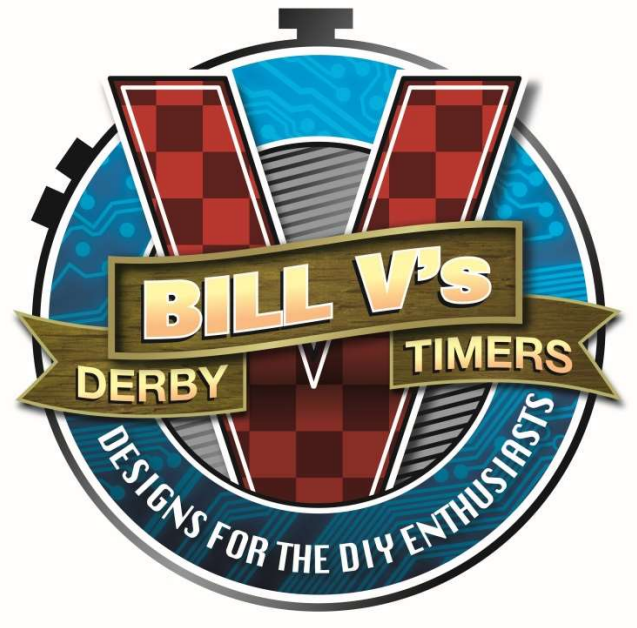
BILL V'S DERBY TIMERS

# Arduino Based Pinewood Derby 4-Lane Racetrack Timer

Minimum Hardware Design Version 4-2b

# USER MANUAL



# 1 INTRODUCTION

The Pinewood Derby racetrack timer discussed herein is a relatively simple, yet practical design that provides the features needed to conduct a successful race event. This version of the design operates with the authors' race management/display software running on a PC to control the race timer and display the race results.

The Arduino UNO™ board used in this project provides the interfaces to the optical lane sensors, race start (gate) switch and a reset switch and executes the software that performs the timing, monitors the track status and sends the race results and track status to the PC for display.

The race management software running on the PC provides display of the finish order, race times, and track status. Race times are displayed down to 0.0001 seconds.

Features included in this version of the race management software include:

- Ability to view results from previously run heats.
- Generation of a race results file for all heats ran.
- Ability to rerun a heat

# 2 SOFTWARE INSTALLATION

**NOTE:** The following instructions are for the installation of the customized Pinewood Derby Elapsed Time Timer software consisting of source code files PWD_ETTimer04-1b.ino, PWD_ETDisplay04_2b.pde and other files that support them.

## 2.1 Arduino UNO Software Installation

### 2.1.1 Arduino Software Environment Installation

Download and install the free Arduino Integrated Development Environment (IDE) onto your PC. For complete step-by-step instructions on how to download and install the Arduino Integrated Development Environment (IDE), visit https://www.arduino.cc/ . Versions are available for Windows, Mac OS X, and Linux. The environment makes it easy to write code and upload it to the board. The WEB site also provides tutorials to help you every step of the way.

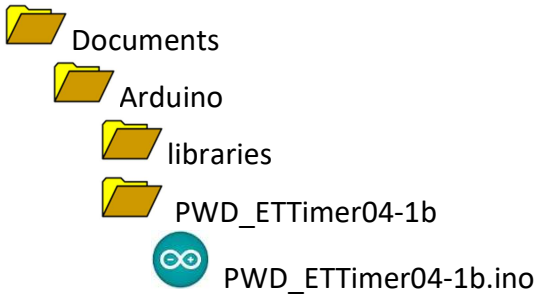
***NOTE**: Installation of the Arduino IDE also installs the necessary USB drivers for your PC to communicate with the Arduino via a serial RS-232 type (i.e. COM1, COM2, etc.) communication link. These drivers are also used by the race management software discussed in Section 2.2 below.*

### 2.1.2 Arduino Source Code Installation

Once you have the Arduino development environment installed (per Section 2.1.1) perform the following to install the Arduino source code onto your PC. Note that the Arduino source code file must reside in a folder having the same name (less the '.ino' file extension) in order to launch correctly.

1. Create a subfolder called 'PWD_ETTimer04-1b' under the "Arduino" folder. The "Arduino" folder was created during the install, usually under your Documents folder.
2. Copy the file 'PWD_ETTimer04-1b.ino' from the zip file provided and paste it into the 'PWD_ETTimer04-1b' folder created in the previous step.

