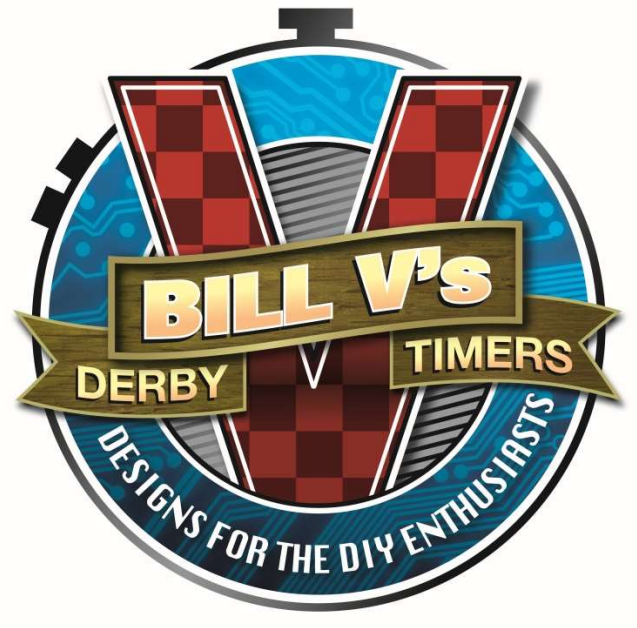
BILL V'S DERBY TIMERS

Arduino Based Pinewood Derby 6-Lane Racetrack Timer

Enhanced Design Version 5

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 requires the timer to run with the authors' race management software running on a PC to control the race timer and display the race results.

The Arduino UNO™ microcontroller board used in this project provides the interfaces to the optical lane sensors and race start (gate) 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. In addition, this version also provides display of racer names and car numbers imported from a user provided roster file.

Race management software features include:

- Race events that are run from a Race Plan which provides car lane assignments for each heat.
- Automatic generation of the race plan from a user provided roster.
- Display of car number and racer name assigned to each lane.
- Lane masking – both manual & automatic (aka: "BYE" lane).
- Ability to skip or rerun a heat.
- Ability to view results from previously run heats.
- Generation of the following reports for viewing/printing:
    - Race Plan – Print out is typically used to assist in staging cars for the race
    - Race results report – Provides lane times of each car for each heat ran
    - Standings report – Available at the completion of the race. Provides race standings (1'st, 2'nd, 3'd, etc.), average time and fastest time for each racer.

1.1 PC Requirements

The race management software is specifically tailored to run on Microsoft Windows based machines but should run on any machine that supports the MS Windows file structure format. It has been verified to run on MS Windows 7 and later versions including MS Windows 10. Additionally, the race management software uses your PCs default word processor and spreadsheet software for viewing and printing race results reports. In particular, your PC must be able to open files having the '.doc' and '.csv' file extensions. Software packages such as Microsoft Office, LibreOffice, Apache OpenOffice and other free office packages support these file formats.

2 SOFTWARE INSTALLATION

NOTE: The following instructions are for the installation of the Pinewood Derby Elapsed Time Timer software consisting of source code file PWD_ETTimer06-Ver5.ino, PWD_RaceManager06.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). 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.

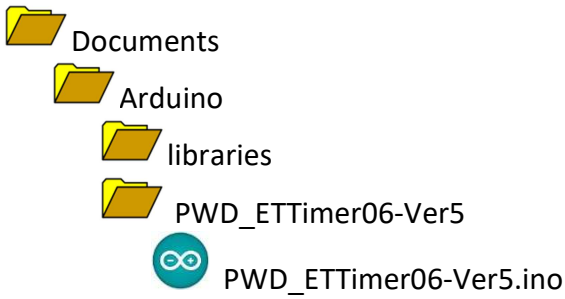
***NOTE**: Installation of the Arduino IDE also installs the necessary USB drivers for your PC to communicate with the Arduino via a serial RS-232 type (i.e. COM1, COM2, etc.) communication link. These drivers are also used by the race management software discussed in Section 2.2 below.*

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:

1. Create a subfolder called 'PWD_ETTimer06-Ver5' under the "Arduino" folder. The "Arduino" folder was created during the install, usually under your Documents folder.
2. Extract the file 'PWD_ETTimer06-Ver5.ino' from the zip file provided and copy it into the 'PWD_ETTimer06-Ver5' 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 a USB cable.	Verify the Arduino board is powered up (Power LED is illuminated)
3	Double click on the 'PWD_ETTimer06-Ver5.ino' file to launch the file and bring up the Arduino integrated development environment (IDE).	Arduino integrated development environment window is displayed with the PWD_ETTimer06-Ver5 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** 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 Race Manager software and you should be ready to go.

2.2 Race Management Software Installation

Download and install the free Processing3 integrated development environment (IDE) from the Processing.org website. For complete instructions on how to download and install the Processing3 development environment, visit https://processing.org. The WEB site also provides tutorials and step-by-step instructions to help you every step of the way. Note: This version is specifically developed for MS Windows PCs.

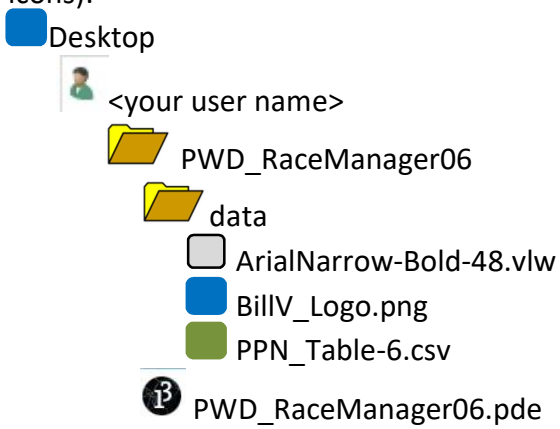
For Windows machines:

- Use File Explorer to view the contents of the Processing zip file (e.g. processing-3.5.3-windows64.zip) you downloaded from the processing.org website. 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 the Processing3 program.

