the signals from the geophones are at 1/2 of the supply voltage (see the voltage divider with 10K resistors and $0.1\mu F$ shunting capacitors which prevent oscillations from forming. Oscillations are also prevented with a 5.1K resistor shunt across the input. With the very high input impedance, leaving an input open resulted in oscillations forming when no geophone was connected. The typical geophone has a lower impedance, and is not significantly loaded by the 5.1K resistor. See section 6 for refinements.

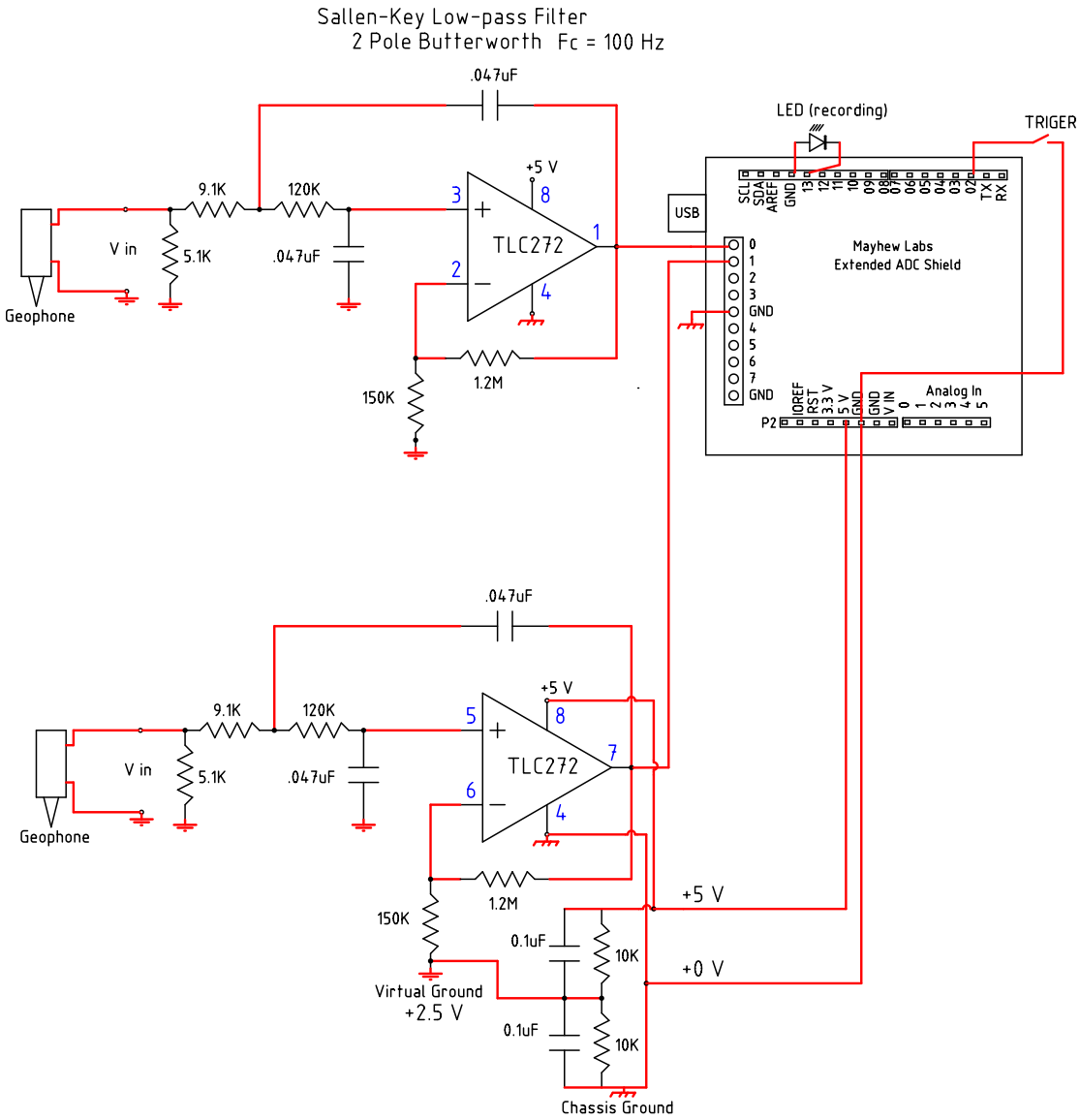


Figure 2: Low-pass Butterworth filters connect to the Mayhew shield. **See Figure 8 for refinements**.

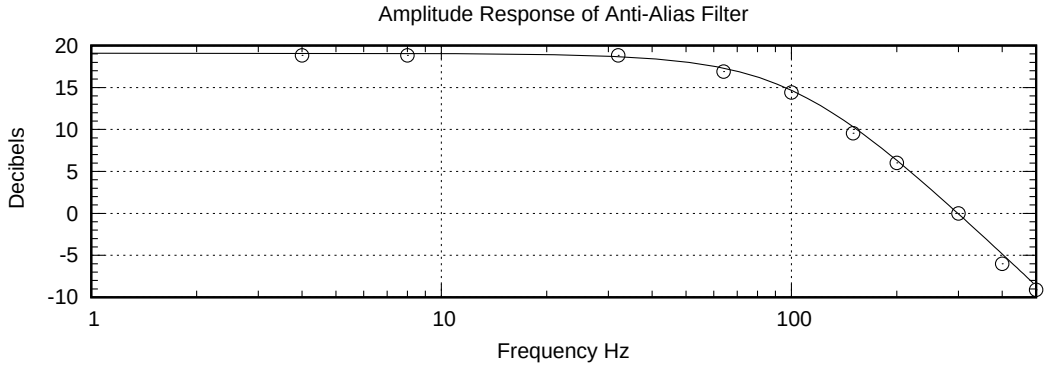


Figure 3: Measured and calculated amplitude response of anti-alias filter.

2.4 Wiring Filter/Pre-amp and Mayhew/Arduino

Figure 4 shows the point to point wiring of the anti-alias filter with gain. Resistors are stood on end to permit making connections on the top side of the board. Components are soldered on the bottom side (not in view). IC sockets are used for the TLC272 op-amps.

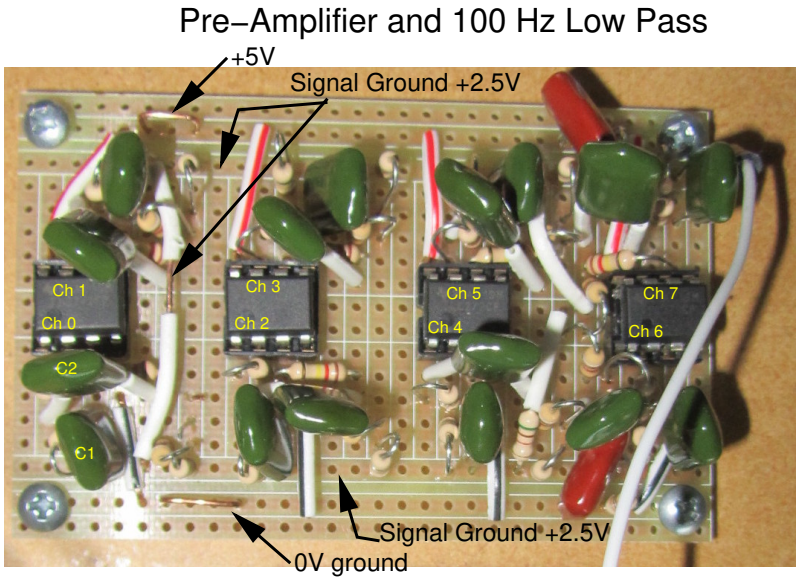


Figure 4: Wiring of the op-amps to form anti-alias filters with gain.

Figure 5 shows the wiring that connects the filter/pre-amp board to the Mayhew/Arduino board. The input signals pass from the banana plugs through a D-connector to the filter/pre-amp board. The D-connector also passes the LED signal (data recording) from pin 13 on the Mayhew to the LED mounted on the panel (not shown). The trigger signal is also passed through the D-connector. The Mayhew and Filter boards are screwed to a hardboard which is then later taped to the inside of the Seahorse water resistant container. The panel is then screwed to the Seahorse.

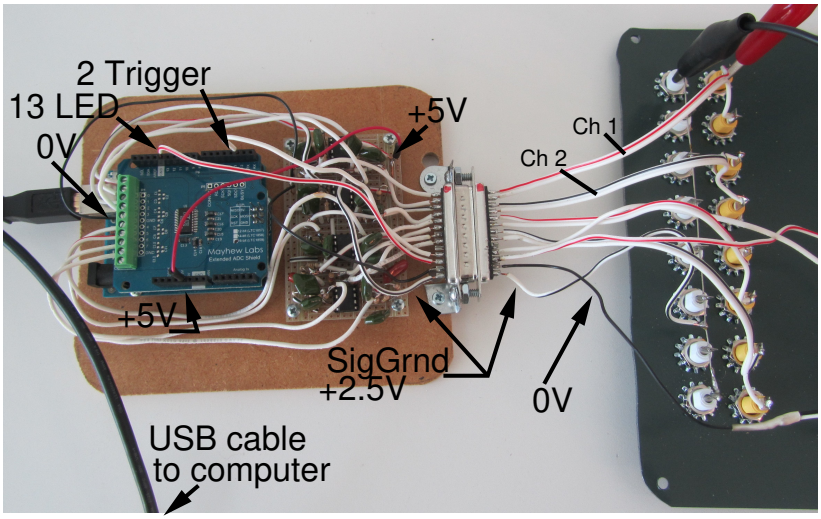


Figure 5: Wiring of filter/pre-amp board to the Mayhew and back of instrument panel. The Mayhew shield plugs directly over the Arduino.

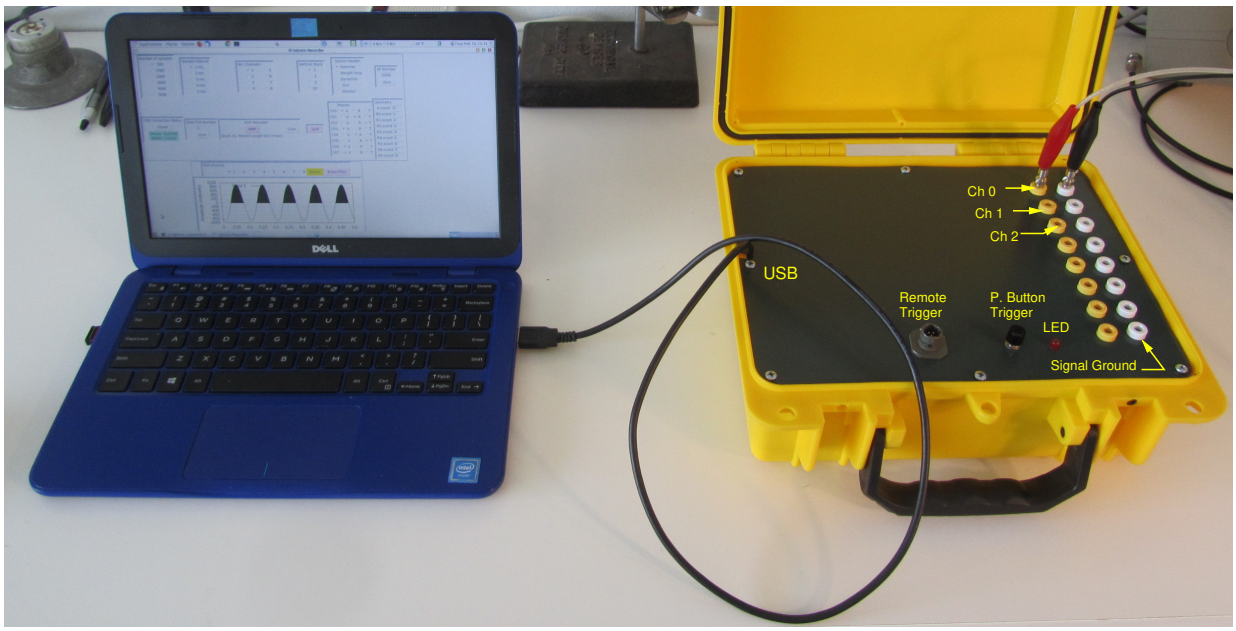


Figure 6: Final set up connecting the recorder to the laptop. The laptop software controls the recorder. There is no on/off switch on the recorder, it is on when plugged in with the USB. The LED will blink when connecting the recorder to the running laptop, and will stay lit when data are being collected. See Figure 10 for the revised panel.

Figure 6 shows the final set up connecting the seismic recorder to the laptop computer. The USB cable both provides power to the seismic recorder and captures the digital data on the fly when recording. **NOTE: This is important. Always check the status button on the software after connecting the USB to the computer.** This will set up the serial parameters for the USB. This can be done with the GUI or from the command line (see section 4.1.4).

2.5 Obtaining Components for Assembly

A number of online sources sell electronic, connector, and enclosure components. These include:

- **Arduino**. The Arduino Uno is the microcontroller board for the project.
  https://www.arduino.cc/
- **Mayhew Extended ADC Shield**. This analog to digital converter board plugs directly into the Arduino Uno. It comes in 12 bit, 14 bit, and 16 bit versions. This project software is configured for the 16 bit, but that is easily edited.
  http://mayhewlabs.com/products/extended-adc-shield
- **Digikey**. Good for banana plugs, sockets, and electronics.
  https://www.digikey.com/
- **Jameco**. Good for integrated circuits (like the op-amps), capacitors, resistors, PC circuit boards.
  https://www.jameco.com
- **Seahorse**. Good for the enclosure. They also sell panels, but I cut mine from 16 gage steel, obtained from Home Depot 12 in x 12 in.
  https://seahorse.net
  https://seahorse.net/shop/protective-cases/se300/

3 Laptop and Arduino Software

Software is installed in both the laptop computer and the Arduino. If you download the file, **SeisR-1.0.1.tgz**, you will not need to do a lot of typing. There is also a ZIP archived version for those less familiar with TAR and GZIP. It is **SeisR-1.0.1.zip**. The appendix contains listings of the source codes.

3.1 Prerequisites

The best idea is to use your operating system package manager to install these resources. Citations below are for books on these software packages. The Graphical User Interface, GUI is a **wish** program. Program **wish** is an interpreter and **seisR.tk** does not need to be compiled. But, you will need to install TCL and TK on your computer (Ousterhout, 1994). You will also need to be able to compile the C++ programs, so a compiler is needed. I recommend the GNU compiler (Various, 2019). You will also need to install Gnuplot for the graphics (Janert, 2010).

For converting from SEG-2 to other formats, you may wish to install my Basic Seismic Utilities (BSU) package (Michaels, 2018). The BSU package has binaries and source code, is open source, and can be run in Linux, Microsoft Windows, and Apple's Macbook-Pro. It is all included in a single ZIP archive, **BasicSeismicUtilities_PaulMichaels.zip**. Specific subsets are also available on the web if you don't want to download it all. There are many applications in the BSU package. Program **egg2seg** will convert a SEG-2 formatted file to BSU's processing format (bsegy). Program **seg2txt** can be used to go from bsegy to ASCII text format (perhaps for Matlab or Octave processing).

3.2 Compiling Software

There are Make files for compilation. If you download my file, **SeisR-1.0.1.tgz** or **SeisR-1.0.1.zip**, extract the archive. The best place to do this in Linux is from `/usr/local/src`. Type the following:

```
tar xvzf SeisR-1.0.1.tgz
cd SeisR-1.0.1
```

Next are the steps to follow:

- ```
  make
  ```

  If you have all the compiler packages installed, the response will be
  `g++ -g -Wall Status.cpp -o Status`
  `g++ -g -Wall SEIS16v2.cpp -o SEIS16v2`
  And you will have two new executable files, **Status** and **SEIS16v2**.

- ```
  make install
  ```

  If you have not edited the Makefile PREFIX, the response will be
  `install Status /usr/local/bin`
  `install SEIS16v2 /usr/local/bin`
  `install seisR.tk /usr/local/bin`

- ```
  cd Arduino
  cd A16
  ```

  With the USB cable connecting the Seismic Recorder and Laptop, do the next step:

- ```
  Status
  make
  make upload
  ```

  If all goes well and after a bunch of output, the binary will be compiled and then uploaded to the Arduino inside the Seismic Recorder. Because we are using optimization level 3, we don't use the IDE.