This is a typical folder/file hierarchy but may vary:



### 2.1.3 Uploading the PWD_ETTimer code to your Arduino Board

Perform the following steps to upload the PWD_ETTimer code to the timer unit's Arduino microcontroller.

| Step | Action | Comments |
|---|---|---|
| 1 | Ensure your PC is powered up and ready. | |
| 2 | Connect the track timer (Arduino board) to your PC USB port via the USB cable. | Power indicator on the Arduino board illuminates. |
| 3 | Double click on the 'PWD_ETTimer04-1b.ino' file to launch the file and bring up the Arduino integrated development environment (IDE). | Arduino IDE window is displayed with the PWD_ETTimer04-1b source code listing. |
| 4 | Select the 'Tools/Board' pull-down menu to select/verify the Arduino Uno board is selected. | |
| 5 | Select the 'Tools/Port' pull-down menu to select/verify the COM port selection (i.e. COM1, COM2, etc.). | Arduino COM port is selected / verified. |
| 6 | Select "→" (upload) to start the compile and upload process. | The program will compile and automatically upload to the Arduino board. |
| 7 | **OPTIONAL**<br>To verify a successful upload, perform the diagnostic test procedure located at the end of this document. | |

Refer to the Arduino website for more in depth instructions if problems are encountered. Note that once the software has successfully been uploaded to the Arduino board, it is there permanently, unless overwritten by another upload. Hence, you only have to perform these steps once. Future use of the Arduino timer during your race events does not require an upload. Just connect the Arduino timer to your PC, launch the Display software and you should be ready to go.

## 2.2 Race Management / Display Software Installation

Download and install the free Processing integrated development environment (IDE) from the Processing.org website. For complete instructions on how to download and install the Processing development environment, visit https://processing.org. Versions are available for Windows, Mac OS X, and Linux. The WEB site also provides tutorials and step-by-step instructions to help you every step of the way.

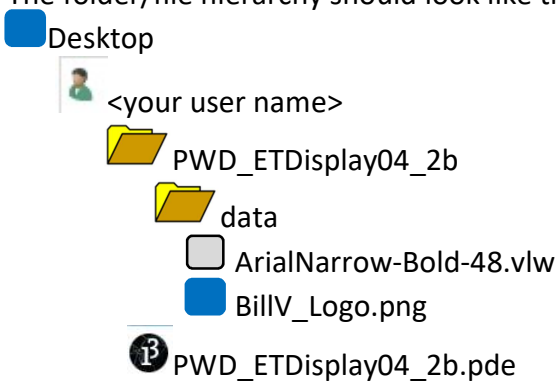
For Windows machines:

- Use File Explorer to view the contents of the Processing zip file (e.g. processing-3.5.3-windows64.zip) you downloaded. It should contain a folder called 'processing-x.x.x' (where x.x.x is the version#).
- Drag the 'processing-x.x.x' folder into your C:\Program Files\ folder.
- Double click 'processing.exe' to launch the program and cause it to install. If everything goes right it should create a Processing folder under your Document folder and start with the Processing IDE screen displayed.
- Exit Processing.

Once you have the Processing environment installed, perform the following to install the files associated with the race management/display software (see folder/file hierarchy example below). Note that the Processing source code file must reside in a folder having the same name (less the '.pde' file extension) in order to launch correctly.

1. Create a subfolder called 'PWD_ETDisplay04_2b' under your Desktop/Username folder.
2. Copy the file 'PWD_ETDisplay04_2b.pde' from the zip file provided and paste it into the 'PWD_ETDisplay04_2b' folder just created.
3. Now create a subfolder called 'data' under the 'PWD_ETDisplay04_2b' folder.
4. Copy the files 'ArialNarrow-Bold-48.vlw' and 'BillV_Logo.png' from the zip file provided and paste them into the 'data' folder.

The folder/file hierarchy should look like this (except for file icons):




### 2.2.1 Launching the Race Management Software

***NOTE**: In order for the race management software to initialize properly the Arduino based timer must be connected to the PC via a USB port and powered up.*

Ensure the timer is powered up and connected to your PC USB port via a USB cable before proceeding. Double click on the 'PWD_ETDisplay04_2b.pde' file to launch the file and bring up the Processing development environment. Select the Run button (▶) (upper left) on the PDE window to start the race management software. If the Arduino COM port is properly detected you should get a screen similar to Figure 1 below.