Perform the following to install the files associated with the race management software (see folder/file hierarchy example below). Note that the Processing3 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 folder called 'PWD_RaceManager06' under your Desktop/Username folder.
2. Extract the file 'PWD_RaceManager06.pde' from the zip file and copy it into the 'PWD_RaceManager06' folder just created.
3. Extract the files 'Roster_Template.csv' and 'Sample_Roster.csv' from the zip file and copy them into the 'PWD_RaceManager06' folder as well.
4. Create a subfolder called 'data' under the 'PWD_RaceManager06' folder.
5. Extract the files 'ArialNarrow-Bold-48.vlw', 'BillV_Logo.png' and 'PPN_Table-6.csv' from the zip file and copy them into the 'data' folder.

Below is a typical folder/file hierarchy for the race manager program that should look like this (except for file icons):



This version of the race management software allows you to store the race roster files in the folder of your choice. Once the race roster is imported the software automatically remembers the location (i.e. file path) and will save race results and standing reports to the same location at completion of the race.

- - CAUTION - -

Do not store race roster files under folders such as 'program files' that are protected and require Administrator authorization to access. Doing so will cause the race manager program to crash.

Perform the following to create a folder under which your race documents such as race rosters will be saved.

1. Create a folder called 'Race_Documents' under your Desktop/Username folder.
2. Extract the files 'Roster_Template.csv', and 'Sample_Roster.csv' from the zip file and copy them into the 'Race_Documents' folder.

Below is a typical folder/file hierarchy for you race documents. Remember that the race manager software will automatically save the race reports (race plans, race results & race standings) to the same folder in which the race roster was imported from.

Desktop
- <your user name>
  - Race_Documents
    - Roster_Template.csv
    - Sample_Roster.csv

2.2.1 Launching the Race Management Software

Ensure the timer is connected to your PC via a USB cable and powered up before proceeding.
Double click on the 'PWD_RaceManager06.pde' file to launch the file and bring up the Processing3 integrated development environment. Select the Run button (▶) (upper left) on the PDE window to launch the race manager software. The software will open with the SETUP page displayed as shown in Figure 1 below.

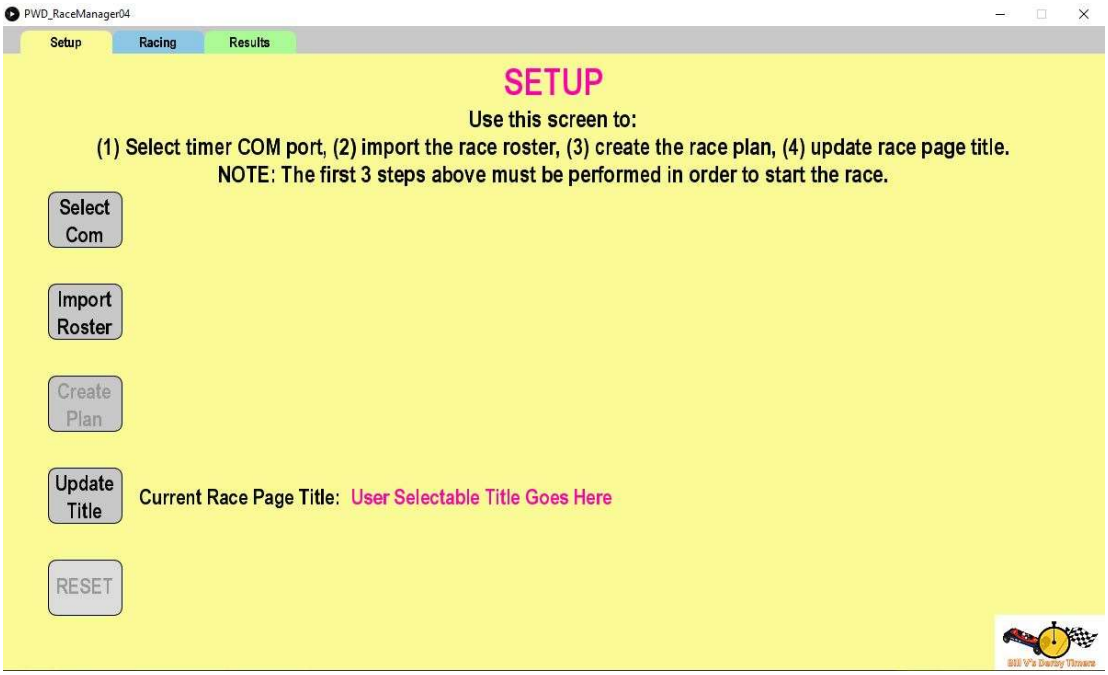


Figure 1 - Program Startup Screen

2.2.2 Making 'PWD_Racemanager06.pde' Code A Stand-alone Executable

The 'PWD_RacemManager06.pde' source file can be converted into a standalone JAVA executable so you won't have to launch the Processing3 Development 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 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 from the PC 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 from the PC 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. The 10-second time-out does not apply for lanes that have been masked and thus ignored by the timer.

3.2 Using the Race Management Software

The race management 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 such as lane assignment and saving race result data for post-race processing.

