Race-Log Vehicle Performance Tracking Software

By

Chad Pertner

Submitted to
The Faculty of the Information Engineering Technology Program
In Partial Fulfillment of the Requirements for
The Degree of Bachelor of Science
In Information Engineering Technology

University of Cincinnati
College of Applied Science

May 2001

Race-Log Vehicle Performance Tracking Software

By

Chad Pertner

Submitted to
The Faculty of the Information Engineering Technology Program
In Partial Fulfillment of the Requirements for
The Degree of Bachelor of Science
In Information Engineering Technology

University of Cincinnati
College of Applied Science

May 2001

_____ _____
Author: Chad S. Pertner Date

_____ _____
Faculty Project Advisor: Robert Schlemmer Date

_____ _____
Department Head: Lawrence G. Gilligan Date

Acknowledgements

Thanks to all those who have enlightened me so much
through my lifelong learning experience.

Table of Contents

List of Figures

Abstract

Race-Log Vehicle Performance Tracking Software by Chad Pertner

This database application provides an automated logbook for drag racing records. The database will keep track of results and the vehicle specifications that produced those results. In addition this application will contain utilities to execute calculations for common equations used by racers, supply a NHRA track directory with information on each location, and provide a maintenance reminder utility. The program was constructed using Visual Basic 6.0 and Access 2000. The reason this topic was chosen is because of my strong involvement in both Information Engineering and automotive racing technology and development

This project also served as partial fulfillment of my degree of bachelor of science in Information Engineering Technology, and was a valuable learning experience for myself. Time management was essential to the successful deployment of this project. Considerable amounts of time were spent learning new technologies through books and the Internet, along with brushing up on previously learned skills such as Visual Basic and Access. Recommendations for future changes would be a way to automatically populate the database. This currently must be manually entered by the user and is the most time consuming task. Ways of automatic data entry would possibly include a change to the timing system at the track to include bar coded information on the time slips that could be scanned into the race-log program. This project has proved itself to be a valuable learning experience on my part.

Race-Log Vehicle Performance Tracking Software

1. Statement of Need

The NHRA is the world's largest motorsports sanctioning body with more than 85,000 members, 144 member tracks, 32,000 licensed competitors, and nearly 4,000 member-track events. By far the most popular form of drag racing is a handicapped form of competition known as "E.T. Bracket Racing." In this form of racing, two vehicles of varying performance potentials can race on a potentially even basis. The *anticipated* elapsed times for each vehicle are compared, with the slower car receiving a head start equal to the difference of the two. With this system, virtually any two vehicles can be paired in a competitive drag race. In order to have the most consistent run, the conditions must be duplicated as closely as possible to previous runs. The racer chooses a “dial-in” for the car, or his *anticipated* elapsed time for the race. The vehicle that runs the closest to it’s dial-in will win the race. Determination of the dial-in is done by reviewing past performance using a logbook.

The variance in weather conditions throughout the day plays the biggest role in vehicle performance from one run to another. The first run may have been made in the morning when the air was cool and dense, making the car run a better time than later in the day. This makes it difficult to predict what the car will do throughout the season when the race day may be extremely humid and hot, and the car responds accordingly.

2. Review of the literature

The reason this topic was chosen is because of my strong involvement in both Information Engineering and automotive racing technology and development. The overall objective was to provide a software application that will assist the racer in organizing information, predicting vehicle performance and calculating race data. Research was done on existing programs, many of which I have previously used. It was found that existing programs either contained only small utilities, such as a performance calculator utility, or were more like games. After discussion with members of my local racing community, including Michael E. Braun, a veteran Pro-Stock driver (see figure 1), plans were made to create an application that does it all. The contents of such a program were discussed and the result is Race-Log.

Figure 1 – Pro Stock Drag Car

3. Product Description

Drag racers usually keep a logbook to record and track all the data from each race including track, weather, and tuning information. This data is used to predict the elapsed time for each drag race. Having records of everything from tire pressure to track elevation above sea level allows the racer to predict the elapsed time for the next race with greater accuracy. Because of the amount of data and the number of runs that are collected, it would be much easier to sort and compare data of this type by developing a database application allowing the racer to log and compare each run. The application also assists in predicting the vehicle's performance once enough data is collected to determine how much the performance is affected by the elements (mainly atmospheric conditions) involved in each run. When the current elements are entered, such as temperature and humidity, the application adjusts the performance and provides an accurate elapsed time prediction for the race. (See figure 2.)