Other pop-up messages will appear if a COM port issue is detected and will require resolution to continue program initialization.

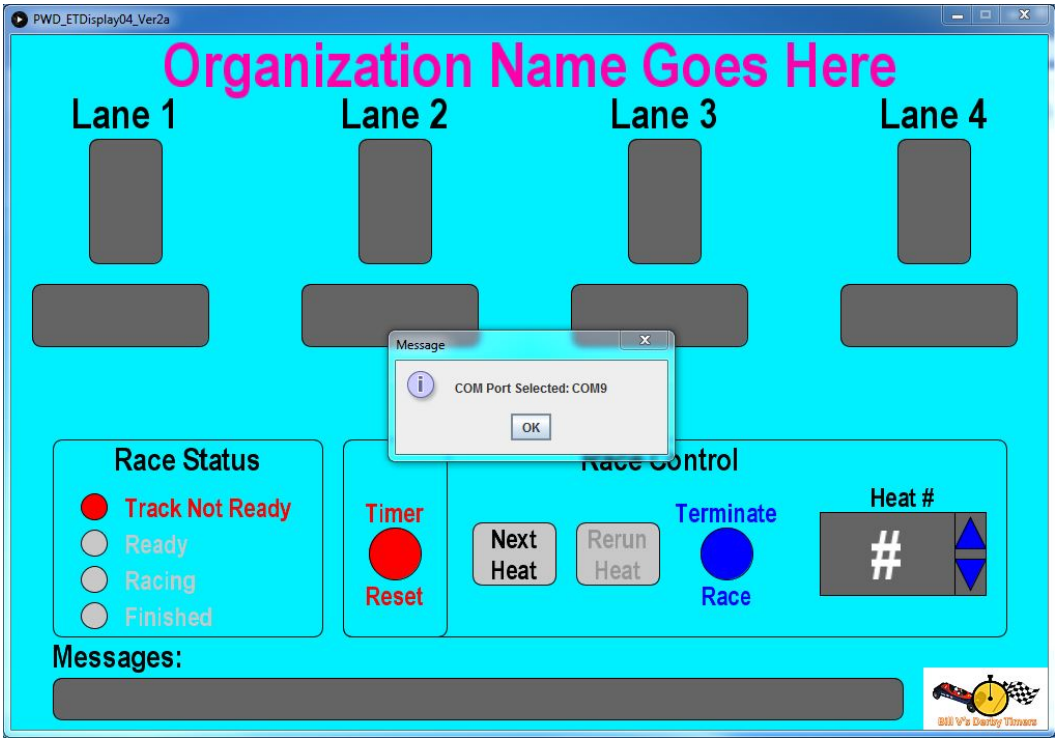



**Figure 1** - Example Program Startup Screenshot

Once a COM port has been successfully selected the opening screen as shown in Figure 2 will be presented.