3.2.1 Roster Preparation

Prior to program start up the user must create a roster file containing the first name, last name, car number and a group name for each racer participating in the race (See Figure 4). Typically races are conducted one group at a time but the software does allow for more than one group (e.g. Tigers & Bears) to race together which is sometimes done if a group has a small number of members. In such cases, the 'Group' column is used to parse race results by group in the Group Standings report. User defined columns such as the 'CarCheck' column shown in the figure below may be added to the right of the four required columns. These extra columns will be ignored by the software. The file must have a header row with the header text exactly as shown in the figure below and be saved in comma-separated values (.csv) format.

LastName	FirstName	CarNo	Group	CarCheck
Gonzales	Speedy	5	Tigers	
McQueen	Lightning	2	Tigers	
Nogas	John	11	Bears	
Tired	Phlat	3	Bears	

Figure 2 – Example Roster File Format

The roster must contain a minimum of 2 but no more than 40 racers. Car numbers must be within the range of 1 to 99 and can contain no letters or other non-numeric characters. The roster will be used to automatically generate race plan that satisfies the Partial Perfect-N (PPN) criteria. These types of plans are the most commonly used by scouting and AWANA organizations and satisfy the conditions where (1) each car races the same number of times, (2) each car races in each lane and (3) where possible (depending on the number of races in the group), each car races against different opponents in each heat.

3.2.2 Race Setup

The race manager software has 3 displays consisting of a SETUP page, RACING page and RESULTS page. Each page is selected by clicking on the appropriate page select tab at the top left of the race management screen.

Prior to running a race the race manager software must be setup to (1) establish communication with the Arduino based timer hardware, (2) load a race roster with racer names and car numbers and (3) generate a race plan (schedule) from which racers are selected for each heat of the race. These tasks are accomplished on the SETUP page.

Perform the following steps to complete the race setup:

Step	Action	Comments
1	Connect the track timer to your PC USB port via a USB cable.	Verify the Arduino board is powered up (Power LED is illuminated)
2	Launch the race management software (Ref. Section 2.2.1) and verify the SETUP page is selected.	Race manager SETUP page is displayed (See Fig. 1).
3	Click on the Select Com button and follow instructions provided on the pop-up message windows.	• If COM port selection was successful, the selected COM port is displayed next to the Select Com button (See Fig. 3). • If COM port selection was unsuccessful, appropriate error messages will be displayed via pop-up text boxes and you must correct the problem before continuing.

Step	Action	Comments
4	Click on the Import Roster button and select the race roster file to import via the pop-up selection menus. Refer to Section 3.2.1 for creating a roster file.	The roster file will be loaded and checked for proper format and content. • If successful the total number of racers will be displayed to the right of the Import Roster button (See Fig. 3). • If unsuccessful, appropriate error messages will be displayed via pop-up text boxes and you must correct the problem before continuing.
5	Click on the Update Title button. A pop-up dialog box will open from which you can enter a race title (e.g. “Pack 768 Grand Prix”).	The race title is used as the header on the Racing Page and on the race plan document.
6	Click on the Create Plan button.	The race plan is generated and total number of heats in the race will be displayed to the right of the Create Plan button (See Fig. 3).
6a	**OPTIONAL**: Once the race plan is created select the Results page and click on the View/Print Race Plan button. This will launch your PC’s word processor from which you can view and print the document.	A hard-copy printout of the race plan is helpful in readying/staging the cars for the current and on-deck heats.