4 Calibrating the Seismic Recorder

The sample interval may need to be calibrated. There will be some computational overhead that may vary with the speed of the laptop and the connection of the Arduino to the laptop via the USB cable. See section 4.1.4 for a command line alternative to GUI.

1. Boot the laptop and login.
2. Plug the USB cable into the laptop from the recorder.
3. Type `seisR.tk`

   This will start the controller program and provide the graphical interface. If it does not come up, you may need to fix your PATH variable. The default location for the executable is in `/usr/local/bin`.
4. **TEST THE CONNECTION STATUS**. Do this by clicking on the USB Connection Status "Check" button. It will either come up in red saying that there is no connection, or green background giving the device name and baud rate. If you don't get the green message, check to make sure you have done step (2) above.
5. Click on the 1000 sample button, sample interval 1 ms.
6. Click the ARM.
7. Trigger the recorder manually by the push button. You can connect a geophone and do a tap test during recording, or have nothing connected to channel 0. We are checking the timing.
8. After the data collection finishes, a message will appear below the arm button. Ideally, it should read `Stack 01: Record Length 1000 (msec).`

   It might indicate a different record timing length. If it displays a smaller value, say 994 msec, then we need to increase the sample interval in the Arduino code. If it displays a larger value, say 1005 msec, we need to decrease the sample interval. The variation is due to the computational overhead being different for different hardware. In practice, a value of 998 or 1002 is probably good enough (ie. 2 ms out of 1000 ms). But the explanation of how to adjust this will follow.

4.1 Editing Arduino Program A16.ino

The factor that needs updating is called **shft**. This is computed from a linear function. This is on line 120 of the A16.ino code. The problem is with the variables **slope** and **intercept** on lines 43 and 44 of the code. The variable **k** is the number of channels. We comment out line 120 with two slashes at the beginning of the line:

```
// shft = ceil((slope*(float) k ) + intercept); //shft = overhead
```

Then we insert a line above that will be temporary.

```
shft = 0;
```

We will conduct a trial and error adjustment of **shft** for the two cases that are as follows:

1. Number of samples = 1000, No. Channels = 1, Sample interval = 1 ms.
   This should produce a 1000 msec recording
2. Number of samples = 1000, No. Channels = 8, Sample interval = 5 ms.
   This should produce a 5000 msec recording

4.1.1 First Case 1 Channel

The initial value of **shft = 0** will result in a recording aperture that is longer than the desired 1000 msec for the first case.

1. Quit the seisR.tk GUI
2. Save the modified A16.ino file.
3. Recompile and Upload (`make` followed by `make upload`).
4. Start seisR.tk, set number of samples, channels, sample interval for case 1.
5. click on Status button.
6. click on ARM button, then trigger manually. Observe the temporal aperture. For example, it might display 1844 (msec) below the ARM button. This is larger than the desired 1000 msec.
7. Return to editing file A16.ino
8. Change **shft = 0;** to **shft = 800** (about 1844-1000).
9. Save the modified A16.ino
10. Recompile and Upload (`make` followed by `make upload`).
11. Start seisR.tk, set number of samples, channels, sample interval for case 1.
12. click on Status button.
13. click on ARM button, then trigger manually. Observe the temporal aperture. For example, it might display 1039 (msec) below the ARM button. This is larger than the desired 1000 msec but closer to the desired.
14. Quit the seisR.tk GUI
15. Return to editing file A16.ino
16. Change **shft = 800;** to **shft = 839** (about 1039-1000 added to 800).
17. Save the modified A16.ino
18. Recompile and Upload (`make` followed by `make upload`).
19. Start seisR.tk, set number of samples, channels, sample interval for case 1.
20. click on Status button.
21. click on ARM button, then trigger manually. Observe the temporal aperture. For example, it might display 1000 (msec) below the ARM button. This is the desired temporal aperture. GREAT! Now move on to the next case and repeat the procedure for 8 channels.

4.1.2 Second Case 8 Channels

Following the same procedure as above, but with 8 channels and 5 ms sample interval we might see the following:

- **shft = 0** with a result of 9063 msec. This is way larger than 5000.
- **shft = 4000** with a result of 5038 msec. Why 4000? This is about the difference between 9063 and 5000.
- **shft = 4038** (5038-5000 added to 4000) with a result of 5000 msec. GREAT! We can now compute the slope and intercept needed to make line 120 give the desired **shft** value. We solve two equations in two unknowns as follows.

4.1.3 Solving for slope and intercept

The two equations are:

$$slope \cdot 1 + intercept = 839 \tag{5}$$
$$slope \cdot 8 + intercept = 4038 \tag{6}$$

Subtracting equation 5 from equation 6 gives

$$slope(8-1) = slope \cdot 7 = 3199 \tag{7}$$

We solve for slope

$$slope = \frac{3199}{7} = 457.0 \tag{8}$$

Now we can use either equation 5 or 6 to solve for the intercept. Let's use equation 5.

$$intercept = 839 - 457.0 = 382.0 \tag{9}$$

1. We remove the temporary line in A16.ino, delete the line **shft = 4038;**. This is following the above example, your value may differ from 4038 when you run your calibration.
2. We remove the comment double slashes from line 120. It will return to the following:
   `shft = ceil((slope*(float) k ) + intercept); //shft = overhead`
3. Then we can edit lines 43 and 44, or comment them out and replace with alternative lines. You would replace the slope and intercept values with those determined from your own testing.

4.1.4 Using the Command Line

Programs Status.cpp and SEIS16v2.cpp can be run from the command line in a terminal. Doing this can be faster than running the GUI when calibrating the time base. For example,

```
Status
```

The program responds typically as

```
Device: ttyACM0              or          Not Connected
BAUD: 115200
```

and if connected, sets the serial parameters. The command line help for SEIS16v2 is found by typing the command in a terminal,

```
SEIS16v2
```

```
USAGE: SEIS16v2 nsamp ndt nchannels nstack file source phones SP SX RX
Ex: SEIS16v2 1000 5000 8 1 0 H VRTVRTVV 0 5 1 2 3 4 5 6 7 8
source: H=hammer W=weight_drop D=dynamite G=gun V=vibrator
phones: V=vertical R=radial T=transverse (8 channels)
SP: Shot point number, SX Shot point X-coord
RX: Array of X-coordinates geophones
```

There are defaults for all parameters. You only need to type the first few parameters when doing the time base calibration. For example, sample interval is in microseconds, so 1000 samples, 1 millisecond (1000 microseconds), 1 channel, type:

```
SEIS16v2 1000 1000 1
```

Press the trigger button and it will respond:

```
Stack 01: Record Length 1001 (msec)
```

confirming a 1.001 second recording aperture which is satisfactory.

5 Collecting Data

There are no protections against excessive voltages on the signal inputs and trigger input. Adding protections might be a good idea. The following are recommendations for collecting data with the current version:

1. Deploy cables, geophones, and a source if one will be used. Connect the source trigger cable to the trigger input unless you are going to record ambient signals using the push button trigger.
2. Turn on the laptop computer.
3. Shutdown any services not needed on the laptop (ex. wireless, ssh, printer, apache web host, etc.) This is not essential, but why leave these on, they might lead to interrupts that could affect sample interval accuracy.
4. Connect the USB cable from the recorder to the laptop USB port.
5. Check the USB status with the **Check** button. If you don't get a green status showing the serial device and baud rate, check your connections or the trouble shooting section.
6. Click on the desired input settings:
   - Number of samples.
   - Sample Interval.
   - No. Channels.
   - Vertical Stack (this is how many source efforts will be added to a single file record.
   - Source Header.
   - Phone orientations. V=vertical, R=inline radial, T=transverse
   - Geometry (coordinates in meters of source and geophones. This is kept very simplistic (assume a straight line recording). If you have 3 component phones, then 3 R_i-xcord values can be the same. If you don't like this restriction, edit the **seisR.tk** and **SEIS16Rv2.cpp** files.

     NOTE: This is only for the SEG-2 headers, and if you keep an observer's log on paper (and you should!), you can do more complicated shooting with elevations and both x, and y coordinates. Geometry headers can always be edited later during processing.
   - Set the SP Number if needed.

     NOTE: This is NOT incremented automatically. You have to do it manually, it is just a label in the headers, has nothing to do with file numbers.
   - Set the Data File Number if needed. When the program starts, this is initialized to zero. But if you have been recording and already have a number of files saved on disk, you don't want to write over them. So set this to 1 more than where you left off. **The data file number increments automatically after each file is written**.

     NOTE: If your executables are in your path, you can create a directory to record the day's data. Change into that directory before you start seisR.tk, and then the data files will be stored in that directory.
7. Check to make sure all the geophone and trigger cables are connected.
8. Click on the ARM button.
9. Activate the source (contact closure trigger, ie. momentary short of trigger lines). For ambient recording, press the push button on the recorder. The red LED should light up during the recording, and go out once the recording maximum time has completed.

   NOTE: If you do a vertical stack of more than 1, the red LED is your communication that data recording for a source effort is completed. The ARM button will stay hot (bright red) continuously until all source efforts to be summed are collected. WATCH the red LED for the all clear to take the next source effort/trigger.
10. To terminate the seisR.tk program, click on the **Quit** button.

5.1 Trouble Shooting

- **Unexpected Long Recording** The software is fairly bullet proof and will try to avoid bad choices. For example, if you increase the number of channels, it may reset the sample interval to a larger value in the GUI. This is because there is too much overhead to permit a large number of channels to satisfy a short sample interval. It is possible to defeat this protection in the GUI, but the code **SEIS16Rv2** is a backup that will not be defeated. Depending on the order of your checking options you can create this effect. Say you click on 8 channels and then reset the sample interval to 1 ms. **SEIS16Rv2** will override your choice for sample interval. If you see the recording aperture has increased to a value larger than you expected, it is because the code reset the sample interval to a larger value, but has no way to tell the GUI about it. This will be obvious in the plot.

- **Status Won't Work**. OK, so the code **Status.cpp** (lines 46-55) is limited in the expectation of a device name. If your Arduino does not use either `/dev/ttyACM0` or `/dev/ttyACM1`, then you may need to edit the **Status.cpp** program for what your Arduino will use. Try using your IDE to figure this out. Either add an additional device name to look for, or edit one of the ones I have included.

- **Things are really SCREWED UP !!**. Right, this can happen if you forget to check the **Status** before ARM and recording data. For the worst case, quit the GUI (seisR.tk), unplug the USB, and start over. MAKE SURE you check status first thing next time.

- **The Plot didn't work**. Right, you might get an error message instead of a plot if you choose a plot channel that is beyond the number of channels being recorded. No biggy. Click OK and cancel the message. Then choose a channel to plot in the range of those acquired. Then click **Replot**.

  NOTE: If you are acquiring more than one channel, you can set the plot to show any valid channel. In fact, after looking at one channel, you can move on to another channel and **Replot**.

  Unless you click **Erase Plots**, plots will remain in the directory with names like `plot0.png`. If you switch to recording fewer channels at some point, the old plots will still exist, possibly leading to confusion (like how come there is a plot 7 when I only recorded 5 channels?).

- **Channel Order, Number of Channels**. Channels are added starting from channel 0 working up to channel 7. For example, if you choose to record 3 channels, these will be channels 0,1, and 2. **You can NOT just record channel 8 and no other channels**. If you want to record only 1 channel, it will be channel 0. This decision is implemented in the Arduino and the order it scans channels. You can change this, but you will have to edit **A16.ino** and recompile/upload.

- **OOPS, I plugged the signal input into a high voltage!!**. OK, if you blow the anti-alias filter op-amp, you can pull that chip out of the socket and replace it. Review the first comment under **Collecting Data**.

- **SEIS16Rv2 hangs on bytes_avail**. It is hard to predict what the laptop serial buffer length might be. Line 43 of program **SEIS16Rv2.cpp** specifies a BLOCKLEN value that, if set too large, will assume a buffer longer than the one that actually exists. Setting the BLOCKLEN is a trial and error project. If the program does not hang, recommend you leave it alone.

- **The SEG-2 Fixed Gain needs changing**. If your Anti-alias filter includes a gain different than the one I built, then look at line 803 of program **SEIS16Rv2**. For example, if your gain is 20 instead of my 19, change the definition of FG to `FG= 20;`

- **Cross-talk between channels**. Right, this will happen. If you don't go to extra effort shielding the signal lines (and maybe even if you do), you will notice some cross-talk picked up on a channel without a signal input. For example, if you choose 2 channel recording, but only input a 100 mv P-P sine wave into channel 0, you will see it also on channel 1. It will be about 1 mv P-P on channel 1. In other words, the cross-talk appears to be down by 40dBv.

- **Data Shifted**. If you put the laptop on a charger and then hook up test equipment (oscilloscope for example), you may short the chassis and signal grounds through the AC house outlet ground. Unplug the charger.

- **USB serial settings seem wrong**. With the IDE and Arduino plugged in, use the command
```
stty -F /dev/ttyACM0 -a
```
(assuming the IDE shows this device). Use the result to modify lines 095-109 in Status.cpp.

- **Clipping of the Signal**. Unless you are picking first break refraction data, clipping eliminates the option of further processing of digital signals. Clipping will occur in either of two ways:
  - **Analog Clipping**. The signal input to the pre-amplifier (anti-alias filter) is too large. The default is to use the Arduino 5 volt supply for the op-amps. This makes the virtual ground ideally at 2.5 volts, and analog clipping will occur on signals exceeding about 200 mv (remember the gain is almost 10). So a 200 mv input becomes almost 2 volts. If we add this 2 volt peak signal that to the 2.5V virtual signal ground, the ADC sees about a 4.5 volt input, and that is getting close to the supply. As it turns out, there is also a DC offset in the op-amp output. The solution turns out to be shunting the voltage divider lower resistor to bring the virtual ground close to 2.0 volts (see section 6.1).
  - **Digital Clipping**. The Mayhew ADC can be set to either a 5 volt or a 10 V digitization range. The default software setting is to use the 5 volt range which results in more resolution (counts per volt). Programs **SEIS16Rv2.cpp** and **A16.ino** can be modified to the 10 volt range for large signals (see section 6.3).

- **Signal too Weak**. The other extreme opposite of clipping. Either add another pre-amp stage or a source with more energy.

6 Modifications

Some improvements to the original plan became apparent with a bit of testing. For this discussion, the pre-amplifier and anti-alias filter module will be referred to as the pre-amp.

1. **Pre-amp output bias** observed when a test signal was clipped. Shunting the lower resistor in the virtual signal ground addressed that problem, shifting the virtual ground below 2.5 volts for a 5 volt pre-amp supply.

2. **Adding a second alternative DC power supply** for the pre-amp permitted recording with both the 5 volt and 10 volt ranges on the Mayhew ADC. The 5 volt range powers the pre-amp with the Laptop via the USB. The 10 volt range requires an external battery to realize any benefit (caution, max is 16 volt for Op-amps). I picked 9 volts for setting a second bias shunt resistor. A DPDT toggle switch permits switching both pre-amp supply and shunt resistors. Figure 8 shows the modification.

3. **Recording DC signals**. DC signals are not recorded with the original software which removed the mean of the recorded signal. This was addressed by adding a BIAS variable which permits using a measured constant voltage. This involved adding a chassis ground banana plug socket (black) on the panel so that the voltage of the virtual ground (white) can be measured without opening the case.

4. **Switch to UNIPOLAR** option of the Mayhew. Unsigned integers are now transfered to the Laptop where either the subtraction of the mean or the constant voltage **BIAS** are subtracted to produce a bipolar final result. Why? Because all signals are positive above chassis ground. Permits an extra bit of resolution and slightly faster conversions than BIPOLAR mode.

6.1 Pre-amp Output Bias

The pre-amp virtual ground was chosen mid-way between the supply rails. This choice resulted in the output signal being biased slightly toward the higher voltage rail. A test signal was applied to note where analog clipping began. The result produced analog clipping on the peak of a sine wave test signal before clipping on the trough. Figure 7 (A) shows the situation, the clipped sine wave exhibits more clipping on the peaks than the troughs.

Modification. A shunt was applied across the lower 10K resistor of the voltage divider that produced a virtual ground. The virtual signal ground was shifted from 2.5 volts down to about 2.0 volts (actually 1.949 volts). The shunt is a 20 turn 50K potentiometer (set to 16.47 K Ω). This modification affects all 8 channels simultaneously since they share a common virtual signal ground. Figure 7 (B) shows the change on the recorded signal.