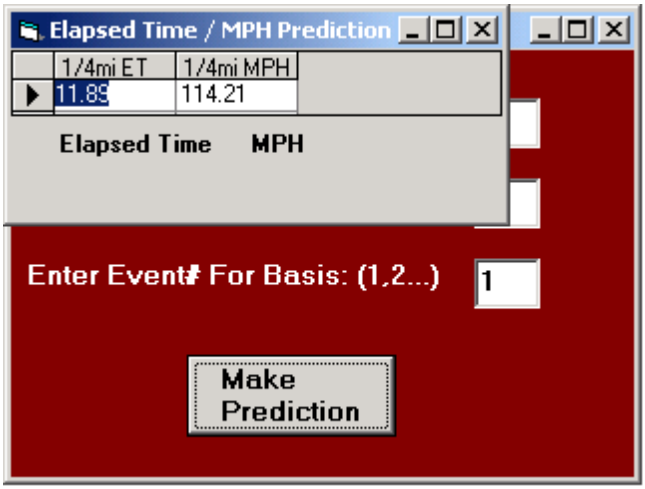


figure 2. Prediction utility example

This database application provides an automated logbook for drag racing records. The database keeps track of results and the vehicle specifications that produced those results. In addition this application contains utilities to perform calculations for common equations used by racers (see figure 3.), and provides an NHRA track directory with information on each location (see figure 4.). The racer will benefit greatly from an application that computes data regarding vehicle response to varying weather conditions, along with all the essential vehicle records from past races.