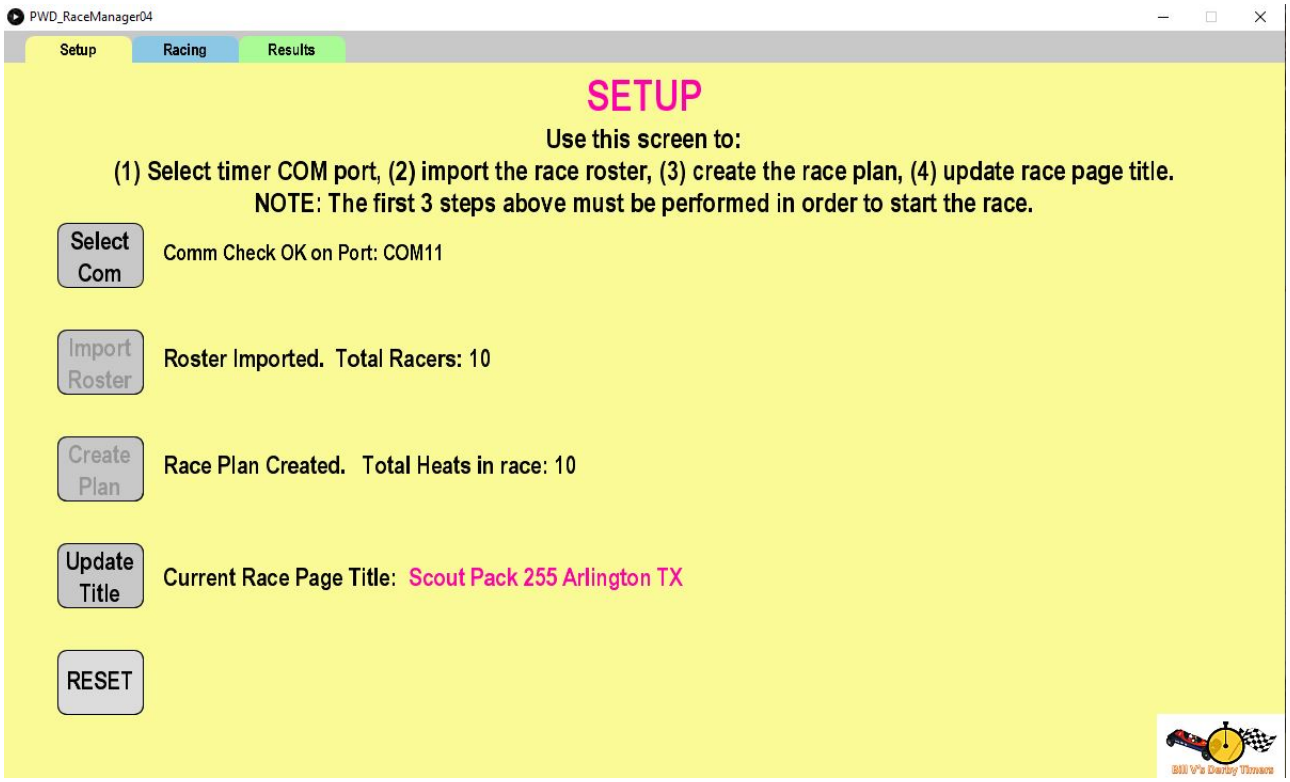


Figure 3 – Example Setup Page

3.2.3 Conducting the Race

Once all the race preparation steps have been completed the race event can begin. Click on the Racing tab to select the racing page. Refer to Figure 4 at the end of this section.

Control of the timer is performed via mouse clickable buttons on the racing page (Ref. Fig. 4). They perform the functions as listed in the table below. Additional detail is provided in Section 3.3 of this document.

Table 1 – Racing Page Button Functions

Button	Function
Next Heat	• Sends a Reset command to the Arduino timer to ready it for the next race. • Increments the heat count to the next non-ran heat. • Displays the car number and racer name assigned to each lane for that heat.
Rerun Heat	• Sends a Reset command to the Arduino timer to ready it for the next race • 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).
Gate Release (Not Activated)	• (Not functional in this release) Sends a gate release signal to the timer unit which in turn activates the gate release solenoid to launch the cars.
Terminate Race	• Causes the race event to terminate. • Displays the race terminated message. • Disables the 'Next Heat' and 'Rerun Heat' buttons. • Generates the Race Results and Standings data files using data available from any heats that were run prior to termination.
Heat Up/Down Arrows	• Increments/decrements the heat count. • Allows viewing of prior ran heats or advancement to un-ran heats in case you need to skip a heat. • Displays the car number and racer name assigned to each lane for that heat. • Displays the race results for that heat if that heat has already been run.
Lane Mask Checkboxes	• Alternately displays or clears a check-mark in the checkbox each time it is clicked (See Figure 10). • Sends the appropriate lane mask and un-mask commands to the Arduino timer.

3.2.4 Running a Race

Perform the following steps to run a race:

1. Ensure all the race preparation steps have been completed on the SETUP page per Section 3.2.2.
2. Select the Racing page (Ref. Fig. 4).
3. Ensure the gate switch is closed and all lane sensors are un-obstructed.
4. Click on the 'Next Heat' button (Refer to Fig. 5).
    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.
    c. Car numbers and racer names are displayed for each lane according to the race plan.
5. Stage the cars on the track at the start line.
6. If any lanes are designated as a BYE lane a check-mark will appear in the appropriate Mask checkbox to designate it as such and "BYE Lane" will be displayed in the name box (See Fig. 10).
7. When ready, start the race by activating the gate release handle.
    a. The race status advances to the 'Racing' state.
8. Wait for all cars to cross the finish line or when 10 seconds has elapsed, whichever occurs first (Refer to Fig. 6).
    a. Race results are displayed and race status advances to the 'Finished' state
9. Repeat steps 4 through 8 till all heats have been ran.
10. At completion of the race the race results files will be generated. These files can be accessed via the Results page and viewed/printed by clicking on the appropriate button.