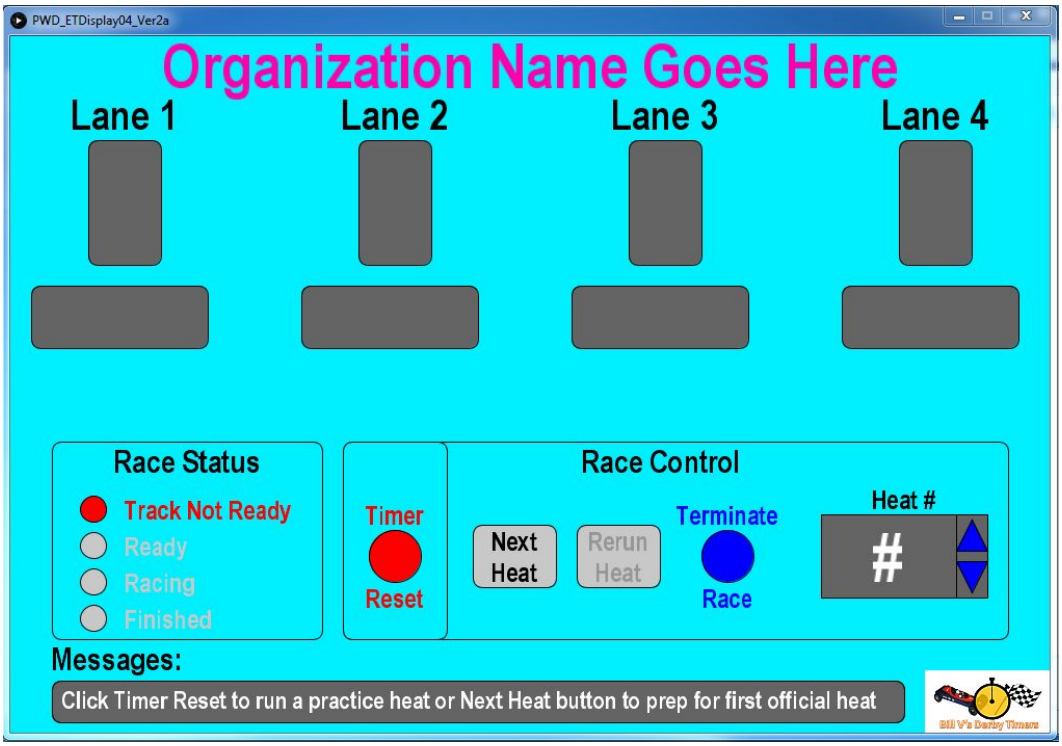



**Figure 2** - Opening Screen Example

### 2.2.2 Changing the Display Heading

This custom version of the race management software ('PWD_ETDisplay04_2b.pde') is delivered with the heading of "Organization Name Goes Here". The heading text can be changed to suit your needs. To change the heading, perform the following:

- In the Processing Display Environment scroll down to the "void setup()" section of the code and find the line: **text ("Organization Name Goes Here", 500,50); //Title – To be updated by the end user**
- Edit the text within the double quotes as desired. Do NOT change any other part of the line. Note that the text will automatically center on the screen when the program is run so you don't have to worry about adding extra spaces.
- Select save (Ctrl S) to permanently save your changes.

### 2.2.3 Changing the Background Color

To change the background color of the display, perform the following:

- In the Processing Display Environment scroll down to the "void setup()" section of the code and find the lines:
  - Rx = 0; //Set background color red hue
  - Gx = 240; //Set background color green hue
  - Bx = 255; //Set background color blue hue
- Refer to the color selector tool (found under 'tools') to determine the RGB values for the desired color.
- Modify the three values Rx, Gx, and Bx with the new values that range from 0 to 255 as determined using the color selector tool.
- Select save (Ctrl S) to permanently save your changes.

### 2.2.4 Making 'PWD_ETDisplay04_2b.pde' Code A Stand-alone Executable

The 'PWD_ETDisplay04_2b.pde' source file can be converted into a standalone JAVA executable so you won't have to launch the Processing Display Environment each time you use the program. It may require the installation of JAVA on your PC. Please visit https://processing.org for complete instructions on how to perform this task. This is an optional task and does not need to be performed to run the program.

# 3 TIMER OPERATION

## 3.1 Arduino Based Timer Operation Overview

Operation of the Arduino based timer is straight forward. The track timer code running on the Arduino cycles through four states as follows:

- Track Not Ready
- Ready
- Racing
- Finished

The "Track Not Ready" state is displayed at initial power-up. The "Track Not Ready" state is also displayed when the Arduino timer receives a reset command and it senses that one or more of the optical lane sensors is obstructed or the Gate Switch is in the wrong state to start the race. Either condition will cause an error message to be displayed in the message box at the bottom of the screen of the race management software running on the PC.

The "Ready" state is displayed when the timer receives a reset command and the optical lane sensors and Gate Switch are in the correct state to start the race.

The "Racing" state is displayed when the timer has sensed activation of the Gate Switch indicating the cars have left the gate and the race is underway.

The "Finished" state is displayed when all cars have crossed the finish line or when 10 seconds have elapsed, whichever occurs first. The 10-second time-out is for cases where a car fails to cross the finish line or a lane was not used. In those cases, dashes will be displayed for the lane time and the Finish Order box will remain blank.

## 3.2 Timer Operation – Using the Race Management/Display Software

The race management/display software running on the PC provides the human interface that allows for control of the timer and display of race results. Additionally it performs a number of race management functions and saves the race results to a data file for post-race processing.