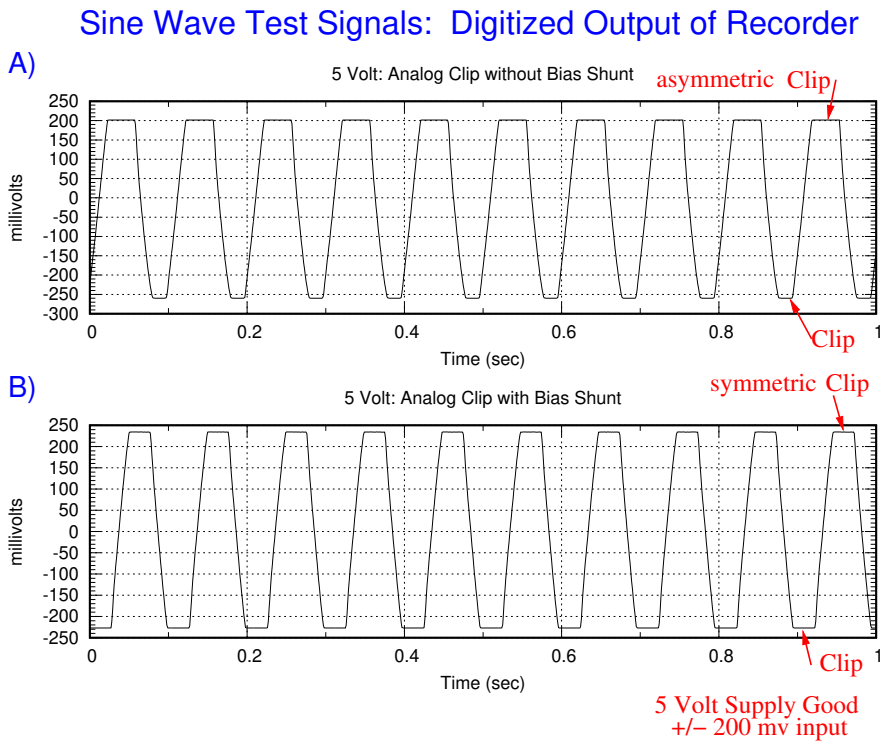


Figure 7: Input exceeds $\pm$ 200 *mv*. A) Sine wave amplitude clips asymmetrically, B) Sine wave clips symmetrically with bias resistor. Mayhew set for 5V range in this instance.

6.2 Adding Second Pre-amp Power Supply

The initial concept was to use the USB wire from the Laptop to power the pre-amp. As long as the input signal's amplitude does not exceed $\pm$200 *mv*, this works well. Recall that the current design is UNIPOLAR, range setting 5 volts on the Mayhew. The modifications include adding a 2.1 mm male panel jack to provide an alternative pre-amp DC power. A double pole double throw (DPDT) toggle switch permits switching between the USB and the external battery. With a 9 volt external battery, we also need a different shunt resistor to increase the virtual ground voltage with respect to the chassis ground. See lower right area of Figure 8. Again, a 20 turn, 50 K potentiometer is used, this time set for 23.35 $K\Omega$. This raises the virtual ground voltage to +3.874 volts. This voltage can be measured between the virtual ground (white banana plugs) and the chassis ground (black banana plug, new).

NOTE: In order to use the 9 volt battery, you should recompile and install the software for the 10 volt range. See section 6.3 on how to change the range from 5 volts to 10 volts on the Mayhew.

6.3 Modification Of Software, 5V to 10V Range

One can change the program codes to permit a larger input, but this will only become effective if an external 9 Volt battery powers the pre-amplifier (rather than the Arduino 5 volt supply via the USB). One can also do this for a 12 volt external battery, but the bias shunt resistor will need to be changed for the higher voltage.

- **Arduino Code: A16.ino**. Comment out lines 30 and 31 with double slashes at the beginning of the lines. Uncomment out lines 32 and 33 (remove the double slashes). This will reset the ADC to the 10 volt range. You then must compile with the `"make"` and `"make upload"` commands (see section 3.2).
- **Laptop Code: SEIS16Rv2.cpp**. Comment out line 41 with double slashes and uncomment in line 42 by removing the double slashes. Recompile (see section 3.2) and install.

With the above changes, you should be able to digitize input signals with a maximum amplitude of $\pm$ 400 *mv* instead of $\pm$ 200 *mv*.

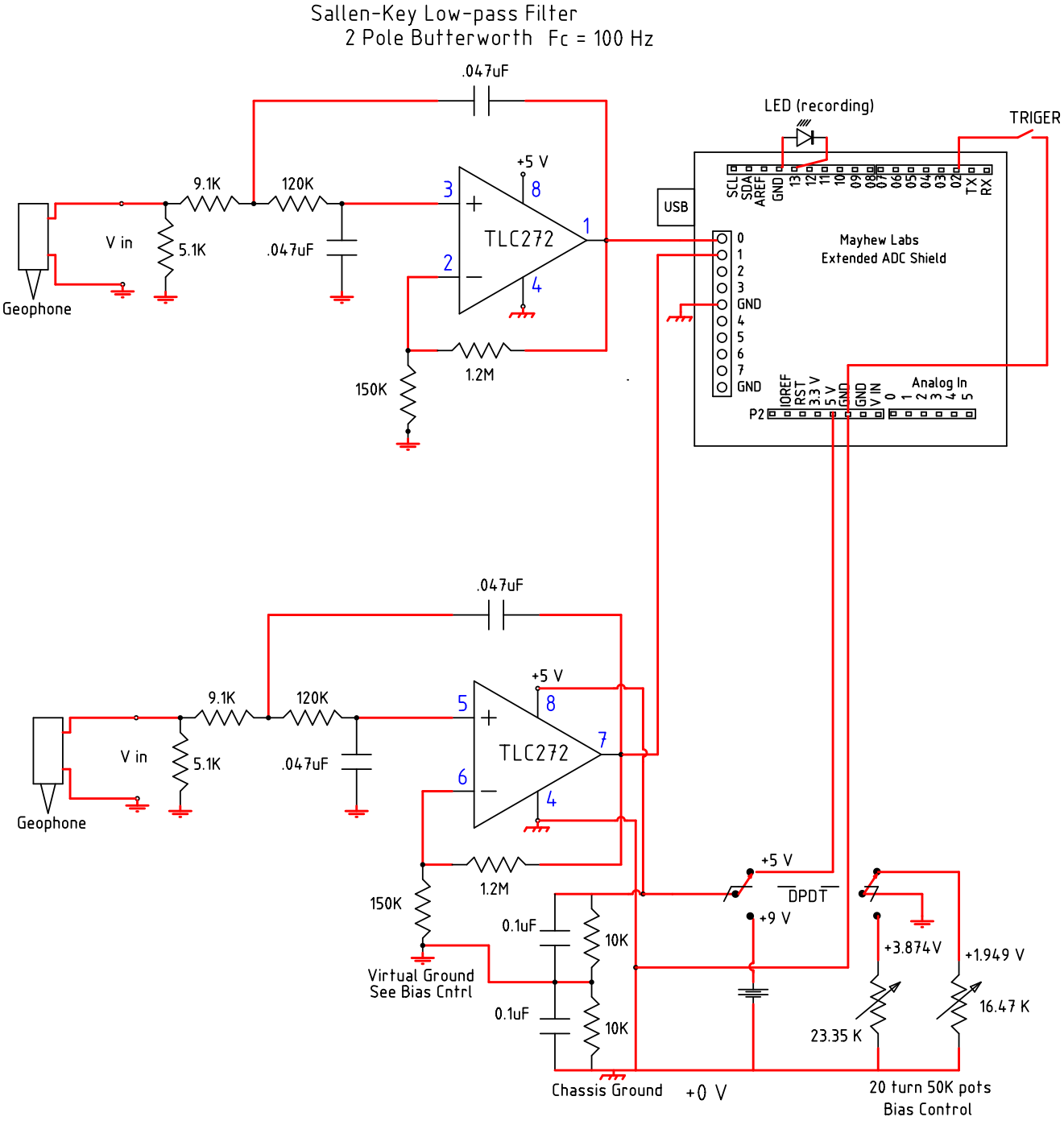


Figure 8: Modifications applied to pre-amp. The shunt on the 10K-10K voltage divider sets the bias. A DPDT toggle switch permits alternative voltages to power the pre-amp and simultaneously switch bias resistors which shunt virtual to chassis ground. To capture DC, edit the **BIAS** parameter in SEIS16v2.cpp to match in microvolts the corresponding bias value (see 6.6.1).

6.4 Test Signals, Unipolar and Range Impact

With the change to UNIPOLAR Mayhew calls and the alternative power supply for the pre-amp, test signals were used to fully understand the choices that are required.

Figure 9 (A) illustrates the 5V range setting employed with a $\pm$ 200mv input signal, bias resistor set at 16.47 $K\Omega$. The virtual ground is at 1.949 volts above chassis ground. The gain for a 10 Hz signal is 19 dbV (factor of 8.9125). Thus the peak input voltage becomes 1.782 + 1.949 = 3.731 volts at the analog to digital converter (ADC). This will not be clipped with a 5V range. The minimum voltage presented to the ADC will be 0.166 volts. This will not be clipped. The software backs out the gain for the plot and storing the data in the SEG-2 file.

Figure 9 (B) illustrates how analog clipping results with a larger amplitude signal ($\pm$ 400mv). The virtual ground voltage is the same as in (A) and the peak voltage after gain presented to the ADC is 3.564 + 1.949 = 5.513 volts. This exceeds the supply rail for the pre-amp filter. It is analog clipping because it happens before the signal reaches the ADC. The evidence for analog clipping will be values that fluctuate in a small ripple about the clipped maximum.

Figure 9 (C) shows digital clipping that occurs with the larger, 9 volt supply voltage for the pre-amp. The virtual ground is now 3.874 volts, and we calculate the voltage at the ADC as 3.564 + 3.874 = 7.438 volts, and this will NOT clip in the pre-amp for which the supply rail is now at 9.000 volts. However, since the ADC range is set to 5V, this will become digitally clipped. If one looks at the values, the evidence for digital clipping is exactly the same values (no fluctuation around the clipped value). CAUTION: If the battery voltage is not regulated, then as the battery becomes discharged, the virtual ground voltage changes, making the above calculations approximate for a fresh battery. There is no clip at the low values since the minimum will be about 3.564 -3.874 = -.31 volts. The plot shows this as just below -300 mv. The other thing to remember is that the mean value is subtracted before plotting and writing to a SEG-2 formatted file. The mean for a clipped signal will be shifted from what it would be for an unclipped signal.

Figure 9 (D) shows how recompiling the codes for a 10V range permits avoiding clipping all together for a $\pm$ 400mv signal.

6.5 Mayhew Floating Point to Integer for USB and SEG-2 Files

The Mayhew provides floating point voltages converted from the 16 bit ADC. The Arduino software, A16.ino, undoes this so that 16 bit unsigned integers can be sent over the USB. The relevant lines are 28-33, 62, 63, 68-70, 76-80, 174-183, 189-194.

The variable **ch0** is returned as a float from the Mayhew ADC. Relevant constants are diff and imax. These are computed:

```
061: //const int imax=pow(2,Nbits-1);  // 2^Nbits-1 since bipolar
062: const unsigned int imax=(unsigned int) pow(2,Nbits);  // 2^Nbits since Unipolar
063: const float diff = RNG/(float) imax;  // volts/count
```

I included line 61 for those who might want to do bipolar conversions, but it is commented out here. The float is converted to integer as follows:

```
179:   ch0=extendedADCShield.analogReadConfigNext(kk, SINGLE_ENDED, UNIPOLAR, Range);
180:   fcounts = ch0/diff;  // fcounts is volts/(volts/count)
181:   value = round(fcounts);
182:   return(value);
. . . .
193:    Serial.write(lowByte(value));
194:    Serial.write(highByte(value));
```

The 16 bit integers are sent to the laptop. Program SEIS16Rv2.cpp then converts these integers to microvolts. Relevant codes in SEIS16Rv2.cpp are:

```
095:     const unsigned int imax = floor(pow((double)2.,(double)(Nbits)));
096:     const float diff = RNG/(float)imax;  //volts/count
097:     float mvolts;
```

```
098:     int    microvolts;
. . . .
247:          read(serialfd,&Lbyte,1);
248:   ivalue=Lbyte;
249:   read(serialfd,&Hbyte,1);
250:   ivalue=Hbyte*256+ivalue;
251:          mvolts = ((float)(ivalue) * diff)*1000.0;
252:   microvolts = floor( (mvolts * 1000.0 +.5)) ;
```

The integer value microvolts is what will be written to the SEG-2 file. If there is a vertical stack, microvolts are accumulated in RAM variable s1[]. The array s1[] is then written to the output file after removing either the mean value or a constant.

```
260:  s1[sendcount] = microvolts + s1[sendcount];
. . . .
349:  WriteSEG2(nsamp,nchannels,fsamin,s1,ybase,fname,nstack,source,phones,nSP,
350:  sx,rx);
```

The SEG-2 headers include a value that converts the integer to millivolts. This computation uses the fixed gain of the preamp, FG.

```
845:  sprintf(STRING,"  SAMPLE_INTERVAL %.6f  ",fsamin);
846:  /* Arduino A/D -->SEIS16v2 s1[] is in microvolts, SEG-2 Standard
847:  requires a descaling factor applied to the SEG-2 data which will
848:  yield mvolts */
849:  writestring(STRING,RES,LTC,h1);
850:
851:     DSF = 1.000 / (1000. * pow((double) 10., (double) ((float) FG)/20.0));
852:  sprintf(STRING,"  DESCALING_FACTOR %14.7E  ",DSF);
853:  writestring(STRING,RES,LTC,h1);
```

The program you use to read SEG-2 files will matter, depending on the target file format. My egg2seg program converts to my bsegy processing format. The code, egg2seg, recognizes what factor would result in millivolts, but chooses to write files in microvolts since that retains the precision in converting to floats (bsegy format). For more on this topic, see my Basic Seismic Utilities (BSU) processing package available on my documentation and download pages.