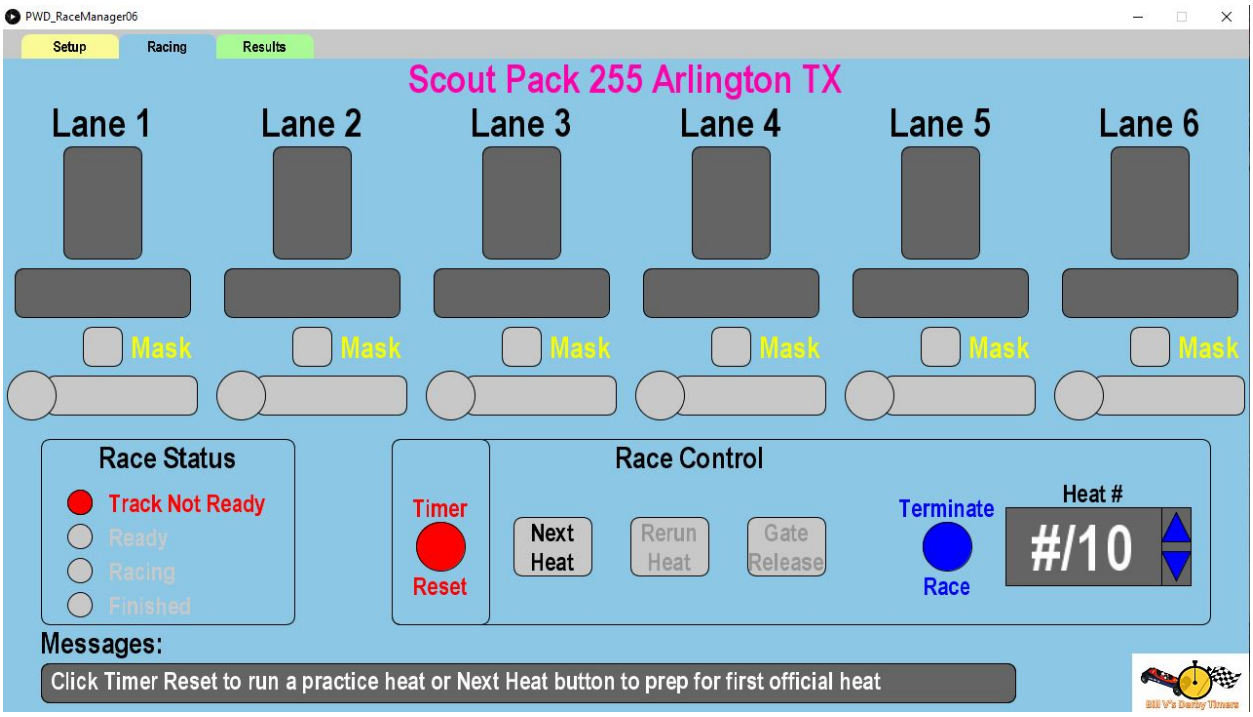


Figure 4 – Initial Timer Startup Screen

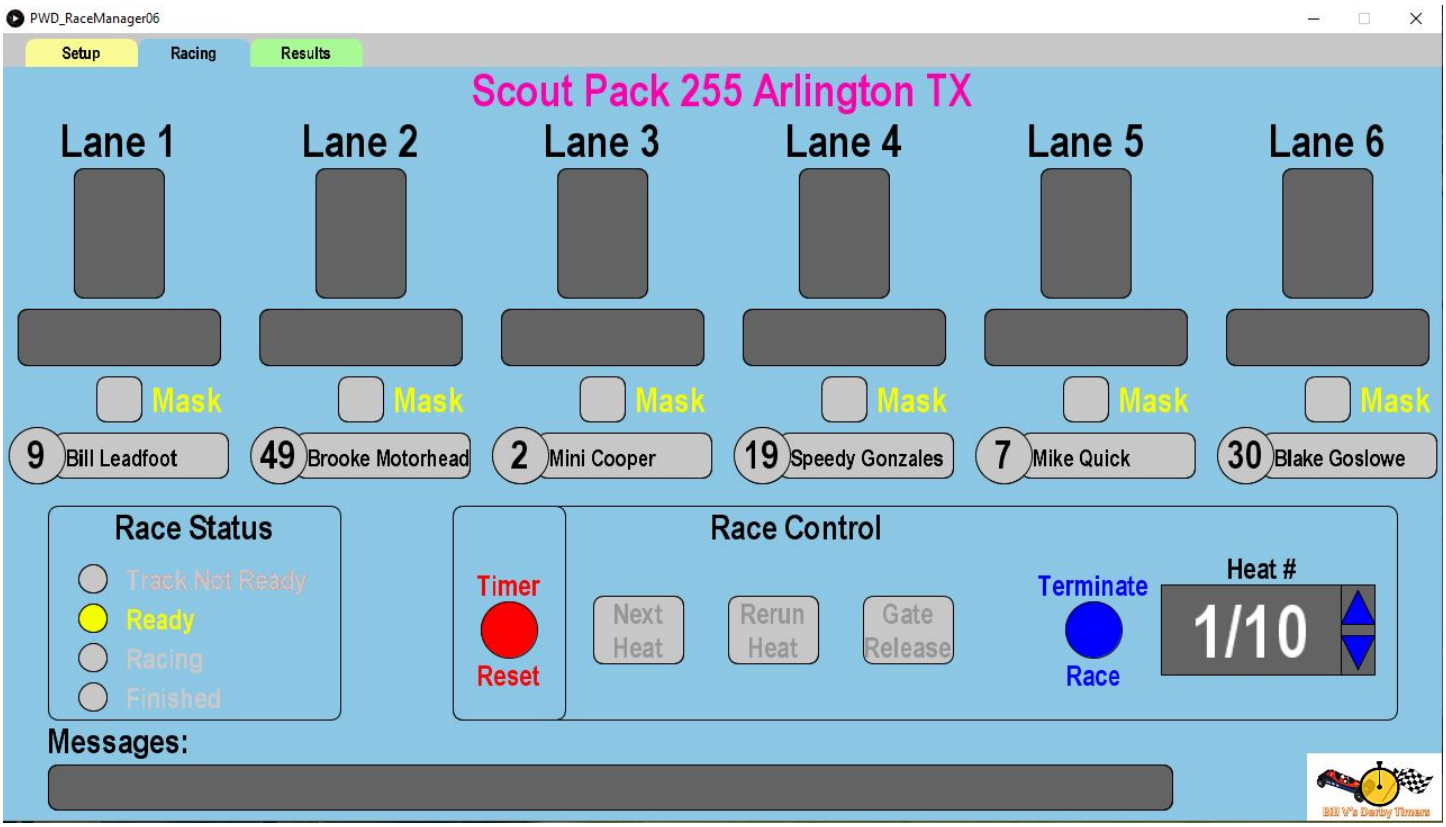


Figure 5 – Timer Ready Example

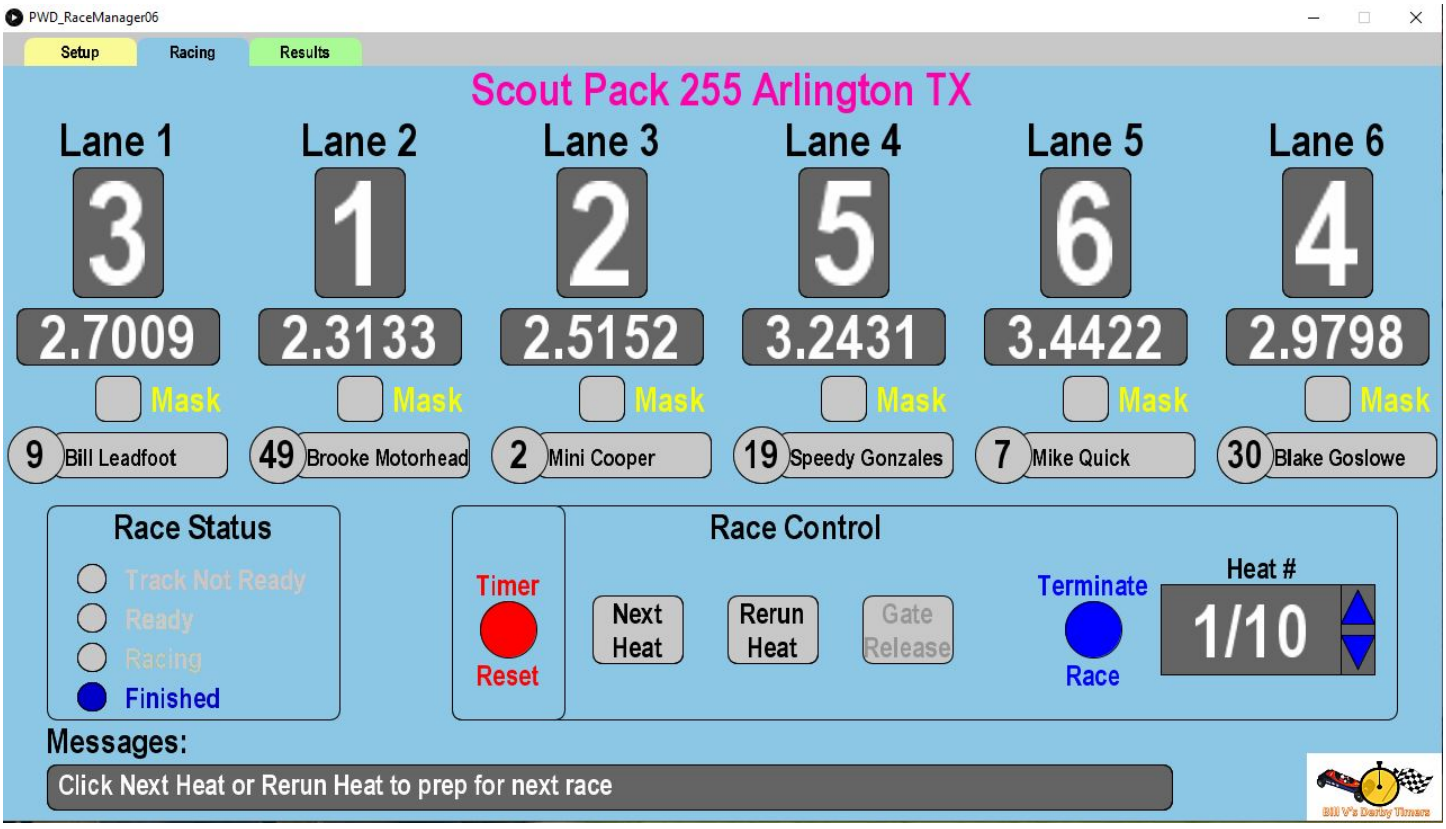


Figure 6 – Race Finished Example

3.2.5 Race Results

At the completion of the race (all heats ran) the race management software creates the overall standings report file from the data in the race results table (See Fig. 7). Race standings in the overall standings report is determined by the cumulative total time of all heats ran for each car, with the lowest total time being assigned 1'st place, second lowest time 2'nd place, etc. If the roster contains more than one race group, the software also generates a group standings report in which race standings (position) are parsed by group (See Fig. 8).

Position	CarNo	Driver	Group	TotalTime	HeatsRan	AvgTime	FastTime
1	49	Brooke Motorhead	Tigers	14.8032	6	2.4672	1.6761
2	68	Paul Jones	Wolves	15.1086	6	2.5181	1.4353
3	2	Mini Cooper	Tigers	19.8096	6	3.3016	1.5291
4	7	Mike Quick	Bears	19.8222	6	3.3037	1.5161
5	50	Harley Davidson	Bears	20.8374	6	3.4729	2.3243