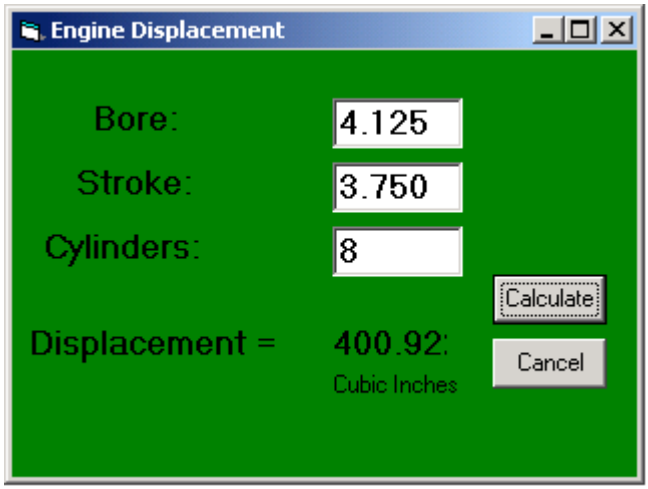


figure 3. – calculation utilities

figure 4. – NHRA track lookup

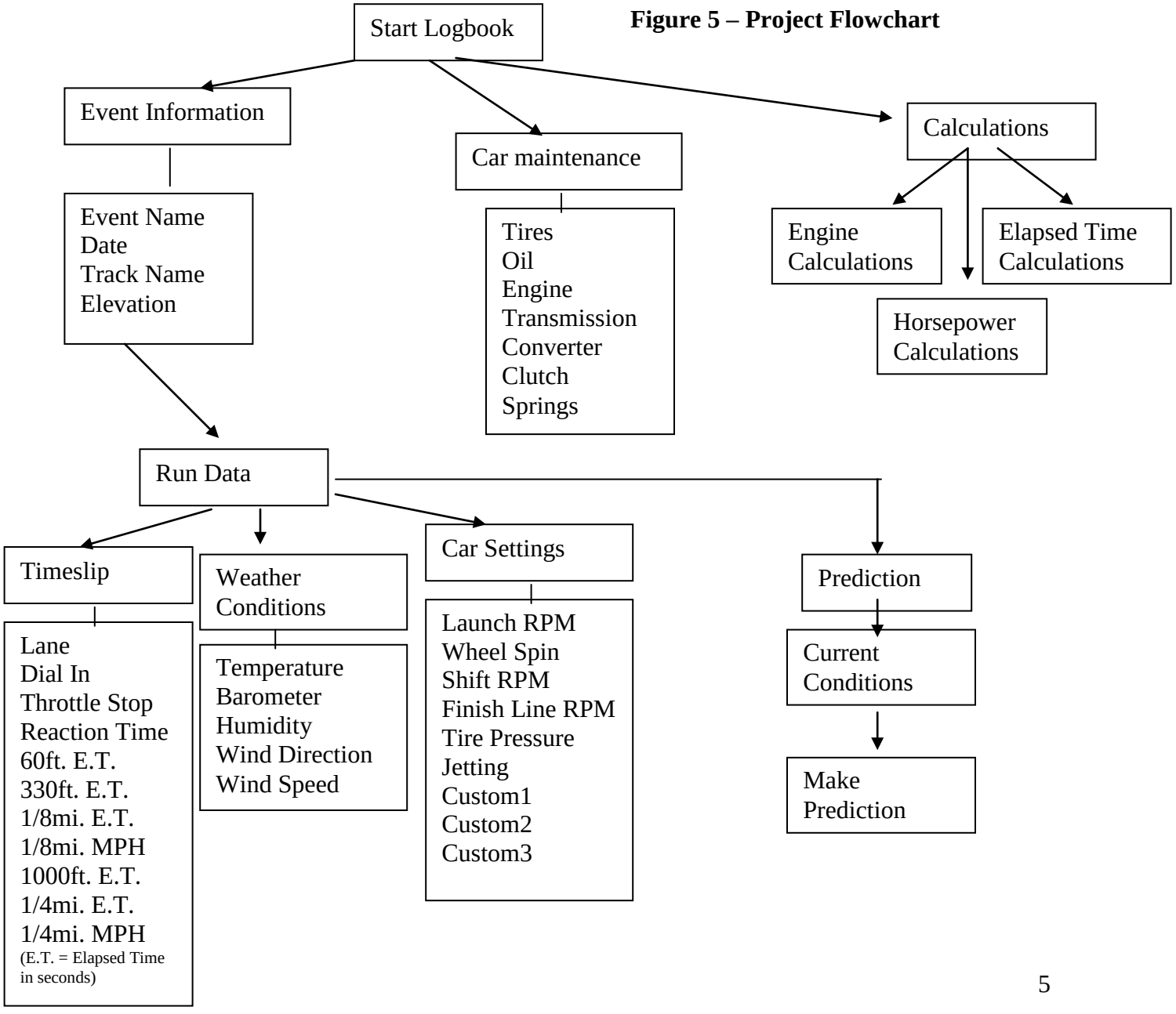
3.1 User Profile

The users of this application may or may not have any experience with personal computers. The logbook program does not require a high level of computer literacy to use, and basic installation instructions provide the necessary information to the user. When the program is launched, dialogue boxes contain clear and easy to navigate fields. Each field provides help topics in the form of pop-up messages that are accessed by pausing the mouse pointer over the field. No knowledge of database design is required since all of the data entry is contained within the application window.

3.2 Design protocols (see figure 5. for project flowchart)

When the application is first opened the user can select to enter race data, or use one of the many other racing utilities. In addition to logbook operations, the user may also choose at any point to enter the car maintenance application, or the vehicle calculations menu. After the information from at least two races is entered, a prediction can be made. The prediction gathers information about the current conditions, and provides the change in elapsed time based on past performance.

Figure 5 – Project Flowchart



The program uses common menu bars and command buttons that are used in most windows applications for navigation. The background colors are specific for each data entry box (white for time slip data, red for weather, yellow for car settings, green for car specifications, grey for calculations, and black for event information). Help topics are in the form of pop-up messages on each textbox. Pausing the mouse pointer over the control provides information about what type of data can be entered into the textbox.

4. Statement of Deliverables

In summary, the objective of this project was to design a comprehensive software application that will assist drag racers in keeping track of all the vehicle information, and provide a valuable ET prediction tool for use in bracket racing. Through the use of various programming capabilities the application now possess the following features:

- Comprehensive Database to log and track race information
- ET Prediction Tool
- Vehicle Maintenance Reminder
- Engine, Chassis, and Performance Calculators
- NHRA Track Directory

The overall objective is to provide a software application that will assist the racer in organizing information, predicting a vehicles performance, and calculating race data.

5. Development

5.1 Planning and Timeline

The project was constructed in four phases. Phase I included researching the basic components involved in logging information and prediction analysis. Analysis of what should be included in this type of software by interviewing members of the local racing community was also performed during this step. Phase II included creating useful tables and queries using information from Phase I, and creating the database relationships that were needed to accommodate the necessary data into the application. Phase III focused on creating VB forms that will enhance the usability of the application. This phase also involved error control and improving the overall look of the product. Phase IV focused on forming the working application and beta testing. This also included minor appearance changes, and tweaking of the database code to make the application more user friendly.