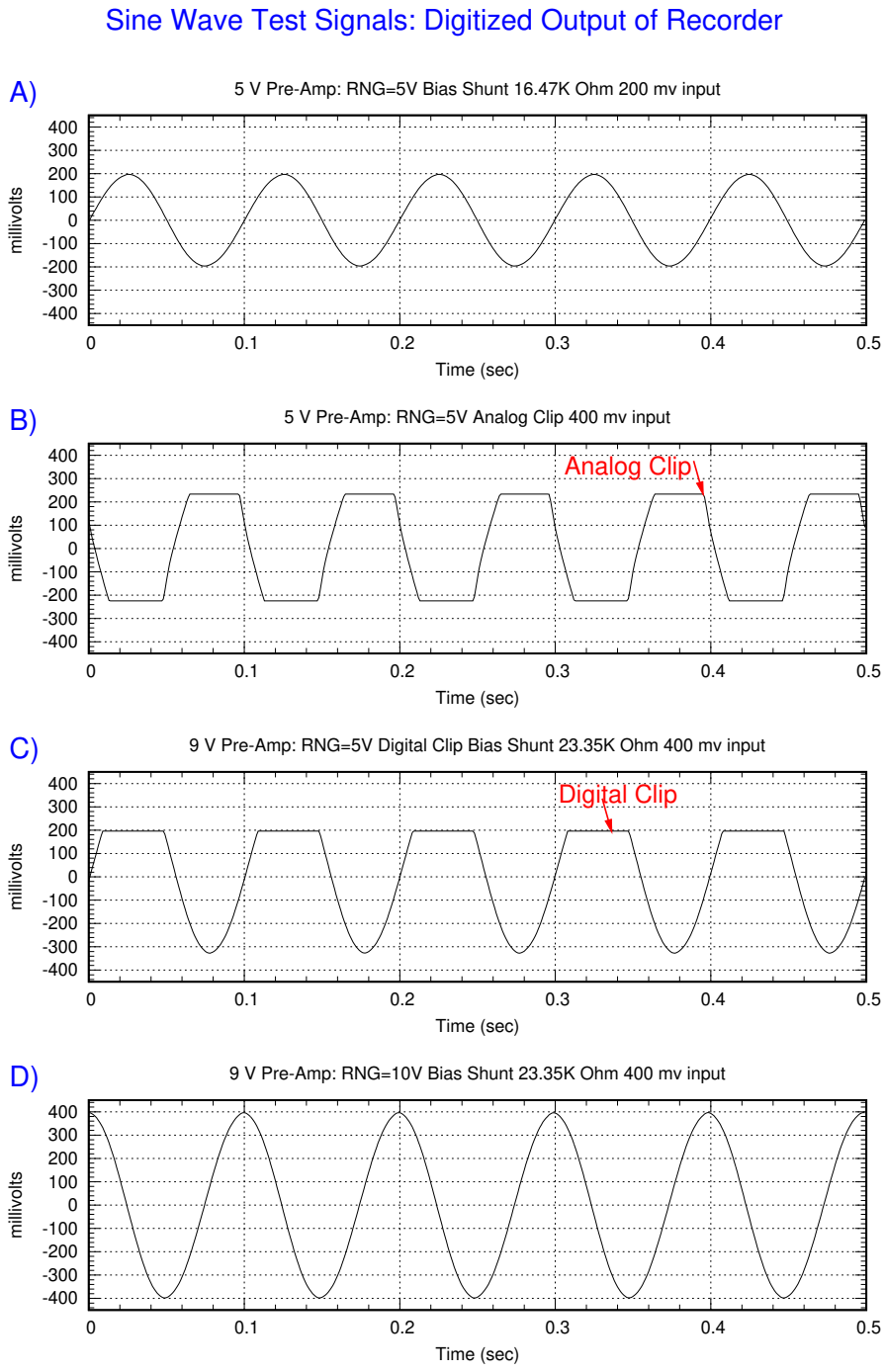


Figure 9: Case DC not recorded, mean removed. A) Recording a 10 Hz $\pm$ 200*mv* signal using 5 volt pre-amp supply, software range RNG=5V. B) Recording, increasing amplitude of input to $\pm$ 400*mv* illustrates analog clipping. C) Switching to 9 volt alternative battery for pre-amp but RNG=5V, clipping now digital. This requires recompiling software for range RNG=10V. D) Neither analog or digital clipping with the larger software range (RNG=10V) and alternative 9 volt battery.

6.6 Recording DC (0 Hz Frequency) Signals

While geophones and sources do not produce DC signals, recording with a short temporal aperture of a low frequency signal may introduce a recording that has a DC bias. The default setting in the software removes the mean before displaying or saving recordings to a file. See functions **meanval()** and **saveplot()** in the laptop program **SEIS16v2.cpp** for the relevant code. For example, consider recording a low frequency sine wave with an aperture time of half a cycle. Subtracting the mean value will shift the result. Or, maybe you want to use the recorder for another application where DC content is important. Here is how that is done.

6.6.1 Modifications

First, I have edited the original **meanval()** function to permit an alternative return value of a constant. This constant is the voltage in microvolts of the difference between the virtual signal ground and the chassis ground. The variable name for this value is **BIAS**. There are also changes to **saveplot()**. **IMPORTANT: This value will vary with the supply voltage of the pre-amp**. Consider the alternatives of the laptop 5 volts vs. a 9 volt external battery. To make the determination of this value easy, I added an additional banana plug socket (black) to the panel that brings the chassis ground to the top of the panel. A digital volt meter is then used to measure the voltage between the chassis ground (black) and the virtual signal ground (white). With my electronics, the values measured in microvolts were:

- **5 Volt Laptop Pre-amp Supply**: BIAS = 1949000 microvolts (1.949 volts)
- **9 Volt Battery Pre-amp Supply**: BIAS = 3874000 microvolts (3.874 volts)
- **Original Default (remove DC)**: BIAS = 0 (this is a switch to remove the mean and not record DC.

These values depend on the shunt applied to the voltage divider to center the output signal between the supply rails (see section 6.4 and Figure 9). NOTE: The default setting in the code is to remove DC. So to record DC, you need to edit the BIAS value for your intended pre-amp supply voltage, recompile, and then make install. See lines 112 to 114 in the appendix, **SEIS16v2.cpp**. The modified panel is shown in Figure 10.

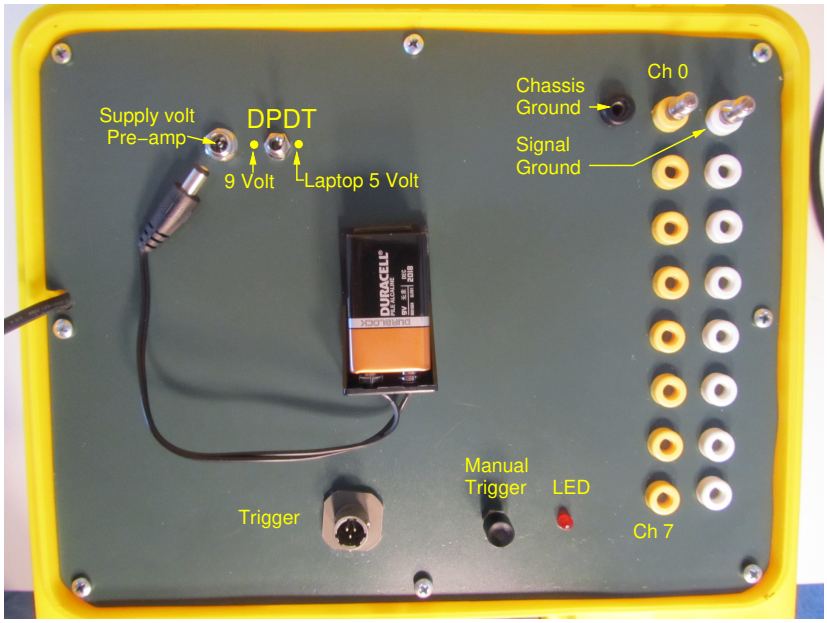


Figure 10: Modified Panel. DC BIAS is measured between chassis and signal grounds (black and white banana plug sockets). Convert voltage to microvolts for **BIAS** constant in SEIS16Rv2.cpp to record DC signals. Recommend voltage regulation for external power source stability (particularly if recording DC is an objective).

References

Janert, P. K. 2010. *Gnuplot in Action, Understanding data with graphs*. Manning.

Michaels, P. 2018 (Apr.). *Basic Seismic Utilities*. https://doi.org/10.18122/geo_data/3/boisestate.

Ousterhout, J. K. 1994. *Tcl and the TK Toolkit*. Addison-Wesley Professional Computing Series.

Pullan, S. E. 1990. Recommended standard for seismic (/radar) data files in the personal computer environment. *Geophysics*, **55**(9), 1260–1271.

Tedeschi, F. P. 1979. *The Active Filter Handbook*. Tab Books.

Various. 2019. *GCC, the GNU Compiler Collection*. https://gcc.gnu.org/.

Index

7 Software Appendix

One should download the software archive, **SeisR-1.0.0.tgz** since it includes the libraries not listed here. **The line numbers (start of a line, xxx:) are given for reference, should not be included if you cut and paste from these listings.**

7.1 Arduino, A16.ino

7.1.1 Makefile

```
001: # Arduino Make file. Refer to https://github.com/sudar/Arduino-Makefile
002: # Some commands:
003: # type make
004: # type make upload
005: # type make clean
006:
007: BOARD_TAG    = uno
008: OPTIMIZATION_LEVEL = 3
009:
010: #include ../../Arduino.mk
011: include /usr/share/arduino/Arduino.mk
```

7.1.2 A16.ino

```
001: //   $Id: A16.ino,v 1.10 2019/04/02 23:00:47 pm Exp $
002: /* |-----------------------------------------------------------|
003:    |A16.ino     Seismic Recorder Arduino program               |
004:    |Author: Dr. P. Michaels, PE <pmsolid@cableone.net>         |
005:    |or <paulmichaels@boisestate.edu>                           |
006:    |  NOTICE:                                                  |
007:    |  Copyright (C) 2019 Paul Michaels                         |
008:    |  This program is free software; you can                   |
009:    |  redistribute it and/or modify it under the terms         |
010:    |  of the GNU General Public License as published           |
011:    |  by the Free Software Foundation; either version          |
012:    |  3 of the License, or (at your option) any later          |
013:    |  version.  This program is distributed in the             |
014:    |  hope that it will be useful, but                         |
015:    |  WITHOUT ANY WARRANTY; without even the implied           |
016:    |  warranty of MERCHANTABILITY or FITNESS FOR A             |
017:    |  PARTICULAR PURPOSE.  See the GNU General Public          |
018:    |  License for more details.                               |
019:    |  You should have received a copy of the GNU               |
020:    |  General Public License along with this program;          |
021:    |  If not, see <https://www.gnu.org/licenses/>              |
022:    |-----------------------------------------------------------|
023:  */
024: // UNIPOLAR VERSION
025: #include <ExtendedADCShield.h>
026: #include <SPI.h>
027:
028: #define nchar 20
029: #define Nbits 16
030: #define Range RANGE5V   /*Set RNG to match Range choice */
031: #define RNG 5.  // +/- volts
032: //#define Range RANGE10V   /*Set RNG to match Range choice */
033: //#define RNG 10.  // +/- volts
034:
```

```
035: //       optimization "s", IDE
036: //const float slope = 479.0;   // overhead in microseconds
037: //const float intercept = 393.5;  // slope*k + intercept
038: //       desktop dell
039: //       optimization "3", make file
040: //const float slope = 453.2857; // overhead in microseconds
041: //const float intercept = 376.7142; // slope*k + intercept
042: //       opt = 3 blue dell laptop
043: const float slope = 448.7142;  // overhead in microseconds
044: const float intercept = 374.2857; // slope*k + intercept
045: unsigned int shft =0;
046: int i,j,k;                    // k= number channels
047: int trig = LOW;
048: int nsamp;
049: unsigned int ndt;  //sample interval in microseconds
050: char buf[nchar];
051: char nsampbuf[nchar];
052: char ndtbuf[nchar];
053: char nchbuf[nchar];
054: //String S ;          /*dummy string to clear serial for next trigger  */
055: char ch;
056:
057:
058:
059: const int ledPin = 13;
060: const int inputPin = 2;
061: //const int imax=pow(2,Nbits-1);  // 2^Nbits-1 since bipolar
062: const unsigned int imax=(unsigned int) pow(2,Nbits);  // 2^Nbits since Unipolar
063: const float diff = RNG/(float) imax;  // volts/count
064:
065: //Initialize the ADC Shield with default pins and 16-bit ADC (LTC1858)
066: ExtendedADCShield extendedADCShield(Nbits);
067:
068: float ch0;
069: unsigned int value;
070: float fcounts;
071: int armed = 1;
072: const long DELAY = 1000;
073: int fieldx;
074: int parm[3];
075:
076: // Prototype functions----------------------------------------------------
077: unsigned int getvalue(int m);
078: int SampleSendValues(int m);
079: void sendBinary(unsigned int value);
080: void sendBinary(long value);
081:
082: // SETUP-----------------------------------------------------------------
083: void setup() {
084:   Serial.begin(115200, SERIAL_8N1);
085:   pinMode(ledPin, OUTPUT);
086:   pinMode(inputPin, INPUT);
087:   digitalWrite(inputPin,HIGH);  /*turn on pull up resistor for trigger pin 2
088:                                 trigger by shorting pin 2 to ground  */
089:   digitalWrite(ledPin,LOW);
090:   //SPI.begin must be called here on Due only
091:   SPI.begin();
092:   //Throw out first read (junk data) and configure input 0 as single ended bipolar -5 to +5V
093:   extendedADCShield.analogReadConfigNext(0, SINGLE_ENDED, UNIPOLAR, Range);
```

```
094:  // while(ch=Serial.read() >= 0) ; //flush the receive buffer
095:   Serial.setTimeout(DELAY);
096:
097: } //end setup
098:
099: // LOOP----------------------------------------------------------------------
100: void loop() {
101:
102:       if ( Serial.available() >= (nchar - 1) ) {
103:  for (fieldx =0; fieldx < 3; fieldx++)
104:  {
105:  parm[fieldx] = Serial.parseInt(); //get a parameter
106:  } //next fieldx
107:
108:  nsamp = parm[0];
109:  ndt = parm[1];
110:  k = parm[2];
111: /*          S= Serial.readString();
112:           S.toCharArray(buf,nchar);
113:           for (i=0; i<5; i++) nsampbuf[i]=buf[i];      // 99999 99999 99999
114:           for (i=6; i<11; i++) ndtbuf[i-6]=buf[i];    //  5 digits with space between
115:           for (i=12; i<17; i++) nchbuf[i-12]=buf[i];
116:           nsamp = atoi(nsampbuf);
117:           ndt = atoi(ndtbuf);
118:           k = atoi(nchbuf);
119: */
120:             shft = ceil((slope*(float) k ) + intercept); //shft = overhead
121:             if (ndt > shft) {
122:             ndt = ndt - shft;  //subtract extra code overhead
123:             } //endif extra
124:
125:             Serial.print('H');
126:
127:
128:   /*
129:           Serial.println(nsamp);
130:           Serial.println(ndt);
131:           Serial.println(k);
132:           Serial.println(shft);
133:  */
134:
135:           } //end if serial available
136:
137:
138:
139:           i = 0;
140:           if (armed == 1) {
141:           trig = digitalRead(inputPin);
142:           }
143:            if (trig == HIGH)  {
144:              digitalWrite(ledPin,LOW);
145:            } else {
146:              digitalWrite(ledPin,HIGH);
147:      //       Serial.flush();
148:
149:             long int starttime=millis();
150:
151:             armed = SampleSendValues(k);
152:
```

```
153:             long duration = millis() - starttime;
154:
155:
156:   //          Serial.print('q');              /* send done signal */
157:
158:             sendBinary(duration);  /* send long value  */
159:      Serial.flush();
160:    //        Serial.println(" ");
161:    //        Serial.print(duration);
162:    //        Serial.flush();
163:    //        S = Serial.readString();
164:         /* clear serial line read into dummy string
165:                                          and turn off TX, otherwise next trigger
166:                                          will pick up garbage left in serial line*/
167:
168:          } //end Trigger loop
169:          armed = 1;
170:          trig = HIGH;
171:
172: }  //endloop
173:
174: // GETVALUE FUNCTION-----------------------------------------------------------------------
175: unsigned int getvalue(int kk)
176: {
177:   //Read input 0, set up input 0 as bipolar -RNG to RNG Volts (single ended neg goes to ground)
178:   // bipolar relative to board ground +/-RNG volts
179:   ch0 =  extendedADCShield.analogReadConfigNext(kk, SINGLE_ENDED, UNIPOLAR, Range);
180:   fcounts = ch0/diff;  // fcounts is volts/(volts/count)
181:   value = round(fcounts);
182:   return(value);
183: }  //end getvalue
184:
185: // SAMPLE SEND VALUES FUNCTION-------------------------------------------------------------
186: int SampleSendValues(int k)
187: {
188:   int kk;
189:   for (j=0; j<nsamp; j++) {
190:      extendedADCShield.analogReadConfigNext(0, SINGLE_ENDED, UNIPOLAR, Range);
191:     for (kk = 0; kk<k; kk++) {
192:   value = getvalue(kk+1);
193:    Serial.write(lowByte(value));
194:    Serial.write(highByte(value));
195: // Serial.print(lowByte(value));
196: // Serial.print(" ");
197: // Serial.print(highByte(value));
198: // Serial.print(" ");
199: // Serial.println(value);
200:     }
201: //Serial.println(value);
202:   if (ndt <= 10000)  {
203:   delayMicroseconds(ndt);                   /* delay ndt microsec.  */
204:   } else {
205:   delay(ndt/1000);                               /*delay msec */
206:   } //endif
207:   } //next j
208:   return(0);
209: }
210:
211: // SEND BINARY FUNCTIONS-------------------------------------------------------------------
```

```
212: void sendBinary(int value)
213: {
214:  Serial.write(lowByte(value)); //send low byte
215:  Serial.write(highByte(value)); //send high byte
216: }
217: //-------------------------------------------------------------------------------------------
218: void sendBinary(long value)
219: {
220:   //first send the low 16 bit integer value
221:   int temp = value & 0xFFFF;  //get value of lower 16 bits
222:   sendBinary(temp);
223:   // then send the higher 16 bit integer value:
224:   temp = value >>16; //get higher 16 bits
225:   sendBinary(temp);
226: }
```

7.2 Laptop Software

The line numbers (start of a line, xxx:) are given for reference, should not be included if you cut and paste from these listings.