Control of the timer is performed via mouse clickable buttons on the display. They perform the functions as listed in the table below. Additional detail is provided in Section 3.4 of this document.

**Table 1 – Display Button Functions**

| Button | Function |
|---|---|
| Next Heat | • Sends a Reset command to the Arduino timer to ready it for the next race.<br>• Increments the heat count to the next non-ran heat. |
| Rerun Heat | • Sends a Reset command to the Arduino timer to ready it for the next race<br>• Does NOT increment the heat count. |
| Timer RESET | • Sends a Reset command to the Arduino timer to ready it for the next race (Done in cases where the timer returned an error message such as the gate switch being in the incorrect state requiring a reset without incrementing the heat number). |

| Button | Function |
|---|---|
| Terminate Race | • Causes the race event to terminate.<br>• Displays the race terminated message.<br>• Disables the 'Next Heat' and 'Rerun Heat' buttons. |
| Heat Up/Down Arrows | • Increments/decrements the heat count.<br>• Allows viewing of prior ran heats. |

### 3.2.1 Program Start-up

***Note***: This version of the race manager software includes improved COM port detection/selection capability. At startup it will check the PC for all currently active COM ports and respond as follows:

- No COM ports Detected:
  - Displays a warning pop-up message informing the user no COM ports were detected and then exits the program when the user clicks on OK.
- One COM port Detected:
  - Assumes it is the Arduino COM port, automatically selects it and displays a pop-up message informing the user what COM port was selected.
- Two or more COM ports Detected:
  - Displays a pop-up providing a list of available COM ports and allows the user to select which COM port to use. Once selected, a second pop-up message is displayed informing the user what COM port was selected.
- Selected COM Port unavailable:
  - Displays a pop-up informing the user a COM port is not available and may be in use by another program.

Once the COM port selection has been satisfied the user will be presented with the opening screen (Fig. 2).

### 3.2.2 Running a Race Using the Race Management Software

Perform the following steps to run a race using the race management software:

1. Ensure the timer is powered up and connected to the PC.
2. Start the race management software and acknowledge the COM port selection.
3. Ensure the gate switch is closed and all lane sensors are un-obstructed
4. Click on the 'Next Heat' button (Refer to Fig. 3).
   a. The race status advances to the 'Ready' state if the track lane sensors and gate switch are in the correct state to start the race.
   b. The Heat# advances to first un-ran heat.
5. Stage the cars on the track at the start line.
6. When ready operate the gate release handle to launch the cars.
   a. When gate activation/release occurs the race status advances to the 'Racing' state.
7. Wait for all cars to cross the finish line or when 10 seconds has elapsed, whichever occurs first (See Fig. 4).
   a. Race results are displayed.
   b. The race status advances to the 'Finished' state
8. Repeat steps 4 through 7 till all heats have been ran.
9. At completion of the race click on the Terminate Race button to end the program. The race results file can be accessed and printed out using your favorite spreadsheet software as described in the next section (See Section 3.2.3).