Time management was essential to the successful deployment of this project. Considerable amounts of time were spent learning new technologies through books and the Internet. Phase I was completed by the end of Senior Design I. Phase II was completed during the first half of Senior Design II. This may go beyond this period to research valuable technologies that will enhance the appeal of the product. Phase III was finished by the end of Senior Design II, leaving the final quarter to perform beta testing and debugging. This also provides me with enough time to make any changes in design and work on visual enhancements.

5.2 Budget

The expense of the project was minimal. The costs originate from the price of the software and needed literature to support the project. The following are estimated expenses that were incurred:

- **1299.00** for **Microsoft Visual Basic 6.0**.enterprise edition
- **339.00** for **Microsoft Access 2000** premium edition
- **$209.00** for **Microsoft Windows 98**
- **$800.00** for the **computer** (333Mhz Pentium II, 128M of ram, 12G hard drive)
- **$150.00** for **various media and other expenses** required for completion of the project.

2797.00 Total

5.3 Hardware and Software Requirements

This application was designed to run on a standard personal computer with minimal software requirements. As the database grows and the owners desire it, the application could be converted to accommodate multiple users and have more data capacity. This is an advantage of using Access with ADO coding since it is compatible with more robust environments such as SQL Server. The hardware and software requirements include:

- Windows 95
- Run-time version of the Visual Basic application
- Pentium processor or better
- At least 50 MB of hard disk space available for optimal performance
- 32 MB of RAM
- A good laser quality printer

6.0 Proof of Design

All programming techniques were learned in the three quarters of visual basic and access classes, along with personal experience. Many resources were used in the creation of this project, such as books from the library and online sources – see attached bibliography. The ideas for the creation and implementation of race-log were my own creation, along with input from friends in the form of brainstorming that resulted in a finely sculptured finished product. Many of the modules in the project such as the engine and performance calculators were just everyday formulas that were transformed into Visual Basic code. The access database simply stores all the race data and provides the ability to study the vehicles performance trends. Graphing utilities allow visual comparisons of all the races by use of an ADO (active data object) connection that allows the graphs to be updated while the program is in use. Race-log was intended as a racing tool that will provide utilities that are useful in drag racing. AN NHRA track lookup was included to provide information on all the tracks, such as the address and race times and dates. This simply pulls the requested data from the database using a select statement built into the Visual Basic code. The maintenance reminder is a useful tool that will keep track of how many runs are on each component of the car, this will allow the racer to prepare better for each race by replacing worn components before failure occurs.

7.0 Conclusions and Recommendations

The overall objective was to provide a software application that will assist the racer in organizing information, predicting a vehicles performance, and calculating race data. This project also served as partial fulfillment of my degree of bachelor of science in

Information Engineering Technology, and was a valuable learning experience for myself. Time management was essential to the successful deployment of this project. Considerable amounts of time were spent learning new technologies through books and the internet, along with brushing up on previously learned skills such as Visual Basic and Access. Recommendations for future changes would be a way to automatically populate the database. This currently must be manually entered by the user and is the most time consuming task. Ways of automatic data entry would possibly include a change to the timing system at the track to include bar coded information on the time slips that could be scanned into the race-log program.

Appendix A.
Database Relationships

Appendix B.
Visual Basic Code Examples (2)