7.2.1 Makefile

```
001: # Makefile for C++ Arduino serial USB com and Arduino control
002: # $Id: Makefile,v 1.4 2019/02/11 19:52:48 pm Exp $
003: # Paul Michaels <pmsolid@cableone.net> OR <paulmichaels@boisestate.edu>
004: PREFIX = /usr/local
005: EXEC=  Status SEIS16v2
006:
007: all: $(EXEC)
008:
009: SEIS16v2: SEIS16v2.cpp
010:  g++ -g -Wall SEIS16v2.cpp -o SEIS16v2
011:
012: Status: Status.cpp
013:  g++ -g -Wall Status.cpp -o Status
014:
015: install: ${EXEC}
016:  install Status ${PREFIX}/bin
017:  install SEIS16v2 ${PREFIX}/bin
018:  install seisR.tk ${PREFIX}/bin
019:
020: clean:
021:  rm -f Status
022:  rm -f SEIS16v2
023:
024: uninstall:
025:  rm -f ${PREFIX}/bin/Status
026:  rm -f ${PREFIX}/bin/SEIS16v2
027:  rm -f ${PREFIX}/bin/seisR.tk
```

7.2.2 seisR.tk (GUI)

This file will be installed when Makefile is run with the command, `make install`.

```
001: #!/usr/bin/wish
002: # $Id: seisR.tk,v 1.7 2019/02/18 14:52:21 pm Exp $
003: #   |-----------------------------------------------------------|
004: #   |seisR.tk    Seismic Recorder Requires TCL/TK installed     |
005: #   |Author: Dr. P. Michaels, PE <pmsolid@cableone.net>         |
006: #   |or <paulmichaels@boisestate.edu>                          |
007: #   |-----------------------------------------------------------|
008:
009: set MSG1 "  NOTICE:
010:   Copyright (C) 2019 Paul Michaels
011:   <pmsolid@cableone.net>
012:   This program is free software; you can
013:   redistribute it and/or modify it under the terms
014:   of the GNU General Public License as published
015:   by the Free Software Foundation; either version
016:   3 of the License, or (at your option) any later
017:   version.  This program is distributed in the
018:   hope that it will be useful, but
019:   WITHOUT ANY WARRANTY; without even the implied
020:   warranty of MERCHANTABILITY or FITNESS FOR A
021:   PARTICULAR PURPOSE.  See the GNU General Public
022:   License for more details.
023:   You should have received a copy of the GNU
024:   General Public License along with this program.
025:   If not, see <https://www.gnu.org/licenses/> "
026:
027: wm maxsize . 1000 800
028: wm minsize . 300 300
029: wm title . "Seismic Recorder"
030:
031: #set bgc #ffffddddbbbb
032: #set bgc2 #bbbbdddd0000
033: set bgc gray90
034: set bgc2 gray90
035: set armc red
036: set ltred #ffffbbbbbbbb
037:
038: #set fNumber
039: if { [info exists fNumber] }  {incr fNumber 0001} else {set fNumber 0000}
040: #set  spN
041: if { [info exists spN] }  {incr spN 0} else {set spN 0000}
042: #set  sx
043: if { [info exists sx] }  {incr sx 0} else {set sx 0}
044: #set r$ix
045: for {set i 0} {$i < 8 } {incr i +1} {
046: if { [info exists rx$i ]} {incr rx$i 0} else {set rx$i [expr $i+1]}
047: }
048:
049: # MASTER RECORDING FRAME
050: grid [frame .m -borderwidth 4 -relief ridge -padx 1  -bg $bgc ] -row 0 -column 0
051: # MASTER PLOT FRAME
052: grid [frame .p -borderwidth 4 -relief ridge -padx 1  -bg $bgc ] -row 1 -column 0
053:
054: # NUMBER SAMPLES
055: grid [frame .m.nsamp -borderwidth 4 -relief sunken -padx 1  -bg $bgc ] -row 0 -column 0
056: grid [label .m.nsamp.label1 -text "Number of samples" -bg $bgc ] -row 0 -column 1
```

```
057: grid [radiobutton .m.nsamp.b1 -text " 500" -variable nsampl -value "500" -bg $bgc ] \
058:  -row 1 -column 1
059: grid [radiobutton .m.nsamp.b2 -text "1000" -variable nsampl -value "1000" -bg $bgc ] \
060:  -row 2 -column 1
061: grid [radiobutton .m.nsamp.b3 -text "2000" -variable nsampl -value "2000" -bg $bgc ] \
062:  -row 3 -column 1
063: grid [radiobutton .m.nsamp.b4 -text "3000" -variable nsampl -value "3000" -bg $bgc ] \
064:  -row 4 -column 1
065: grid [radiobutton .m.nsamp.b5 -text "4000" -variable nsampl -value "4000" -bg $bgc ] \
066:  -row 5 -column 1
067: grid [radiobutton .m.nsamp.b6 -text "5000" -variable nsampl -value "5000" -bg $bgc ] \
068:  -row 6 -column 1
069: .m.nsamp.b1 select
070:
071: # NUMBER OF CHANNELS
072: grid [frame .m.nch -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 0 -column 2
073: grid [label .m.nch.label1 -text "No. Channels" -bg $bgc2 ] -row 1 -column 0
074: grid [radiobutton .m.nch.b1 -text "1 " -variable nch -value "1" -bg $bgc2 \
075: -command ".m.ndt.b1 select"  ] -row 2 -column 0
076: grid [radiobutton .m.nch.b2 -text "2 " -variable nch -value "2" -bg $bgc2 \
077: -command ".m.ndt.b2 select" ] -row 3 -column 0
078: grid [radiobutton .m.nch.b3 -text "3 " -variable nch -value "3" -bg $bgc2 \
079: -command ".m.ndt.b2 select"  ] -row 4 -column 0
080: grid [radiobutton .m.nch.b4 -text "4 " -variable nch -value "4" -bg $bgc2 \
081: -command ".m.ndt.b3 select"  ] -row 5 -column 0
082: grid [radiobutton .m.nch.b5 -text "5 " -variable nch -value "5" -bg $bgc2 \
083: -command ".m.ndt.b3 select"  ] -row 2 -column 1
084: grid [radiobutton .m.nch.b6 -text "6 " -variable nch -value "6" -bg $bgc2 \
085: -command ".m.ndt.b4 select"  ] -row 3 -column 1
086: grid [radiobutton .m.nch.b7 -text "7 " -variable nch -value "7" -bg $bgc2 \
087: -command ".m.ndt.b4 select"  ] -row 4 -column 1
088: grid [radiobutton .m.nch.b8 -text "8 " -variable nch -value "8" -bg $bgc2 \
089: -command ".m.ndt.b5 select"  ] -row 5 -column 1
090: .m.nch.b1 select
091:
092: # SAMPLE INTERVAL
093: grid [frame .m.ndt -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 0 -column 1
094: grid [label .m.ndt.label1 -text "Sample Interval" -bg $bgc2 ] -row 0 -column 1
095: grid [radiobutton .m.ndt.b1 -text "1 ms" -variable ndt -value "1000" -bg $bgc2 ] -row 1 -column 1
096: grid [radiobutton .m.ndt.b2 -text "2 ms" -variable ndt -value "2000" -bg $bgc2 ] -row 2 -column 1
097: grid [radiobutton .m.ndt.b3 -text "3 ms" -variable ndt -value "3000" -bg $bgc2 ] -row 3 -column 1
098: grid [radiobutton .m.ndt.b4 -text "4 ms" -variable ndt -value "4000" -bg $bgc2 ] -row 4 -column 1
099: grid [radiobutton .m.ndt.b5 -text "5 ms" -variable ndt -value "5000" -bg $bgc2 ] -row 5 -column 1
100: .m.ndt.b1 select
101:
102: # VERTICAL STACK
103: grid [frame .m.stk -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 0 -column 3
104: grid [label .m.stk.label1 -text "Vertical Stack" -bg $bgc2 ] -row 1 -column 1
105: grid [radiobutton .m.stk.b1 -text " 1" -variable stk -value "1" -bg $bgc2 ] -row 2 -column 1
106: grid [radiobutton .m.stk.b2 -text " 2" -variable stk -value "2" -bg $bgc2 ] -row 3 -column 1
107: grid [radiobutton .m.stk.b3 -text " 5" -variable stk -value "5" -bg $bgc2 ] -row 4 -column 1
108: grid [radiobutton .m.stk.b4 -text "10 " -variable stk -value "10" -bg $bgc2 ] -row 5 -column 1
109: .m.stk.b1 select
110:
111: # SOURCE HEADER
112: grid [frame .m.src -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 0 -column 4
113: grid [label .m.src.label1 -text "Source Header" -bg $bgc2 ] -row 1 -column 1
114: grid [radiobutton .m.src.b1 -text "Hammer      " -variable src -value "H" -bg $bgc2 ] \
115:  -row 2 -column 1
```

```
116: grid [radiobutton .m.src.b2 -text "Weight Drop" -variable src -value "W" -bg $bgc2 ] \
117:  -row 3 -column 1
118: grid [radiobutton .m.src.b3 -text "Dynamite     " -variable src -value "D" -bg $bgc2 ] \
119:  -row 4 -column 1
120: grid [radiobutton .m.src.b4 -text "Gun            " -variable src -value "G" -bg $bgc2 ] \
121:  -row 5 -column 1
122: grid [radiobutton .m.src.b5 -text "Vibrator       " -variable src -value "V" -bg $bgc2 ] \
123:  -row 6 -column 1
124: .m.src.b1 select
125:
126: # SOURCE NAME
127: grid [frame .m.sloc -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 0 -column 5
128: grid [label .m.sloc.label1 -text "SP Number"  -bg $bgc2 ] -row 1 -column 1
129: grid [entry .m.sloc.myEntry -relief ridge -bg $bgc2 \
130: -width 4 -textvariable spN -justify left ] -row 3 -column 1
131: grid [button .m.sloc.b1 -text "Zero" -command {set spN 0}  -bg $bgc2 -relief raised] \
132:  -row 4 -column 1
133:
134: # STATUS FRAME
135: grid [frame .m.stat -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 1 -column 0
136: grid [label .m.stat.label1 -text "USB Connection Status" -bg $bgc2 ] -row 2 -column 0
137: grid [button .m.stat.b1 -text "Check" \
138:  -command {set msg1 [exec Status]; MSG $msg1 } -bg $bgc -relief raised] -row 3 -column 0
139:
140: # DATA FILE NAME FRAME
141: grid [frame .m.data -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 1 -column 1
142: grid [label .m.data.label1 -text "Data File Number"  -bg $bgc2 ] -row 1 -column 1
143: grid [entry .m.data.myEntry -relief ridge -bg $bgc2 \
144: -width 4 -textvariable fNumber -justify left ] -row 3 -column 1
145: grid [button .m.data.b1 -text "Zero" -command {set fNumber 0000} -bg $bgc2 -relief raised]\
146:  -row 4 -column 1
147:
148: # ARM BUTTON
149: grid [frame .m.arm -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 1 -column 2
150: grid [label .m.arm.label1 -text "         Arm Recorder" -bg $bgc2 ] -row 2 -column 2
151:
152: grid [button .m.arm.b1 -text "ARM" \
153:  -command {catch {clearplots}; set msg2  \
154: [ exec SEIS16v2 $nsampl $ndt $nch $stk $fNumber \
155:  $src $r0$r1$r2$r3$r4$r5$r6$r7 $spN $sx \
156:  $rx0 $rx1 $rx2 $rx3 $rx4 $rx5 $rx6 $rx7 ];  \
157:   ARMCMD $msg2 ; incr fNumber 1 ; goplot "plot$pchnl.png" $bgc2 }  \
158:  -activebackground red -bg $ltred -relief raised] -row 3 -column 2
159: grid [button .m.arm.b2 -text "Clear" \
160:  -command {set msg2 " "; ARMCMD $msg2}] -row 3 -column 3
161:
162: #QUIT BUTTON
163: grid [frame .m.exit -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 1 -column 3
164: grid [button .m.exit.b1 -text "Quit" -command exit -bg $ltred -activebackground lightgreen \
165:  -relief raised ] -row 1 -column 0
166:
167: # GEOPHONE HEADER
168: grid [frame .m.rec -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 1 -column 4
169: grid [label .m.rec.label1 -text "Phones" -bg $bgc2 ] -row 0 -column 1
170: for { set i 0; set m 0} { $i < 8} {incr i +1; incr m +3} {
171: set ROWH [expr 1+$i]
172: set j [expr 1+$m]
173: set k [expr 2+$m]
174: grid [label .m.rec.bb$i -text "Ch$i" -bg $bgc2] -row $ROWH -column 0
```

```
175: grid [radiobutton .m.rec.b$m -text "V" -variable r$i -value "V" -bg $bgc2 ] \
176:  -row $ROWH  -column 1
177: grid [radiobutton .m.rec.b$j -text "R" -variable r$i -value "R" -bg $bgc2 ] \
178:  -row $ROWH  -column 2
179: grid [radiobutton .m.rec.b$k -text "T" -variable r$i -value "T" -bg $bgc2 ] \
180:  -row $ROWH  -column 3
181: }
182: .m.rec.b0 select
183: .m.rec.b4 select
184: .m.rec.b8 select
185: .m.rec.b9 select
186: .m.rec.b13 select
187: .m.rec.b17 select
188: .m.rec.b18 select
189: .m.rec.b21 select
190:
191: # GEOMETRY
192: grid [frame .m.geom -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 1 -column 5
193: grid [label .m.geom.label1 -text "Geometry"  -bg $bgc2 ] -row 1 -column 0
194: grid [label .m.geom.sX -text "S-xcord" -bg $bgc2] -row 2 -column 0
195: grid [entry .m.geom.myEntryS -relief ridge -bg $bgc2 \
196: -width 4 -textvariable sx -justify left ] -row 2 -column 1
197: # Receivers
198: for { set i 0} { $i < 8} {incr i +1} {
199: set ROW [expr 3+$i]
200: grid [label .m.geom.rX$i -text "R$i-xcord" -bg $bgc2] -row $ROW -column 0
201: grid [entry .m.geom.myEntryR$i -relief ridge -bg $bgc2 \
202: -width 4 -textvariable rx$i -justify left ] -row $ROW -column 1
203: }
204:
205: # COPYRIGHT MESSAGE BUTTON
206: grid [frame .m.about -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 2 -column 0
207: grid [button .m.about.b1 -text "About" -command {GPL $MSG1} -bg lightgrey \
208:  -activebackground lightgreen -relief raised ] -row 0 -column 0
209:
210:
211: # SELECT CHANNEL TO PLOT
212: grid [frame .p.chnl -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 0 -column 0
213: grid [label .p.chnl.label1 -text "Plot Channel" -bg $bgc2 ] -row 0 -column 0
214: grid [radiobutton .p.chnl.b1 -text "1" -variable pchnl -value "0" -bg $bgc2 \
215:   ] -row 1 -column 1
216: grid [radiobutton .p.chnl.b2 -text "2" -variable pchnl -value "1" -bg $bgc2 \
217: -command ".m.nch.b2 select"    ] -row 1 -column 2
218: grid [radiobutton .p.chnl.b3 -text "3" -variable pchnl -value "2" -bg $bgc2 \
219: -command ".m.nch.b3 select"  ] -row 1 -column 3
220: grid [radiobutton .p.chnl.b4 -text "4" -variable pchnl -value "3" -bg $bgc2 \
221: -command ".m.nch.b4 select"  ] -row 1 -column 4
222: grid [radiobutton .p.chnl.b5 -text "5" -variable pchnl -value "4" -bg $bgc2 \
223: -command ".m.nch.b5 select"  ] -row 1 -column 5
224: grid [radiobutton .p.chnl.b6 -text "6" -variable pchnl -value "5" -bg $bgc2 \
225: -command ".m.nch.b6 select"  ] -row 1 -column 6
226: grid [radiobutton .p.chnl.b7 -text "7" -variable pchnl -value "6" -bg $bgc2 \
227: -command ".m.nch.b7 select"  ] -row 1 -column 7
228: grid [radiobutton .p.chnl.b8 -text "8" -variable pchnl -value "7" -bg $bgc2 \
229: -command ".m.nch.b8 select"  ] -row 1 -column 8
230: .p.chnl.b1 select
231: grid [button .p.chnl.b11 -text "Erase Plots" -command  {catch {clearplots}} \
232:  -bg $ltred -activebackground red  -relief raised] -row 1 -column 10
233: grid [button .p.chnl.b21 -text "Replot" -command  {catch {goplot "plot$pchnl.png" $bgc2}} \
```

```
234:  -bg yellow -relief raised] -row 1 -column 9
235:
236: # FORM ARM COMMAND FOR SEIS16
237: proc ARMCMD msg2 {
238: destroy .m.arm.value
239: label .m.arm.value -text "$msg2"
240: grid .m.arm.value -row 4 -column 2
241: }
242:
243: # CHECK STATUS CONNECTION
244: proc MSG msg1 {
245: destroy .m.stat.value
246: if { $msg1 == "Not Connected" } {set scolor red} else {set scolor lightgreen}
247: grid [label .m.stat.value -text "$msg1" -bg $scolor  ] -row 4 -column 0
248: }
249:
250: # DISPLAY RECORDING IMAGE
251: proc goplot {filename bgc2} {
252: destroy .p.img
253: grid [frame .p.img  -borderwidth 4 -relief sunken -padx 2 -bg $bgc2 ] -row 2 -column 0
254: image create photo imgobj -file $filename -width 640 -height 240
255: grid [label .p.img.imgobj ] -row 1 -column 0
256: .p.img.imgobj configure -image imgobj
257: }
258:
259: proc clearplots {} {
260: eval exec rm [glob *.png]
261: }
262:
263: proc GPL {MSG1} {
264: toplevel .gpl
265: grid [frame .gpl.about -borderwidth 4 -relief sunken -padx 2 -bg white ] -row 0 -column 0
266: grid [message .gpl.about.m1 -width 20c -justify left -bg white -text $MSG1 ] -row 0 -column 0
267: grid [button .gpl.about.b1 -text "OK" -command {destroy .gpl} -bg lightgrey \
268:  -activebackground lightgreen -relief raised ] -row 1 -column 0
269:
270: }
```