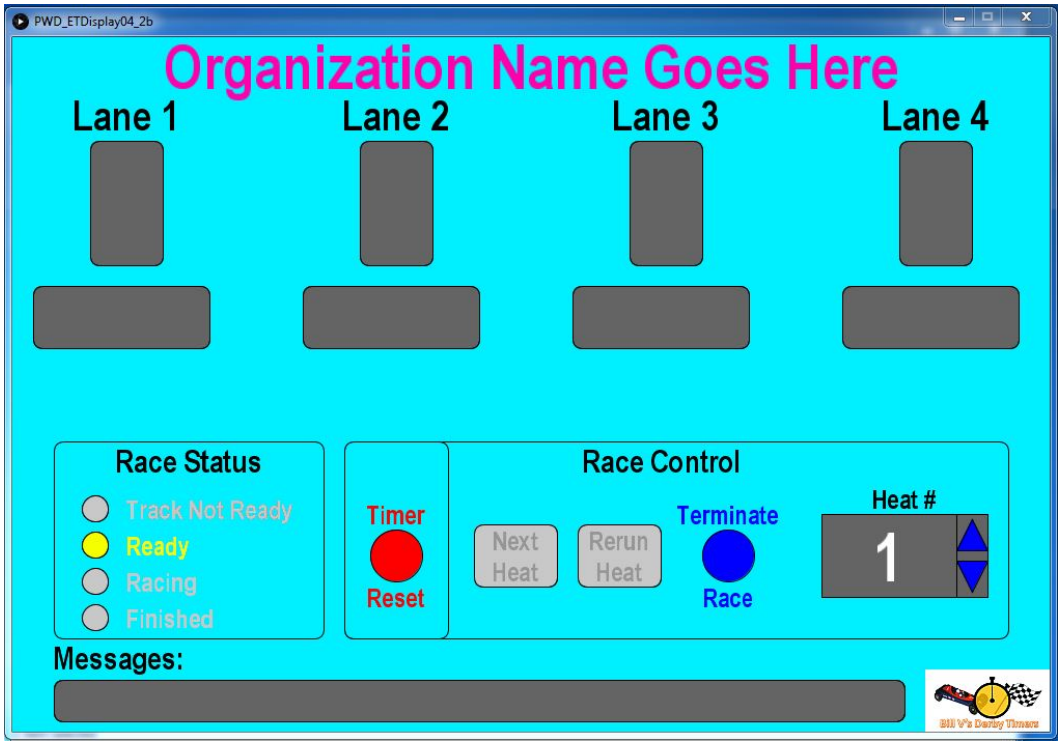



**Figure 3** – Ready To Race Example

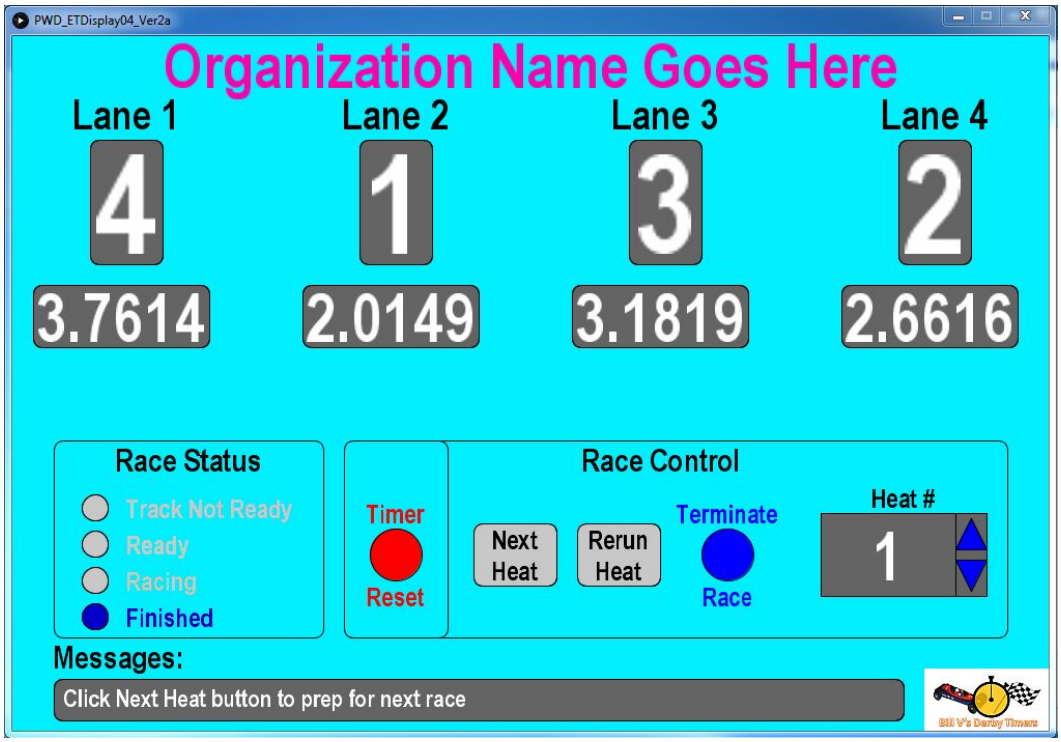



**Figure 4** – Race Finished Example

### 3.2.3 Race Results

The race results file is automatically updated at the completion of each heat to minimize the risk of losing any data. The race management software creates the race results report file from the data in the race results table in comma separated (CSV) text format. An example of a formatted version is shown in Figure 5.