1. Code for ADODB database connection to populate a table on a select case query for prediction analysis.

```
Dim WithEvents adoPrimaryRS As Recordset
Dim mbChangedByCode As Boolean
Dim mvBookMark As Variant
Dim mbEditFlag As Boolean
Dim mbAddNewFlag As Boolean
Dim mbDataChanged As Boolean

Public Sub Form_Load()
  Dim db As New ADODB.Connection
  Dim strSQL As String
  Dim adoPrimaryRS As New ADODB.Recordset
  db.Open "dsn=srdesign"
  strSQL = "select [1/4mi ET],[1/4mi MPH] from RunData2 where EventID = " &
frmPrediction1.txtEvent & " AND temp between " & frmPrediction2.txtHItemp & " and
" & frmPrediction2.txtLOtemp & " AND humidity between " & frmPrediction2.txtHH &
" and " & frmPrediction2.txtLH & ""
  adoPrimaryRS.Open strSQL, db, adOpenStatic, adLockOptimistic
  Set grdDataGrid.DataSource = adoPrimaryRS
  mbDataChanged = False
End Sub

Private Sub Form_Resize()
  On Error Resume Next
  'This will resize the grid when the form is resized
  grdDataGrid.Height = Me.ScaleHeight - 30 - picStatBox.Height
  lblStatus.Width = Me.Width - 1500
  cmdNext.Left = lblStatus.Width + 700
  cmdLast.Left = cmdNext.Left + 340
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
  If mbEditFlag Or mbAddNewFlag Then Exit Sub
  Select Case KeyCode
    Case vbKeyEnd
      cmdLast_Click
    Case vbKeyHome
      cmdFirst_Click
```

```
    Case vbKeyUp, vbKeyPageUp
      If Shift = vbCtrlMask Then
        cmdFirst_Click
      Else
        cmdPrevious_Click
      End If
    Case vbKeyDown, vbKeyPageDown
      If Shift = vbCtrlMask Then
        cmdLast_Click
      Else
        cmdNext_Click
      End If
  End Select
End Sub

Private Sub Form_Unload(Cancel As Integer)
  Screen.MousePointer = vbDefault
End Sub

Private Sub adoPrimaryRS_MoveComplete(ByVal adReason As
ADODB.EventReasonEnum, ByVal pError As ADODB.Error, adStatus As
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
  'This will display the current record position for this recordset
  lblStatus.Caption = "Record: " & CStr(adoPrimaryRS.AbsolutePosition)
End Sub

Private Sub adoPrimaryRS_WillChangeRecord(ByVal adReason As
ADODB.EventReasonEnum, ByVal cRecords As Long, adStatus As
ADODB.EventStatusEnum, ByVal pRecordset As ADODB.Recordset)
  'This is where you put validation code
  'This event gets called when the following actions occur
  Dim bCancel As Boolean

  Select Case adReason
  Case adRsnAddNew
  Case adRsnClose
  Case adRsnDelete
  Case adRsnFirstChange
  Case adRsnMove
  Case adRsnRequery
  Case adRsnResynch
  Case adRsnUndoAddNew
  Case adRsnUndoDelete
  Case adRsnUndoUpdate
  Case adRsnUpdate
  End Select
```

```
  If bCancel Then adStatus = adStatusCancel
End Sub

Private Sub cmdFirst_Click()
  On Error GoTo GoFirstError
  adoPrimaryRS.MoveFirst
  mbDataChanged = False
  Exit Sub

GoFirstError:
  MsgBox Err.Description
End Sub

Private Sub cmdLast_Click()
  On Error GoTo GoLastError
  adoPrimaryRS.MoveLast
  mbDataChanged = False
  Exit Sub

GoLastError:
  MsgBox Err.Description
End Sub

Private Sub cmdNext_Click()
  On Error GoTo GoNextError
  If Not adoPrimaryRS.EOF Then adoPrimaryRS.MoveNext
  If adoPrimaryRS.EOF And adoPrimaryRS.RecordCount > 0 Then
    Beep
     'moved off the end so go back
    adoPrimaryRS.MoveLast
  End If
  'show the current record
  mbDataChanged = False
  Exit Sub
GoNextError:
  MsgBox Err.Description
End Sub

Private Sub cmdPrevious_Click()
  On Error GoTo GoPrevError

  If Not adoPrimaryRS.BOF Then adoPrimaryRS.MovePrevious
  If adoPrimaryRS.BOF And adoPrimaryRS.RecordCount > 0 Then
    Beep
    'moved off the end so go back
```

```
    adoPrimaryRS.MoveFirst
  End If
  'show the current record
  mbDataChanged = False

  Exit Sub

GoPrevError:
  MsgBox Err.Description
End Sub

Private Sub SetButtons(bVal As Boolean)
  cmdNext.Enabled = bVal
  cmdFirst.Enabled = bVal
  cmdLast.Enabled = bVal
  cmdPrevious.Enabled = bVal
End Sub
```