6	5	Rick Jackson	Wolves	22.4472	6	3.7412	2.7567
7	19	Speedy Gonzales	Wolves	23.316	6	3.886	2.2262
8	9	Bill Leadfoot	Tigers	23.3496	6	3.8916	2.1091
9	1	John Nogas	Bears	26.6304	6	4.4384	2.3209
10	30	Blake Goslowe	Bears	27.165	6	4.5275	3.079

Figure 7 – Example Overall Race Standings Report

Position	CarNo	Driver	Group	TotalTime	HeatsRan	AvgTime	FastTime
1	7	Mike Quick	Bears	19.8222	6	3.3037	1.5161
2	50	Harley Davidson	Bears	20.8374	6	3.4729	2.3243
3	1	John Nogas	Bears	26.6304	6	4.4384	2.3209
4	30	Blake Goslowe	Bears	27.165	6	4.5275	3.079
1	49	Brooke Motorhead	Tigers	14.8032	6	2.4672	1.6761
2	2	Mini Cooper	Tigers	19.8096	6	3.3016	1.5291
3	9	Bill Leadfoot	Tigers	23.3496	6	3.8916	2.1091
1	68	Paul Jones	Wolves	15.1086	6	2.5181	1.4353
2	5	Rick Jackson	Wolves	22.4472	6	3.7412	2.7567
3	19	Speedy Gonzales	Wolves	23.316	6	3.886	2.2262

Figure 8 – Example Group Race Standings Report

Click on the Results tab to access the race results reports. Refer to Figure 9. Click on the appropriate button to launch your PC's default word processor or spreadsheet software from which you can view and print the document.