| Heat | Lane1 | Lane2 | Lane3 | Lane4 |
|---|---|---|---|---|
| 1 | 2.4672 | 2.6761 | 2.6471 | 2.1234 |
| 2 | 2.3594 | 2.6925 | 3.1147 | 2.8830 |
| 3 | 3.3016 | 2.5291 | 3.3412 | 3.2964 |
| 4 | 3.3037 | 2.5161 | 2.9897 | 3.2971 |
| 5 | 3.4729 | 2.3243 | 3.0054 | 3.3345 |

**Figure 5 – Example Race Results Report**

#### 3.2.3.1 Race Results File Naming Convention

The filename for the data file saved to disk is created at program startup and includes a date/time value retrieved from the computer's clock. The filename structure is as follows:
"Race_Results_YYYYMMDDHHMMSS.csv", where: YYYY = 4 digit year, MM = 2 digit month, DD = 2 digit day, HH = 2 digit hour, MM = 2 digit minutes, and SS = 2 digit seconds. This guarantees a unique file each time the program is started and prevents any previous files from being accidentally overwritten in case the program has to be restarted.

By default the file is saved to the same folder containing the display source code (.pde). This can be changed by including the path in the filename variable (See the Processing.org website (https://processing.org) under the "saveTable" command).

## 3.3 Additional Information – Race Management Controls

### 3.3.1 Next Heat Button

The 'Next Heat' button when clicked sends a reset command to the Arduino timer to ready it for the next race. In addition it advances the heat count to the next available (not yet run) heat number. For example, if heats 1 – 5 where run and the down arrow in the heat box was clicked so the heat 2 results are displayed, clicking the 'Next Heat' button will advance the heat count to heat 6 since it is the next available (not yet run) heat number. The 'Next Heat' button is deactivated (greyed out) when the timer is in the "Ready" and "Racing" state.

### 3.3.2 Rerun Heat Button

The 'Rerun Heat' button when clicked sends a reset command to the Arduino timer to ready it for the next race. But unlike the "Next Heat' button it **does not** increment the heat count. Instead, the heat number remains unchanged to allow rerun of that heat. Note that when the 'Rerun Heat' button is clicked the user will be presented with a "Are you sure?" pop-up; thereby allowing the user to opt out in the case the selection was made in error. Note also that when a heat is rerun, prior results will be overwritten by the new race results. The 'Rerun Heat' button is only active (not greyed out) when the timer is in the "Finished" state.

### 3.3.3 Heat # Up/Down Arrow Buttons

The Up/Down arrow buttons in the Heat # window allow the user to increment/decrement the Heat number from heat 1 up to the number of heats ran. When selecting a heat that has already been performed, race results (times & finish order) are displayed for that heat.

### 3.3.4 Terminate Race Button

The 'Terminate Race' button provides an easy method to end and close the race management software. When clicked, the user will be presented with a "Are you sure?" pop-up allowing the user to continue in case the selection was made in error. If selected, the race management software will generate the Race Results data file using data available from any heats that were run prior to termination.

### 3.3.5 Timer Reset Button

The 'Timer Reset' button provides a method to issue an independent RESET command to the Arduino timer without having to select the 'Next Heat' or 'Rerun Heat' buttons and thereby possibly affecting the state of the race management software. This is typically done to clear any lane sensor or gate switch error messages reported by the Arduino timer.

# 4 Arduino Timer Communication

Communication between the Arduino based timer and the PC is via the USB interface which has been set up as a serial link running at 9600 baud, 8 bits, no parity, and 1 stop bit (9600/8-N-1). ASCII character strings transmitted between the timer and the PC are used to control the timer as described in the table below.

| ASCII Command | Direction | Description |
|---|---|---|
| R | PC-to-Arduino | Reset – Resets the Arduino timer |
| C | PC-to-Arduino | COM Check – Debug only. Sends the '@' character back to the PC as an acknowledge signal. |
| NRD | Arduino-to-PC | Not Ready – Informs the race management software the timer is in the 'Not Ready' state. |
| RDY | Arduino-to-PC | Ready – Informs the race management software the timer is in the 'Ready' state. |
| RAC | Arduino-to-PC | Racing – Informs the race management software the timer is in the 'Racing' state. |
| FIN | Arduino-to-PC | Finished – Informs the race management software the timer is in the 'Finished' state. |
| GSW | Arduino-to-PC | Gate Switch – Informs the race management software that the Gate Switch is in the wrong state to start the race. |
| TRK<br>-or-<br>TRK, x, y, … | Arduino-to-PC | Track Status – Informs the race management software that the track is not ready for the next race because the Gate Switch or one or more optical lane sensors are obstructed. If a lane sensor is obstructed the TRK message will include the effected lane numbers (separated by commas). |