2. Code to find closest matches to run data entered (Temp and Humidity)

```
Public Sub Command1_Click()
Dim Thi As Single
Dim Tlo As Single
Dim Hhi As Single
Dim Hlo As Single
Dim En As Single

Thi = (CSng(txtTemp.Text) + 6)
Tlo = (CSng(txtTemp.Text) - 6)
frmPrediction2.txtHItemp = Thi
frmPrediction2.txtLOtemp = Tlo
frmPrediction2.Visible = True

Hhi = (CSng(txtHumidity.Text) + 10)
Hlo = (CSng(txtHumidity.Text) - 10)
frmPrediction2.txtHH = Hhi
frmPrediction2.txtLH = Hlo
frmPrediction2.Visible = True

En = (CSng(frmPrediction1.txtEvent))
frmPrediction2.txtEvent = En

' frmRunData22.Visible = True
End Sub
```

Notes

A large part of RaceLog is a relational database. A relational database is like a big electronic filing cabinet with multiple drawers called tables. Each table has many folders called records. Each record has fields of data that is specific to its own table. In this program, the ‘Runs’ table stores all the information pertinent to vehicle performance, while the ‘NHRA Tracks’ table stores the records for track names and addresses.
Racers use weather computers to predict ET, determine the proper air/fuel ratio, and to analyze vehicle performance. When used properly, the computer can help to achieve the utmost in consistency. Following are a few hints and tips to help get the most out of this racing software.

Air/Fuel ratio management

The air/fuel ratio for a gasoline-burning engine must be about 12.5 to 1 for the car to respond properly to changes in atmospheric conditions. There is no benefit to running a lean mixture.

Find a day when the air density is average for the racing season in your area, then richen the fuel mixture until the car slows down. You shouldn't have to re-jet the carburetor until the air density changes by 4 percent or more. By adjusting the fuel mixture for an average air density of 96 percent, for example, the air can range for 92 to 100 percent without having to re-jet.

A quick way to check for the proper fuel mixture is to inspect the car's header collectors. The color inside them should be very dark gray or black. If the color is white or light gray, the mixture is way too lean. As the air density changes over the racing season, you will notice a change in color, lighter as the air density increases and darker as it decreases.

Analyzing Performance

After adjusting the fuel mixture properly, the car should run on the predicted ET consistently. If it doesn't, variables are affecting the combination. A computer will help determine these variables so that they can be eliminated. Begin by correcting every ET to Standard Pressure (STP, also known as sea level conditions) and recording each run in a racer's log. In correcting the ET, we remove the effect that the current atmospheric conditions have on the run. Therefore, the STP ET is the elapsed time the car would have run under perfect weather conditions. This creates a basis to compare on ET against another regardless of the weather conditions. Once all variables have been removed, the STP ET should be almost identical for every run.

ET Prediction

It's a good idea when predicting dial-in to also predict the 60 foot and the 1,000 foot times. As eliminations progress, comparing the predicted 60-foot time to the actual 60-foot time will show exactly what is happening to the traction factor. Moreover, if a racer hits the brakes to avoid breaking out, he or she will be able to tell exactly what the car would have run by comparing the predicted 1,000-foot time to the actual 1,000-foot time.

By comparing the STP ET's from every run, the user can find out exactly how consistent his or her car is. If the STP ET's vary by .020, then the racer may want to dial .010 or more under the predicted ET to avoid breaking out. This is especially true for the car being chased. This driver needs to have the confidence to run it out the back door if necessary.

The horsepower output of every engine is affected by the pressure, temperature, and humidity of the inlet air. Therefore, in order to provide a common basis of comparison, it is necessary to apply a correction factor to account for the difference between standard air (STP) conditions and the actual conditions current at the time and location of measurement. The Society of Automotive Engineers adopted a standard method to be used for correcting observed power at full load. This method is used as a basis for determining maximum horsepower levels with a dynamometer.

You cannot base ET prediction on horsepower output alone. Since the performance level of every car is different, the effect of the air will vary accordingly from one car to another. Two cars with identical horsepower, but having different power/weight ratios and other variables will respond in different degrees to the same change in weather conditions. Therefore, an accurate method of ET prediction may be based on an established performance level of the vehicle. Race-log uses such a system, consisting of a Base Line Horsepower Correction Factor (**HCF**) and a Base Line ET. These two factors are determined by using a record of past performance. This record can often be as simple

as the data from the last run, if the car responds perfectly to varying weather conditions, or an average from a group of runs if it doesn't. This Base Line ET is then 'corrected' to what it should be, based on the current weather conditions given and essentially becomes the ET prediction.