7.2.3 Status.cpp

To capture the serial settings with the IDE and Arduino plugged in, use the command:
`stty -F /dev/ttyACM0 -a`

```
001: //   $Id: Status.cpp,v 1.5 2019/02/17 22:29:00 pm Exp $
002: /* |-------------------------------------------------------------|
003:    |Status.cpp  Seismic Recorder called by TK Program            |
004:    |Author: Dr. P. Michaels, PE <pmsolid@cableone.net>           |
005:    |or <paulmichaels@boisestate.edu>                             |
006:    |  NOTICE:                                                    |
007:    |  Copyright (C) 2019 Paul Michaels                           |
008:    |  This program is free software; you can                     |
009:    |  redistribute it and/or modify it under the terms           |
010:    |  of the GNU General Public License as published             |
011:    |  by the Free Software Foundation; either version            |
012:    |  3 of the License, or (at your option) any later            |
013:    |  version.  This program is distributed in the               |
014:    |  hope that it will be useful, but                           |
015:    |  WITHOUT ANY WARRANTY; without even the implied             |
016:    |  warranty of MERCHANTABILITY or FITNESS FOR A               |
```

```
017:    |  PARTICULAR PURPOSE.  See the GNU General Public           |
018:    |  License for more details.                                 |
019:    |  You should have received a copy of the GNU                |
020:    |  General Public License along with this program;           |
021:    |  If not, see <https://www.gnu.org/licenses/>               |
022:    |-------------------------------------------------------------|
023:  */
024: #include <termios.h>
025: #include <cstdio>
026: #include <cstring>
027: #include <string.h>
028: #include <unistd.h>
029: #include <fcntl.h>
030: #include <errno.h>
031: #include <stdlib.h>
032: #include <time.h>
033:
034:  short debug = 1;
035:  short serialfd;
036:  struct termios attribs;
037:     char deviceserial[] = "/dev/ttyACM0";
038:  speed_t speed;
039:  short usbnumber = 0;
040:
041: //prototypes
042:  void  setusb(short usbnumber);
043:
044: int main () {
045:  if (access("/dev/ttyACM0",F_OK) != -1 ) {
046:  sprintf(deviceserial,"/dev/ttyACM0");
047:  usbnumber = 0;
048:  puts("Device: ttyACM0"); }
049:
050:
051:  if (access("/dev/ttyACM1",F_OK) != -1 ) {
052:  sprintf(deviceserial,"/dev/ttyACM1");
053:  usbnumber = 1;
054:  puts("Device: ttyACM1"); }
055:
056:  if (usbnumber == 0 || usbnumber == 1) {
057:  serialfd = open(deviceserial,O_RDWR | O_NOCTTY | O_NDELAY);
058:  if (serialfd == -1) {
059:  puts("Not Connected");
060:  return (0);
061:  //fprintf(stdout,"ABORT failed to open port \n");
062:  } //endif
063:  }
064:  else {
065:  puts("Device Not found");
066:  return(0);
067:  } //endif
068:
069: /* Set USB */
070:  setusb(usbnumber);
071:
072: /* Check that serialfd points to a TTY device */
073:  if(!isatty(serialfd)) {
074:  fprintf(stdout,"ABORT, not TTY\n");
075:  return(0); } //endif
```

```
076:
077: /* get the current settings */
078:  if(tcgetattr(serialfd, &attribs) < 0) {
079:  fprintf(stdout,"ABORT, can not get configuration\n");
080:  return(0);} //endif
081:
082:
083: if(debug !=0 ) {
084:  speed = cfgetispeed(&attribs);
085:  if (speed == 4098) puts("BAUD: 115200");
086: // fprintf(stdout,"input speed: %lu\n", (unsigned long) speed);
087: // speed = cfgetospeed(&attribs);
088: // fprintf(stdout,"output speed: %lu\n", (unsigned long) speed);
089: // puts("---------------\n");
090:  } //endif
091:
092:   return(0);
093: } //endmain
094:
095: void  setusb(short usbnumber) {
096: char command0[] = "stty -F /dev/ttyACM0 -parenb -parodd -cmspar cs8 115200 \
097:   -hupcl -cstopb cread clocal -crtscts \
098: -ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl ixon -ixoff \
099: -iuclc -ixany -imaxbel -iutf8 \
100: -opost -olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 \
101: -isig -icanon iexten -echo echoe echok -echonl -noflsh -xcase -tostop -echoprt \
102: echoctl echoke -flusho -extproc";
103: char command1[] = "stty -F /dev/ttyACM1 -parenb -parodd -cmspar cs8 115200 \
104:   -hupcl -cstopb cread clocal -crtscts \
105: -ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl ixon -ixoff \
106: -iuclc -ixany -imaxbel -iutf8 \
107: -opost -olcuc -ocrnl -onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 \
108: -isig -icanon iexten -echo echoe echok -echonl -noflsh -xcase -tostop -echoprt \
109: echoctl echoke -flusho -extproc";
110:  if (usbnumber == 0) {
111:  system(command0);}
112:  else {
113:  system(command1);}
114: } //endsetusb
```