In addition to the above serial messages the Arduino passes the race results to the PC display software via a single ASCII string having the following format:

**Times: a.aaaa b.bbbb c.cccc d.dddd**

where:

- a.aaaa = Lane 1 time
- b.bbbb = Lane 2 time
- c.cccc = Lane 3 time
- d.dddd = Lane 4 time

Troubleshooting: Use of the Arduino IDE serial monitor or another serial terminal program can be used to observe these commands. Refer to the diagnostic test procedure on the following pages to assist in troubleshooting any issues. Feel free to contact the author via email at billv923@outlook.com for additional help.

# DIAGNOSTIC TEST PROCEDURE

This test procedure was written to assist in testing and troubleshooting the Arduino based timer hardware and software.

Setup:

- Bring up the Arduino Integrated Development Environment (IDE) on your PC (Refer to Section 2.1)
- (Optional) Select and load the PWD_ETTimer04-1b.ino file.
- Select the 'Tools/Port' pull-down menu to select/verify the port selection (i.e. COM1, COM2, etc.).
- Open the serial monitor by clicking on the little magnifying glass near the upper right corner of the IDE display. Ensure the baud rate is set to 9600.

Perform the following test procedure.

| Step | Action | Expected Results |
|---|---|---|
| 1 | Ensure the Gate Switch is closed and the optical lane sensors are properly illuminated (not obstructed). | N/A |
| 2 | On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button. | The message "RDY" is displayed on the monitor.<br><br>If you get the message "GSW" followed by "NRD" the Gate Switch is in the 'not ready' state. Troubleshoot and correct the problem before continuing.<br><br>If you get the message "TRK, …" followed by "NRD" one or more optical lane sensors are not ready (not properly illuminated). Troubleshoot and correct the problem before continuing. |
| 3 | Obstruct (block) the light illuminating the Lane 1 optical sensor.<br><br>Then on the serial monitor enter the letter R in the command line, press ENTER or click on the Send button. | N/A<br><br>The message "TRK, 1" followed by "NRD" on the monitor indicating optical lane sensor for Lane 1 is not ready. |
| 4 | Repeat step 3 for Lanes 2 through 4. | Same as step 3 except the lane number will be different. |
| 5 | Set the Gate Switch to the open position.<br><br>On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button. | N/A<br><br>The message "GSW" followed by "NRD" is displayed on the monitor. |
| 6 | Set the Gate Switch back to the closed position. | N/A |

| Step | Action | Expected Results |
|---|---|---|
| 7 | Ensure the Gate Switch is closed and the optical lane sensors are properly illuminated. | N/A |
| 8 | On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button. | The message “RDY” is displayed on the monitor. |
| 9 | Momentarily toggle the Gate Switch to the open position and then back to the closed position. | The message “RAC” is displayed on the monitor. |
| 10 | Wait 10 seconds. | After 10 seconds:<br>• The following message is displayed: “Times: 9.9999 9.9999 9.9999 9.9999” followed by the message “FIN” on the monitor. |
| 11 | Ensure the Gate Switch is closed and the optical lane sensors are properly illuminated. | N/A |
| 12 | Momentarily depress the RESET switch. | The message “RDY” is displayed on the monitor. |
| 13 | Momentarily toggle the Gate Switch to the open position and then back to the closed position. | The message “RAC” is displayed on the monitor. |
| 14 | Wave your hand over the optical lane sensors to obstruct the light illuminating them before the 10 second timeout has elapsed. | The following message is displayed on the monitor: “Times: x.xxxx x.xxxx x.xxxx x.xxxx” followed by the message “FIN”, where x.xxxx is the recorded time for lanes 1, 2, 3 and 4. |
| 15 | Repeat steps 11 thru 14 as desired obstructing the light to the lane sensors in different sequences. | Same results as steps 11 thru 14. |
| 16 | On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button. | The message “RDY” is displayed on the monitor. |
| -- | TEST COMPLETE | |