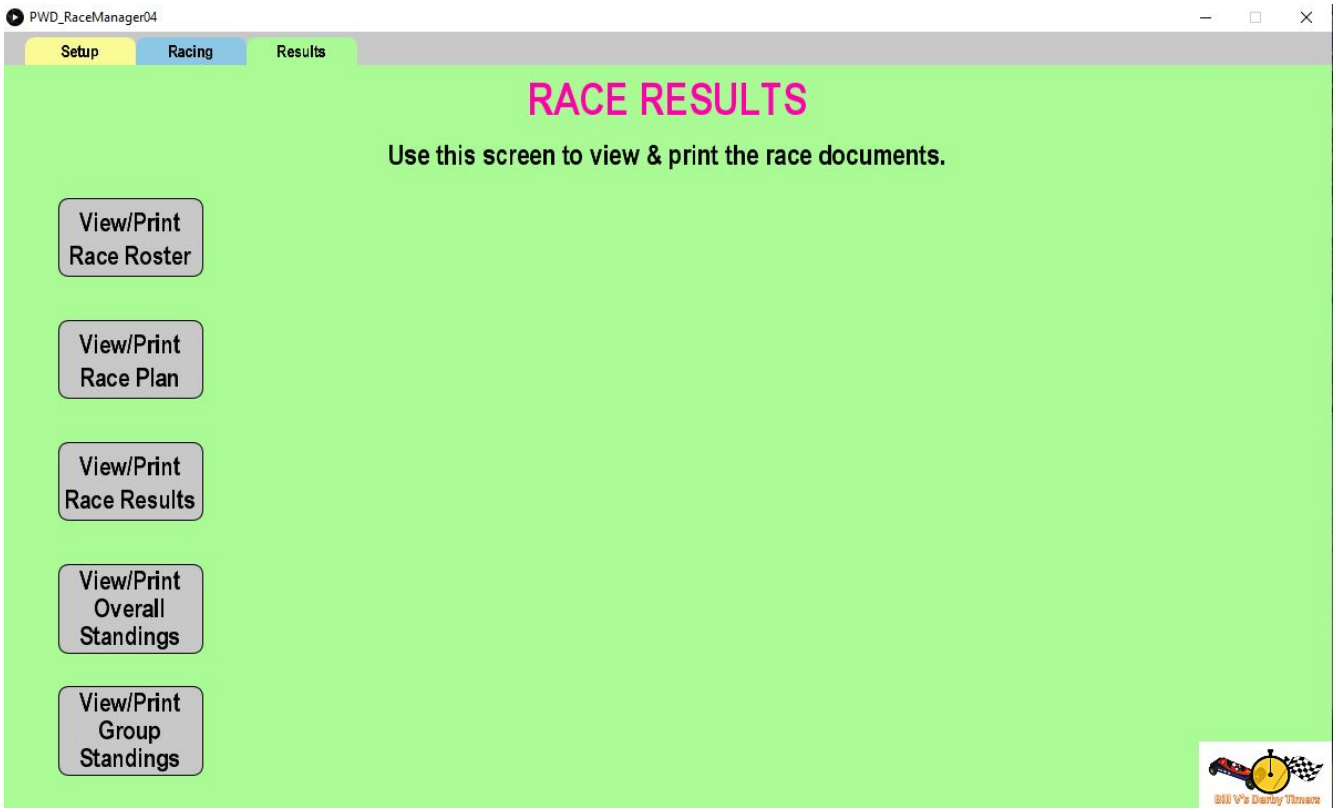


Figure 9 – Race Results Page for Accessing Race Result Reports

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 which is then displayed along with the associated car numbers and racer names assigned to their respective lanes for that heat. 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 maximum number of heats in the race. The maximum number of heats is determined in

the race plan. When stepping through the heats the car numbers and names assigned to each lane for that heat are displayed. When selecting a heat that has already been performed, race results (times & finish order) are also displayed. This feature is useful in cases where you may need to skip ahead to another heat or when a previously ran heat has to be re-run.

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.

3.3.6 Lane Mask Check Boxes

The 'Mask' checkboxes alternately display or clear a check-mark in the checkbox each time it is clicked. Manually checking a 'Mask' checkbox should only be done when running a practice heat in which less than 6 cars are racing. When checked, it sends the mask command for that lane to the Arduino timer. When unchecked it sends the unmask command for that lane to the Arduino timer. The Arduino timer will ignore any masked lanes (i.e. not wait for them to time out) while the race is in progress. Figure 10 shows an example of an automatically checked lane mask box when the automatically generated race plan includes a BYE lane designation in a heat.