7.2.4 SEIS16Rv2.cpp

```
001: // $Id: SEIS16v2.cpp,v 1.6 2019/03/29 22:06:55 pm Exp $
002: /* |-------------------------------------------------------------|
003:    |SEIS16v2.cpp  Seismic Recorder called by TK Program          |
004:    |Author: Dr. P. Michaels, PE <pmsolid@cableone.net>           |
005:    |or <paulmichaels@boisestate.edu>                             |
006:    |  NOTICE:                                                    |
007:    |  Copyright (C) 2019 Paul Michaels                           |
008:    |  This program is free software; you can                     |
009:    |  redistribute it and/or modify it under the terms           |
010:    |  of the GNU General Public License as published             |
011:    |  by the Free Software Foundation; either version            |
012:    |  3 of the License, or (at your option) any later            |
013:    |  version.  This program is distributed in the               |
014:    |  hope that it will be useful, but                           |
015:    |  WITHOUT ANY WARRANTY; without even the implied             |
016:    |  warranty of MERCHANTABILITY or FITNESS FOR A               |
017:    |  PARTICULAR PURPOSE.  See the GNU General Public            |
```

```
018:    |  License for more details.                                  |
019:    |  You should have received a copy of the GNU                 |
020:    |  General Public License along with this program;            |
021:    |  if not, write to the Free Software Foundation,             |
022:    |  Inc., 675 Mass Ave, Cambridge, MA 02139, USA.              |
023:    |-------------------------------------------------------------|
024:  */
025: // UNIPOLAR VERSION
026: #include <stdio.h>
027: #include <termios.h>
028: #include <cstdio>
029: #include <cstring>
030: #include <string.h>
031: #include <unistd.h>
032: #include <fcntl.h>
033: #include <errno.h>
034: #include <stdlib.h>
035: #include <time.h>
036: #include <math.h>
037: #include <float.h>
038: #include <sys/ioctl.h>
039: #define nchar 20
040: #define Nbits 16
041: #define RNG 5.0 //set for +/- 5 Volts (Arduino RANGE5V)
042: //#define RNG 10.0 //set for +/- 10 Volts (Arduino RANGE10V)
043: #define BLOCKLEN 1900 // if hangs at bytes_avail, then decrease BLOCKLEN
044:
045:
046:
047: //Function Prototypes
048:  int   setprop(int serialfd, struct termios attribs, short debug);
049:  int   CheckDTnchan(int ndt, int nchannels);
050:  int   getparm(int argc,char *argv[],
051:          int *nsamp, int *ndt, int *nchannels, int *nstack,
052:  int *fnum, char *source, char *phones, short *nSP,float *sx,
053:  float rx[]);
054:  float Time(int *sendcount, int nchannels,int ndt, int *tmod,
055:  int *tcount);
056:  int  saveplot (int y[], int npts, int nchannels, int ndt, int chnl, int BIAS );
057:  int  meanval(int y[],int npts, int sendcount, int nchannels, int chnl, int BIAS) ;
058:  void writestring(char STRING[], char RES, char LTC,FILE *h1);
059:  int  WriteSEG2(int npts, int nchnl, float fsamin, int s1[],
060:       int ybase[], char ofile[], int stack, char source[],char phones[],
061:       short nSP, float sx, float rx[]);
062:
063: int main(int argc, char *argv[]) {
064:     struct termios attribs;
065:     short serialfd;
066:
067:     char snd[nchar+1];
068:     char c;
069:     int  i;
070:     unsigned short  ivalue= 0;
071:     short ivalueLow;
072:     short ivalueHigh;
073:     int  lvalue = 0;
074:     int nsamp;
075:     int ndt;
076:     int icount=0;
```

```
077:     int istop=0;
078:     int nstack = 1;
079:     int nchannels = 1;
080:     char fname[10];
081:     unsigned char Lbyte;
082:     unsigned char Hbyte;
083:     short debug = 0;
084:     int fnum = 0;
085:     int chnl; //channel number to plot
086:     float fsamin;
087:     short nSP;
088:     float sx;
089:     float rx[8];
090:
091: //    clock_t start,stop;
092: //    float deltexe;
093:     char deviceserial[] = "/dev/ttyACM0";
094:     //const int imax = floor(pow((double)2.,(double)(Nbits-1)));
095:     const unsigned int imax = floor(pow((double)2.,(double)(Nbits)));
096:     const float diff = RNG/(float)imax;  //volts/count
097:     float mvolts;
098:     int    microvolts;
099:     int   mspace;
100:     int  *s1;
101: int bytes_avail;
102: int sendcount = -1;
103: int block = 0;
104: int left = 0;
105: int nmax  ;
106: short iard;
107: int ybase[8];
108: char source[2];
109: char phones[10];
110:
111: /* Alternative Recording of Data with or without DC level
112: Set BIAS to virtual ground value in Microvolts above chassis ground */
113:
114: int BIAS = 0;  //if BIAS == 0 then compute mean and subtract, no DC saved
115: //int BIAS = 1949000; // BIAS used instead of mean value, DC saved 5Volt case
116: //int BIAS = 3874000; // BIAS used instead of mean value, DC saved 9Volt case
117:
118:
119:  if (access("/dev/ttyACM0",F_OK) != -1 ) {
120:  sprintf(deviceserial,"/dev/ttyACM0");
121:  //fprintf(stdout,"Serial Device = /dev/ttyACM0 \n");
122:  }
123:
124:  if (access("/dev/ttyACM1",F_OK) != -1 ) {
125:  sprintf(deviceserial,"/dev/ttyACM1");
126:  //fprintf(stdout,"Serial Device = /dev/ttyACM1 \n");
127:  }
128:
129: // open serial port
130:  serialfd = open(deviceserial,O_RDWR | O_NOCTTY | O_NDELAY);
131:  if (serialfd == -1) {
132:  fprintf(stderr,"ABORT failed to open port \n");
133:  } //endif
134:
135:
```

```
136: // set serial properties
137:        if (setprop(serialfd, attribs, debug) != 0) {
138:  perror(" setprop failure"); }
139:
140: //start loop
141:  while (istop != 1) {
142:
143:
144: // Send number of samples to arduino
145: // fprintf(stdout,"Clocks per second = %ld \n",CLOCKS_PER_SEC);
146:      if (icount == 0) {
147:
148: // blank snd array
149:  for (i=0; i<nchar; i++) {
150:  snd[i]=' ';
151:  }
152:  snd[nchar]='\0';
153:
154:  if(getparm(argc,argv,&nsamp,&ndt,&nchannels,&nstack,&fnum,
155:  source,phones,&nSP,&sx,rx) !=0) return(1);
156:
157:   nmax = nsamp*nchannels ;
158:
159: // ndt =sample interval in microseconds
160:  fsamin = ((float) ndt)/1.E6;
161:         sprintf(fname,"%4.4d.DAT",fnum);
162:
163: // Allocate RAM for recording
164:  mspace = nsamp*nchannels;
165:  s1 = (int*) calloc(mspace,sizeof(int));
166:            if(s1==NULL)
167:          {
168:           fprintf(stderr,"ABORT:s1 can't allocate enough RAM \n");
169:           fprintf(stderr," npts = %d",mspace);
170:           return(1);
171:          }
172:
173: // Zero out s1
174:  for (i=0; i<mspace; i++) s1[i]=0;
175:
176: // Check overhead for specified sample interval
177:  ndt = CheckDTnchan(ndt,nchannels);
178:  if (ndt == 0) {
179:  puts("ABORT !! sample interval channel error");
180:  return (1);}
181:
182:         if(debug !=0 ) fprintf(stderr,"ndt =%d  nchannels=%d \n",ndt,nchannels);
183:
184:  sprintf(snd,"%5.5d %5.5d %5.5d",nsamp,ndt,nchannels);
185:  snd[nchar-1]='\n';
186:  snd[nchar]='\0'; }
187:  else {
188:  sprintf(snd,"%5.5d %5.5d %5.5d",nsamp,ndt,nchannels);
189:  snd[nchar-1]='\n';
190:  snd[nchar]='\0';
191: // puts("ready for trigger ");
192:      } //endif
193:
194: //        fprintf(stdout,"snd=:%s \n",snd);
```

```
195:
196: //send paramaters to Arduino
197:  if(write(serialfd,&snd,nchar) < 0) {
198:  perror("failed to write");
199:  }
200:  if(debug !=0) fprintf(stdout,"snd = %s \n",snd);
201: //----------------------------test code------------------------
202:
203:  iard =0;
204:  while (iard ==0) {
205:  ioctl(serialfd,FIONREAD,&bytes_avail);
206:  if (bytes_avail >= 0) {
207:          read(serialfd,&c,1);
208:  if (c == 'H' ) {
209:     //fprintf(stdout,"%2.2d Arduino Ready:  ",icount);
210:     fprintf(stdout,"Stack %2.2d: ",icount+1);
211:  iard=1;
212:  }
213:  }
214:  } //endwhile
215:
216: //----------------------------test code------------------------
217:
218:
219: // puts("ready for trigger ");
220:
221: i=0;
222: Hbyte=0;
223: Lbyte=0;
224: c='N';
225:  sendcount = -1;
226:  block = 0;
227:  left = 0;
228:  nmax = nsamp*nchannels ;
229: // check if more than buffer bytes to send
230:  if (nmax > BLOCKLEN) {
231:  block = BLOCKLEN;
232:  left = nmax - BLOCKLEN;}
233:  else {
234:  block =nmax;
235:  left = 0; }
236:
237:   while(c !='q')   // Receive from Arduino until 'q' received
238:   {
239:
240:
241:  ioctl(serialfd,FIONREAD,&bytes_avail);
242: // fprintf(stderr,"available =%d \n",bytes_avail);
243:
244:          if (bytes_avail >= 2*block )   // check how many bytes available
245:         {
246:  for (i=0; i<block; i++) {
247:          read(serialfd,&Lbyte,1);
248:   ivalue=Lbyte;
249:   read(serialfd,&Hbyte,1);
250:   ivalue=Hbyte*256+ivalue;
251:          mvolts = ((float)(ivalue) * diff)*1000.0;
252:   microvolts = floor( (mvolts * 1000.0 +.5)) ;
253:
```

```
254: //**    stime=Time(&sendcount,nchannels,ndt,&tmod,&tcount);
255:  sendcount = sendcount + 1;
256: //      stime=((float) (tcount-1) * ndt)/1E6; // time (seconds)
257: //      fprintf(h1," %f %f \n",stime,mvolts);
258: //**    fprintf(h1," %f %8.8ld \n",stimfsamine,microvolts);
259: //      fprintf(h1," %f %d \n",stime,ivalue);
260:  s1[sendcount] = microvolts + s1[sendcount];
261:
262:  } // next i
263:
264: //      fprintf(stderr,"left = %d",left);
265:  if (left == 0) {
266:  c = 'q'; } //endif
267:
268:  if (left >BLOCKLEN) {
269:  left = left - BLOCKLEN;
270:  block = BLOCKLEN;} //do another block of samples
271:  else {
272:  block = left;
273:  left = 0;} //endif
274:
275:         } //endif bytes_avail
276:  }  //end while c !='q'
277:  if(debug != 0) fprintf(stderr,"finished q \n");
278:
279: // Get duration in msec
280:  bytes_avail=0;
281:         while (bytes_avail < 4) {
282:  ioctl(serialfd,FIONREAD,&bytes_avail);
283:  }
284: //         if (bytes_avail == 4) {
285:  if(debug != 0) fprintf(stderr,"bytes avail=%d \n",bytes_avail);
286: // Lower 16 bits
287:          read(serialfd,&Lbyte,1);
288:   ivalue=Lbyte;
289:   read(serialfd,&Hbyte,1);
290:   ivalueLow=Hbyte*256+ivalue;
291: // Higher 16 bits
292:          read(serialfd,&Lbyte,1);
293:   ivalue=Lbyte;
294:   read(serialfd,&Hbyte,1);
295:   ivalueHigh=Hbyte*256+ivalue;
296:
297:   lvalue = ivalueHigh << 16;
298:   lvalue =ivalueLow + lvalue;
299:
300:
301:  fprintf(stdout,"Record Length %d (msec) \n",lvalue);
302: // } //endif 4 bytes
303:
304:
305:
306:    if(debug !=0 ) fprintf(stdout," %d Samples Read by SEIS16v2 \n",sendcount);
307:    icount += 1;
308:    if(icount > nstack-1 ) istop =1;
309:
310:
311:  } //end while istop !=1
312:
```

```
313: // Open Output File
314:  icount = 0;
315: /*
316:         sprintf(fname,"%8.8d.dat",fnum);
317:  FILE *h1;
318:  if ( (h1=fopen(fname,"w"))==NULL)
319:         {
320:           printf("\n ABORT !! Error opening output.dat");
321:           exit(1);
322:         }
323:
324: // write from RAM to a file
325:  int stime;
326:  int tmod;
327:  int tcount;
328:  sendcount=-1;
329:  for (i = 0; i < (nmax); i++)
330:  {
331:  stime=Time(&sendcount,nchannels,ndt,&tmod,&tcount);
332:         fprintf(h1," %f %8.8d \n",stime,s1[i]);
333:  } //next sendcount
334:    fclose(h1);
335: */
336: // find mean value each channel
337:  for (chnl =0; chnl < nchannels; chnl++) {
338:  sendcount = -1;
339:  ybase[chnl] = meanval(s1,nmax,sendcount,nchannels,chnl,BIAS);
340: //fprintf(stderr,"meanval = %d \n",ybase[chnl]);
341:  }
342:
343:
344: // write from RAM to a file
345: /*NOTE:  WriteSEG2 removes mean from data so SEG-2 file will have traces with
346:  zero mean. If run before saveplot(), the plotx.png files will also be
347:  without mean. If you want to see the mean, move saveplot() loop (below) above
348:  WriteSEG2() call.  */
349:  WriteSEG2(nsamp,nchannels,fsamin,s1,ybase,fname,nstack,source,phones,nSP,
350:  sx,rx);
351:
352: // write plot files (channel numbers 0 to 7).  Move above WriteSEG2 to show mean
353:  for (chnl = 0; chnl < nchannels; chnl ++) {
354:  saveplot (s1,nmax,nchannels,ndt,chnl,BIAS);
355:  } //next chnl
356:
357: //   read(serialfd,&c,1);
358:    close(serialfd);
359:    free(s1);
360:     return 0;
361: } //end main
362:
363:
364: float Time(int *sendcount, int nchannels, int ndt, int *tmod, int *tcount)
365:         {
366:   float stime;
367:   *sendcount = *sendcount + 1;
368:   *tmod = *sendcount % nchannels;
369:   if (*tmod == 0)
370:  {*tcount=*tcount + 1 ;}
371:          stime=((float) (*tcount-1) * ndt)/1E6; // time (seconds)
```

```
372:   return(stime);
373:  }  //end Time
374:
375:
376:
377: //------------SETPROP FUNCTION-----------------------------------
378: int setprop(int serialfd, struct termios attribs, short debug)
379: {
380:  speed_t speed;
381:
382:
383: /* Check that serialfd points to a TTY device */
384:  if(!isatty(serialfd)) {
385:  fprintf(stderr,"ABORT, not TTY\n");
386:  return(1); } //endif
387:
388: /* get the current settings */
389:  if(tcgetattr(serialfd, &attribs) < 0) {
390:  fprintf(stderr,"ABORT, can not get configuration\n");
391:  return(1);} //endif
392:
393: if(debug !=0 ) {
394: // debug print structure
395:  fprintf(stderr,"GET TERMIOS STRUCTURE\n");
396:  fprintf(stderr,"c_iflag: %08x \n",attribs.c_iflag);
397:  fprintf(stderr,"c_oflag: %08x \n",attribs.c_oflag);
398:  fprintf(stderr,"c_cflag: %08x \n",attribs.c_cflag);
399:  fprintf(stderr,"c_lflag: %08x \n",attribs.c_lflag);
400:  fprintf(stderr,"c_cc[NCCS]: %08x \n",attribs.c_cc[NCCS]);
401:  } //endif
402:
403:
404: // turn off input processing
405:  attribs.c_iflag &= ~(IGNBRK | BRKINT | ICRNL |
406:  INLCR | PARMRK | INPCK | ISTRIP | IXON);
407:
408: // No line processing
409:  attribs.c_lflag &= ~(ECHO | ECHONL | ICANON | IEXTEN | ISIG);
410:
411: // Turn off character processing
412:  attribs.c_cflag &= ~(CSIZE | PARENB);
413:  attribs.c_cflag |= CS8;
414:
415: // One byte is enought to return from read
416:  attribs.c_cc[VMIN] = 1;
417:  attribs.c_cc[VTIME] = 0;
418:
419: /* set the baudrate */
420: cfsetospeed(&attribs, B115200); /* outut baudrate */
421: cfsetispeed(&attribs, B115200); /* input baudrate */
422: //cfsetospeed(&attribs, 4096); /* outut baudrate */
423: //cfsetispeed(&attribs, 4096); /* input baudrate */
424:
425: if(debug !=0 ) {
426: // debug print structure
427:  fprintf(stderr,"\nSETTING TERMIOS STRUCTURE\n");
428:  fprintf(stderr,"c_iflag: %08x \n",attribs.c_iflag);
429:  fprintf(stderr,"c_oflag: %08x \n",attribs.c_oflag);
430:  fprintf(stderr,"c_cflag: %08x \n",attribs.c_cflag);
```

```
431:  fprintf(stderr,"c_lflag: %08x \n",attribs.c_lflag);
432:  fprintf(stderr,"c_cc[NCCS]: %08x \n",attribs.c_cc[NCCS]);
433:  } //endif
434:
435:
436: /* if there is need for it, set other settings here */
437:
438: /* eventually apply everything for your serialfd descriptor */
439: //if (tcsetattr(serialfd, TCSANOW, &attribs) < 0)
440: if (tcsetattr(serialfd, TCSANOW, &attribs) < 0)
441: {
442:         perror("stdin");
443:         return(1) ;
444:     }
445:
446: if(debug !=0 ) {
447:  speed = cfgetispeed(&attribs);
448:  fprintf(stderr,"input speed: %lu\n", (unsigned long) speed);
449:  speed = cfgetospeed(&attribs);
450:  fprintf(stderr,"output speed: %lu\n", (unsigned long) speed);
451:  puts("---------------\n");
452:  } //endif
453:
454:
455:   return(0);
456: }  //end setprop
457: //-----------------------------------------------------------------
458:
459: /* Function to check ndt > overhead, optimization 3 on arduino */
460:  int  CheckDTnchan(int ndt, int nchannels)
461:  {
462:  switch (nchannels) {
463:  case 1 :
464:  if (ndt < 1000) return(1000); else return(ndt);
465:  break;
466:  case 2 :
467:  if (ndt < 2000) return(2000); else return(ndt);
468:  break;
469:  case 3 :
470:  if (ndt < 2000) return(2000); else return(ndt);
471:  break;
472:  case 4 :
473:  if (ndt < 3000) return(3000); else return(ndt);
474:  break;
475:  case 5 :
476:  if (ndt < 3000) return(3000); else return(ndt);
477:  break;
478:  case 6 :
479:  if (ndt < 4000) return(4000); else return(ndt);
480:  break;
481:  case 7 :
482:  if (ndt < 4000) return(4000); else return(ndt);
483:  break;
484:  case 8 :
485:  if (ndt < 5000) return(5000); else return(ndt);
486:  default :
487:  return (0);
488:  } //endswitch
489:
```

```
490:  } //end CheckDTnchan
491:
492:  int getparm(int argc, char *argv[],
493:      int *nsamp, int *ndt, int *nchannels, int *nstack, int *fnum,
494:  char source[], char phones[], short *nSP, float *sx,
495:  float rx[])
496:  {
497:  int i;
498:
499:  if (argc == 1 ) {
500:  fprintf(stderr,"USAGE: %s nsamp ndt nchannels nstack file source phones SP SX RX \n",
501:  argv[0]);
502:  puts("Ex: SEIS16v2 1000 5000 8 1 0 H VRTVRTVV 0 5 1 2 3 4 5 6 7 8");
503:  puts("source: H=hammer W=weight_drop D=dynamite G=gun V=vibrator");
504:  puts("phones: V=vertical R=radial T=transverse (8 channels)");
505:  puts("SP: Shot point number, SX Shot point X-coord");
506:  puts("RX: Array of X-coordinates geophones");
507:  return(1); }
508:  else {
509:  if (argc >=2) *nsamp = atoi(argv[1]); else *nsamp = 1000;
510:  if (argc >=3) *ndt = atoi(argv[2]); else *ndt = 1000;
511:  if (argc >=4) *nchannels = atoi(argv[3]); else *nchannels = 1;
512:  if (argc >=5) *nstack = atoi(argv[4]); else *nstack =1;
513:  if (argc >=6 ) *fnum = atoi(argv[5]); else *fnum = 0;
514:  if (argc >= 7) strncpy(source,argv[6],1);
515:  if (argc >= 8) strncpy(phones,argv[7],8);
516:  if (argc >=9) *nSP = atoi(argv[8]); else *nSP = 0;
517:  if (argc >=10 ) *sx = atof(argv[9]); else *sx = 0;
518:  for ( i = 11; i<19; i++) {
519:  if (argc >=i ) rx[i-11] = atof(argv[i-1]); else rx[i-11] = 0;
520:  } //next i
521:  } //end else
522:  return(0);
523:  }  //end getparm
524:
525: // SAVE PLOT-----------------------------------------------------------------
526:  int  saveplot (int y[], int npts, int nchannels, int ndt, int chnl, int BIAS )
527:  {
528:  //y[] is in microvolts
529:  int tmod;
530:  int tcount;
531:  int sendcount;
532:  float stime;
533:  int itrace;
534:  char XL[9]="Time (s)";
535:  char    YL[23]="Amplitude (millivolts)";
536:  char Xwin[4]="png";
537:  char ofile[10];
538:  int   ybase =  0;
539:  int   dscale = 1;
540:  if (BIAS != 0 ) {dscale =1000;}
541:  sendcount = -1;
542:  ybase = meanval(y,npts,sendcount,nchannels,chnl,BIAS)/1000; // mv
543:  sprintf(ofile,"plot%1.1d.png",chnl);
544:  itrace = chnl;
545:  int FG; //fixed gain variable
546:  int DSFI; //descale factor inverse
547: /* Hardwired Gain is 19dbv (the 1000 microvolts to mv in gnuplot pipe) */
548:  FG = 19;
```

```
549:     DSFI = round ( (pow((double) 10., (double) ((float) FG)/20.0)));
550:
551:
552: /*
553:  FILE *h2;
554: //...open output file
555:         if( (h2=fopen(ofile,"w"))==NULL ) {
556:           fprintf(stderr,"\n ABORT !! Error opening file %s",ofile);
557:           return(1); }
558: */
559:
560: //  PIPE VERSION
561:  FILE *h2 =popen("gnuplot ","w");
562:
563:    fprintf(h2,"set grid \n");
564:    fprintf(h2,"set output \"%s\" \n",ofile);
565:    fprintf(h2,"set terminal %s size 640,240  \n",Xwin);
566:    fprintf(h2,"set key left \n");
567:    fprintf(h2,"set xlabel \" %s \" \n",XL);
568:    fprintf(h2,"set ylabel \" %s \" \n",YL);
569: // fprintf(h2,"set yrange [-500:500] \n");
570:  //note plot in mv ($2/1000) and data writen below corrected for k-gain DSFI
571:    fprintf(h2," plot '-' u ($1):($2/1000):($3) lc 'black' w filledc above t '' ,");
572:  fprintf(h2," '' u ($1):($2/1000) lc 'black' w l t \"  Trace %d \" \n",itrace+1);
573:    int j;
574:  tmod =0;
575:  tcount = 0;
576:  sendcount = -1;
577:    for(j=0;j<npts;j++) {
578:  stime=Time(&sendcount,nchannels,ndt,&tmod,&tcount);
579:  //Select chnl
580:  tmod = sendcount % nchannels;
581:  if (tmod == chnl) {
582:      fprintf(h2,"%f %d %d \n",stime,y[j]/DSFI,ybase/dscale);
583:  } //endif
584:    } //next j
585:    fprintf(h2,"e\n");  //this is required when std in - plot
586:  tmod =0;
587:  tcount = 0;
588:  sendcount = -1;
589:   for(j=0;j<npts;j++) {
590:  stime=Time(&sendcount,nchannels,ndt,&tmod,&tcount);
591:  //Select chnl
592:  tmod = sendcount % nchannels;
593:  if (tmod == chnl) {
594:      fprintf(h2,"%f %d %d \n",stime,y[j]/DSFI,ybase/dscale);
595:  } //endif
596:    } //next j
597:    fprintf(h2,"e\n");  //this is required when std in - plot
598:  fclose(h2);
599:  return(0);
600: //...close pipe
601:  fflush(h2);
602:  pclose(h2);
603:  } //end saveplot
604:
605: //COMPUTE MEAN============
606:  int meanval(int y[],int nmax, int sendcount,int nchannels,int chnl, int BIAS) {
607:  int i;
```

```
608:  long sum = 0;  //even though a sample fits, 1000's of samples gets big
609:  int average;
610:  int tmod;
611:  if (BIAS == 0 ) {
612:  for (i=0; i<nmax; i++) {
613:  sendcount = sendcount +1;
614:  tmod = sendcount % nchannels;
615:  if (tmod == chnl) {
616:  sum = sum + y[i];
617:  } //endif
618:  } //next i
619:  average = sum / (nmax/nchannels);
620:  } else {
621:  average = BIAS; }
622:  return(average);
623:  } //end meanval
624:
625: //========================WriteSEG2=============================================
626: int WriteSEG2(int npts, int nchnl, float fsamin, int s1[],
627:   int ybase[],char rawrecord[], int stack, char source[], char phones[],
628:  short nSP, float sx, float rx[])
629: {
630: //...declare variables
631:  time_t            start;
632:  struct tm        *ptr;
633:  char TIME[40], DATE[40];
634: //C++ requires char const, not just char
635:  char const *mon[12]={"Jan","Feb","Mar","Apr","May","Jun",
636:         "Jul","Aug","Sep","Oct","Nov","Dec"};
637:
638:  FILE    *h1;
639:  char ofile[12] ;
640:  int i, k;
641:  unsigned short int ID = 0x3a55;   //bytes 0 1
642:  unsigned short int REV = 1; //bytes 2 3
643:  /* Note: 8 channel recorder, but allow max 10 channels for pointers space
644:  This will permit some zeros for buffer to free format section */
645:  unsigned short int M ;  // size of trace point block NCH*4  //bytes 4 5
646:  unsigned short int NCH   ; // number of channels  //bytes 6 7
647:  char LENST = 0x01; // length of string terminator  (bytes) //byte  8
648:  char ST = 0; //string terminator byte //bytes 9 10
649:  char LT = 0x01; //length of line terminator character (bytes) //byte  11
650:  char LTC = 0x0a; // line terminator //byte  12
651:  char RES = 0x00; //reserved byte //bytes 14-31
652:  int ZERO4 = 0;   //4 byte zero
653:  short int ZERO2 = 0;   //2 byte zero
654:  unsigned short int TDBID = 0x4422;
655:  unsigned long int TPB, LOC1, LOC2, EFDB, EDB;
656:  int CHNL ;
657: // int stack ;
658:  char STRING[40];
659:  int FG; //fixed gain variable
660:  double DSF; //descale factor
661:
662:
663: //...set parameters
664:  char lineid[] = "0001";
665:  sprintf(ofile,"%s",rawrecord);
666:  float ry = 0.0,
```

```
667:  rz = 0.0;
668:  float  sy = 0.0,
669:  sz = 0.0;
670: // stack = 1;
671: // nchnl = 8;
672:
673:  char typerecvr[30];
674:  char typesrc[30];
675: // char source = 'H';
676:  switch (source[0]) {
677:  case 'H':
678:  sprintf(typesrc,"HAMMER");
679:  break;
680:  case 'W':
681:  sprintf(typesrc,"WEIGHT_DROP");
682:  break;
683:  case 'D':
684:  sprintf(typesrc,"DYNAMITE");
685:  break;
686:  case 'G':
687:  sprintf(typesrc,"GUN");
688:  break;
689:  case 'V':
690:  sprintf(typesrc,"VIBRATOR");
691:  break;
692:  default:
693:  sprintf(typesrc,"HAMMER");
694:  }
695:
696:  NCH = nchnl;
697:  CHNL = 0;
698:  M=32;  // NCH * 4 if NCH max = 8
699:
700:  for (i=0; i<40; i++) {
701:  TIME[i] = '\0';
702:  DATE[i] = '\0'; }
703:
704:
705: //...open output file
706:         if( (h1=fopen(ofile,"wb"))==NULL )
707:  {
708:  fprintf(stderr,"\n ABORT !! Error opening file %s",ofile);
709:  return(1);
710:  }
711:
712: //...GET TIME AND DATE
713:       time(&start);
714:       ptr=localtime(&start);
715:       sprintf(TIME,"  ACQUISITION_TIME %2.2u:%2.2u:%2.2u  ",
716:          (*ptr).tm_hour,(*ptr).tm_min,(*ptr).tm_sec);
717:       sprintf(DATE,"  ACQUISITION_DATE %2.2d/%3s/%4.4d  ",
718:          (*ptr).tm_mday,mon[(*ptr).tm_mon],(*ptr).tm_year+1900);
719:
720: //...FILE DESCRIPTOR BLOCK
721:  fwrite(&ID,sizeof(short int),1,h1); //bytes 0,1 ID
722:  fwrite(&REV,sizeof(short int),1,h1); //bytes 2,3 revision number
723:  fwrite(&M,sizeof(short int),1,h1);//bytes 4,5 size trace pointer block
724:  fwrite(&NCH,sizeof(short int),1,h1); //bytes 6,7 number of channels
725:  fwrite(&LENST,sizeof(char),1,h1);  // byte 8 size of ST
```

```
726:  fwrite(&ST,sizeof(char),1,h1); //byte 9   string term char used
727:  fwrite(&ST,sizeof(char),1,h1); //byte 10  2nd string term char
728:  fwrite(&LT,sizeof(char),1,h1); //byte 11  size LTC
729:  fwrite(&LTC,sizeof(char),1,h1); //byte 12 line term char
730:  fwrite(&ST,sizeof(char),1,h1); //byte 13 2nd line term char not used
731:  for (i=14; i<32; i++) {
732:  fwrite(&RES,sizeof(char),1,h1); }
733:
734: //...TRACEPOINTER SUBBLOCK
735: // TPB = ftell(h1);    //capture start of block to come back to later
736:  TPB = (long int) 0x20;
737:  for (i=0; i<8; i++) {
738:  fwrite(&ZERO4,sizeof(int),1,h1);} //zero out pointers to start
739:
740: //...Write Date of acquisition
741:  LOC1=ftell(h1);
742:  writestring(DATE,RES,LTC,h1);
743:
744: //...Write Time of acquisition
745:  writestring(TIME,RES,LTC,h1);
746:
747: //...Write Instrument
748:  sprintf(STRING,"  INSTRUMENT PM_Arduino Mayhew_A/D 01  ");
749:  writestring(STRING,RES,LTC,h1);
750:
751: //...Write Instrument
752:  char UNITS[40];
753:  sprintf(UNITS,"  UNITS METERS  ");
754:  writestring(UNITS,RES,LTC,h1);
755:
756: //...capture end of free format location
757:  EFDB=ftell(h1) - 1 ;
758: // printf("location end of FDB = %0lx \n",EFDB);
759:
760:  EDB = EFDB ;  //not really end of data block first time
761:  // EDB is also start of TDB later on
762:
763: /* START OF TRACE DESCRIPTOR BLOCK ==================================== */
764:
765:  for (k=0; k<nchnl; k++) {
766:
767: //return to trace pointer block start and write LOC2
768:  fseek(h1,TPB+k*4,SEEK_SET);
769:  LOC2= EDB + 1 ;  // add one to point beyond end of FDB or EDB
770:  fwrite(&LOC2,sizeof(int),1,h1);
771:
772:
773: //...Write Trace Descriptor block
774:  fseek(h1,LOC2,SEEK_SET);  //return to start of trace descriptor block
775: //printf("%08lx \n",LOC2);
776:  fwrite(&TDBID,sizeof(short int),1,h1);  //write ID code for TDB
777:
778:  LOC1=ftell(h1);  //start of trace descriptor block just after ID
779:  short int TDBS = ZERO2;
780:  fwrite(&TDBS,sizeof(short int),1,h1); //write zero temporary for block size
781:  int SB = 4*npts; //size of data block in bytes
782:  fwrite(&SB,sizeof(int),1,h1); //size of data block in bytes
783:  int NS = npts;  // number of samples in a data block
784:  fwrite(&NS,sizeof(int),1,h1); //number of samples in data block
```

```
785:  char DFC = 0x02; // Data Format Code for 32 bit integer
786:  fwrite(&DFC,sizeof(char),1,h1);
787:  /* Reserved */
788:
789: /* NOTE Strings must start with 2 spaces to allow offset to next replacement int
790:         and must end with 2 spaces to allow for \0 and \0a replacement (eofs eofl) */
791:  for (i=13; i<32; i++) {
792:  fwrite(&RES,sizeof(char),1,h1);
793:   }
794:  sprintf(STRING ,"  CHANNEL_NUMBER %d  ",CHNL+k);
795:  writestring(STRING ,RES,LTC,h1);
796:
797:  sprintf(STRING,"  ALIAS_FILTER %d %d  ",100,6);
798:  writestring(STRING,RES,LTC,h1);
799:
800:  sprintf(STRING,"  HIGH_CUT_FILTER %d %d  ",100,6);
801:  writestring(STRING,RES,LTC,h1);
802:
803:  FG = 19;  //gain in dBv (change if your instrument different)
804:  sprintf(STRING,"  FIXED_GAIN %d  ",FG);
805:  writestring(STRING,RES,LTC,h1);
806:
807:  sprintf(STRING,"  DELAY %.2f  ",0.);
808:  writestring(STRING,RES,LTC,h1);
809:
810:  sprintf(STRING,"  LINE_ID %s  ",lineid);
811:  writestring(STRING,RES,LTC,h1);
812:
813:  sprintf(STRING,"  RAW_RECORD %s  ",rawrecord);
814:  writestring(STRING,RES,LTC,h1);
815:
816:  switch (phones[k]) {
817:  case 'V':
818:  sprintf(typerecvr,"VERTICAL_GEOPHONE");
819:  break;
820:  case 'T':
821:  sprintf(typerecvr,"T_HORIZONTAL_GEOPHONE");
822:  break;
823:  case 'R':
824:  sprintf(typerecvr,"R_HORIZONTAL_GEOPHONE");
825:  break;
826:  default:
827:  sprintf(typerecvr,"VERTICAL_GEOPHONE");
828:  }
829:
830:  sprintf(STRING,"  RECEIVER %s  ",typerecvr);
831:  writestring(STRING,RES,LTC,h1);
832:
833:  sprintf(STRING,"  SOURCE %s  ",typesrc);
834:  writestring(STRING,RES,LTC,h1);
835:
836:  sprintf(STRING,"  SOURCE_STATION_NUMBER %4.4d  ",nSP);
837:  writestring(STRING,RES,LTC,h1);
838:
839:  sprintf(STRING,"  RECEIVER_LOCATION %.2f %.2f %.2f  ",rx[k],ry,rz);
840:  writestring(STRING,RES,LTC,h1);
841:
842:  sprintf(STRING,"  SOURCE_LOCATION %.2f %.2f %.2f  ",sx,sy,sz);
843:  writestring(STRING,RES,LTC,h1);
```

```
844:
845:  sprintf(STRING,"  SAMPLE_INTERVAL %.6f  ",fsamin);
846:  /* Arduino A/D -->SEIS16v2 s1[] is in microvolts, SEG-2 Standard
847:  requires a descaling factor applied to the SEG-2 data which will
848:  yield mvolts */
849:  writestring(STRING,RES,LTC,h1);
850:
851:     DSF = 1.000 / (1000. * pow((double) 10., (double) ((float) FG)/20.0));
852:  sprintf(STRING,"  DESCALING_FACTOR %14.7E  ",DSF);
853:  writestring(STRING,RES,LTC,h1);
854:
855:  sprintf(STRING,"  STACK %d  ",stack);
856:  writestring(STRING,RES,LTC,h1);
857:
858:  LOC2 = ftell(h1);
859:  TDBS = LOC2 - LOC1 + 2 ;  // add 2 bytes to include ID 0x4422
860:
861: //determine padding if needed for Trace Descriptor Block TDB
862: //...Size, TDBS must be divisible by 4
863:  int pad,p1;
864:  p1=TDBS % 4;
865:  if ( p1 != 0) {
866:  pad = 4-p1;
867:  TDBS = TDBS + pad;
868:  } //endif
869: //printf("pad = %d \n",pad);
870: // printf("Trace Descriptor Block Size (bytes) %d \n",TDBS);
871:  fseek(h1,LOC1,SEEK_SET);
872:  fwrite(&TDBS,sizeof(short int),1,h1); //write block size
873:  fseek(h1,LOC2,SEEK_SET);
874:
875:  if ( p1 != 0) {
876:  for (i=0; i<pad; i++) fwrite(&ST,sizeof(char),1,h1);
877:  }
878: // long LOC3;
879: // LOC3=ftell(h1);
880: // printf("LOC3 = %8lx \n",LOC3);
881:
882: //...WRITE DATA
883:  int *pr;
884:  pr = &s1[0] + k;
885:  /*NOTE: DC level is removed from data since pr changes signal
886:   on exit:  *pr = *pr - ybase[k] changes s1, a pointer */
887:  for (i=0; i<npts; i=i+1) {
888:  *pr = *pr - ybase[k];
889:  fwrite(pr,sizeof(int),1,h1);
890:  pr = pr + nchnl;
891:  } //end of data write
892:
893: //...save end of data block
894:  EDB = ftell(h1) - 1; //the last byte of 4byte integer
895:
896:  } // next k chanl
897:  fclose(h1);
898:  return(0);
899: } //end WriteSEG2
900:
901: void writestring(char STRING[], char RES, char LTC,FILE *h1)
902: {
```

```
903:  int nbytes, nbytesf;
904:  nbytes = fprintf(h1,"%s",STRING);
905:  fseek(h1,-nbytes,SEEK_CUR);
906:  nbytesf = nbytes + 1  ;  //replace 2 blanks at begin with stringlength
907: // printf("bytes %d \n",(int) nbytes);
908:  fwrite(&nbytesf,sizeof(short int),1,h1); //write number of bytes forward
909:  //to next 2byte forward of next
910:  fseek(h1,nbytes-3,SEEK_CUR);
911:  fwrite(&RES,sizeof(char),1,h1); //replace 2nd last char with \0 endofstring
912:  fwrite(&LTC,sizeof(char),1,h1); //replace last char with 0xa endofstring
913: }
```

8 GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of

protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the

extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its

content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified

it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the

Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or

authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under

this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of

this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT

HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
```

```
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate
parts of the General Public License.  Of course, your program’s commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.
```

9 Free Documentation License

GNU Free Documentation License
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the

publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of

the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering

more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified

Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
public access to a Transparent copy of the Document, and likewise
the network locations given in the Document for previous versions
it was based on. These may be placed in the "History" section.
You may omit a network location for a work that was published at
least four years before the Document itself, or if the original
publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
Preserve the Title of the section, and preserve in the section all
the substance and tone of each of the contributor acknowledgements
and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
unaltered in their text and in their titles. Section numbers
or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section
may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant. To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version. Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity. If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the

Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its
license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and
multiple identical Invariant Sections may be replaced with a single
copy. If there are multiple Invariant Sections with the same name but
different contents, make the title of each such section unique by
adding at the end of it, in parentheses, the name of the original
author or publisher of that section if known, or else a unique number.
Make the same adjustment to the section titles in the list of
Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History"
in the various original documents, forming one section Entitled
"History"; likewise combine any sections Entitled "Acknowledgements",
and any sections Entitled "Dedications". You must delete all sections
Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other
documents released under this License, and replace the individual
copies of this License in the various documents with a single copy
that is included in the collection, provided that you follow the rules
of this License for verbatim copying of each of the documents in all
other respects.

You may extract a single document from such a collection, and
distribute it individually under this License, provided you insert a
copy of this License into the extracted document, and follow this
License in all other respects regarding verbatim copying of that
document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate
and independent documents or works, in or on a volume of a storage or
distribution medium, is called an "aggregate" if the copyright
resulting from the compilation is not used to limit the legal rights
of the compilation's users beyond what the individual works permit.
When the Document is included in an aggregate, this License does not
apply to the other works in the aggregate which are not themselves
derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these
copies of the Document, then if the Document is less than one half of
the entire aggregate, the Document's Cover Texts may be placed on
covers that bracket the Document within the aggregate, or the
electronic equivalent of covers if the Document is in electronic form.
Otherwise they must appear on printed covers that bracket the whole
aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the

GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See https://www.gnu.org/licenses/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

    Copyright (c) YEAR YOUR NAME.

```
    Permission is granted to copy, distribute and/or modify this document
    under the terms of the GNU Free Documentation License, Version 1.3
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts,
replace the "with...Texts." line with this:

    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other
combination of the three, merge those two alternatives to suit the
situation.

If your document contains nontrivial examples of program code, we
recommend releasing these examples in parallel under your choice of
free software license, such as the GNU General Public License,
to permit their use in free software.
```