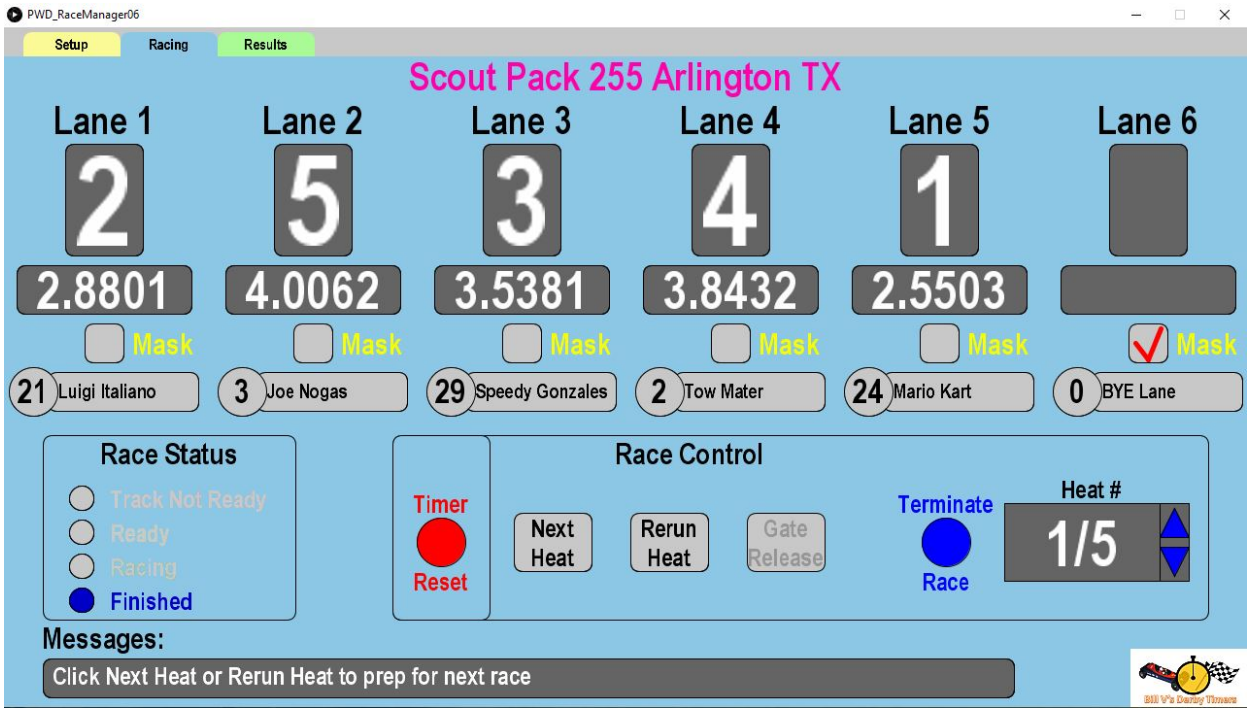


Figure 10 –Lane Mask Example

3.4 Additional Information – Files Created By the Software

The race management software will create four files consisting of a Race Plan, Race Results report, Overall Standings report and a Group Standings report. The Group Standings report is only generated if the roster has more than one group listed (See Fig. 2). The filenames for these reports are created at program startup and include a date/time value retrieved from the computer's clock.

The filename structure is as follows: "Race_Results_YYYYMMDDHHMM.csv"

Where: YYYY = 4 digit year, MM = 2 digit month, DD = 2 digit day, HH = 2 digit hour, and MM = 2 digit minutes.

This guarantees a unique file each time the program is started and prevents any previous files from being accidentally overwritten.

NOTE: By default these files are saved to the same folder containing the race roster. For example, if you have the selected race roster file saved in your 'Documents' folder, the Race Plan, Race Results report, Overall Standings report and the Group Standings report files will all be saved in the same folder.

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
M1 – M6	PC-to-Arduino	Lane Mask commands for lanes 1 thru 6. Instructs Arduino timer to set the corresponding mask flag to cause the timer to ignore that lane.
U	PC-to-Arduino	Lane Unmask command. Instructs Arduino timer to clear all mask flags.
C	PC-to-Arduino	COM Check command. Causes Arduino timer to send the '@' character back to the PC.
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 -or- 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 e.eeee f.ffff

where:

- a.aaaa = Lane 1 time
- b.bbbb = Lane 2 time
- c.cccc = Lane 3 time
- d.dddd = Lane 4 time
- e.eeee = Lane 5 time
- f.ffff = Lane 6 time

Troubleshooting: Use of the Arduino IDE serial monitor or another serial terminal program can be used to observe these commands or to send the reset or mask commands to the Arduino. 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. It does not test the race manager software that runs on your PC.

Setup:

- Bring up the Arduino Integrated Development Environment (IDE) on your PC (Refer to Section 2.1)
- (Optional) Select and load the PWD_ETTimer06_Ver5.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. 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. 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. Then on the serial monitor enter the letter R in the command line, press ENTER or click on the Send button.	N/A 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 each lane.	Same as step 3 except the lane number will correlate with the lane being obstructed. If the lane number does not correlate with the lane being obstructed, check the lane sensor wiring for a miswire, correct the problem and repeat the test.

Step	Action	Expected Results
5	Set the Gate Switch to the open position. On the serial monitor enter the letter R in the command line, press ENTER or click on the Send button.	N/A The message “GSW” followed by “NRD” is displayed on the monitor.
6	Set the Gate Switch back to the closed position.	N/A
7	Ensure 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 the following message is displayed: “Times: 9.9999 9.9999 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 occurred.	The moment the last lane sensor is tripped the following message is displayed on the monitor: “Times: x.xxxx x.xxxx x.xxxx x.xxxx x.xxxx x.xxxx” Followed by the message “FIN”, where x.xxxx is the recorded time for lanes 1 thru 6.
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 C in the command line, press ENTER or click on the Send button.	The message “@” is displayed on the monitor.
17	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	