Air-Fuel

On gasoline burning engines, the air/fuel ratio, must be about 12.5 to 1 in order for the computer to accurately predict performance. A lean running engine will not respond properly, perhaps making less horsepower when the computer says otherwise. Generally, if the air gets better and the car slows down, the engine is too lean. However, if the car picks up too much for the current air, it is probably too rich. An engine running a slightly rich fuel mixture is perfect for ET prediction based on air changes.

How close the actual ET is to the predicted ET depends on any other variables that relate to the consistency of the car's performance.

Cars seem to be a few hundredths slow on the first run of the day. One cause may be due to power train and chassis lubricants that have not yet reached proper operating temperatures. Running the engine and drive train with the car on jacks is part of the solution, however, this does nothing for the heavy grease used in some front wheel bearings. Surface rust on exposed brake rotors and drums may have some effect, especially if your car sits outside during the week. If your car behaves in this fashion, later on by examining your racer's log, you may be able to ascertain just how long the car has to sit before this factor comes back into play. But, first you must know its happening

before you can do anything about it. Anyway, don't be surprised if your first run of the day gets thrown out of our computation. Try not to make the runs too close together. Automatic transmission fluid will heat up and lose viscosity resulting in a drop in line pressure and decreased converter efficiency. These are just some of the variables to watch out for. When you start to analyze the data you are collecting, more ideas will surface than you could imagine. The first step is collecting the data and computing a Horsepower Correction Factor (**HCF**) for every pass you make.

The **HCF** is the correction factor defined by the current weather conditions for the exact time when that particular run was made. For example, on the first time run an example car ran a 9.352 ET with a 60' time of 1.348 under weather conditions of barometer: 29.82, temperature: 73, and relative humidity: 56. After entering these conditions into your computer, it will report the current Horsepower Correction Factor as 1.0374, Vapor Pressure as .458, and the Air Density as 95.736. This is the 'air' for your first pass. If you were going to use this one run as your base line performance, your BASE LINE ET would be 9.352, your BASE LINE 60 FT would be 1.348, and your BASE HP FACTOR would be 1.0374. The following are terms used in the project:

Elevation - Range: -3,000 to +26,000 Feet - Elevation is only used if you are using an altimeter or a corrected barometer. If you're using an uncorrected barometer reading absolute pressure, always enter 0 for the elevation. If you have an altimeter, calibrate it to 29.92 In Hg and enter the reported altitude as the elevation. If your area requires that you use a corrected barometer, then you should always enter the number of feet equal to the amount of correction when you calibrated the barometer.

Barometer - Range: 12 to 35 Inches Hg - If you're using a barometer to obtain the air pressure, enter the barometric pressure from the gauge. When using an altimeter, you must always enter 29.92 for the barometric pressure.

Temperature - Range: 10 to 130 degrees F - Enter the temperature from a digital thermometer or other temperature gauge. Never use the temperature readings reported on your time card. These sensors are not accurate! From round to round, temperature changes are apt to affect performance more than any other variable.

Humidity - Range: 0 to 100 Per Cent - You can use a digital hygrometer to find the relative humidity. At low temperature, relative humidity has the least effect on performance. However, at high temperatures, the effect is much greater. For example, 50% RH at 60 F will result in a .01 loss in power, while at 90 F the addition of 50% RH equals about a .03 loss in power. Changes in temperature are also more critical under high humidity.

Vehicle Weight - The Vehicle Weight (including driver) and **Ballast** is used in the computation for finding the Horsepower Corrected Weight.

The following is more information on the resulting calculations:

Current HCF (Current HP Correction Factor) - The Horsepower Correction Factor is computed using the current weather conditions that you enter. In other words, the HCF equals the current elevation, barometer, temperature, and humidity. When the HCF is higher, less horsepower is produced, and the car will run slower. So, the lower the HCF is, the faster the car will run. You can use the HCF to determine the variance in horsepower or torque that is being produced under different weather conditions. For

example, if your engine produces 400 horsepower with an HCF of 1.012, it will produce 404.80 horsepower at sea level conditions (STP) or 1.0000 (400 x 1.012=404.80).

Air Density (Air Density Index) - The Air Density Index (ADI) is the computed ratio of the current density to that of STP with STP having an index of 100 percent. This factor is used to determine the correct jetting or pill size required under varying weather conditions. The higher the ADI, the more air there is available for combustion. Therefore, more fuel must be burned to maintain the correct air/fuel ratio. When using gasoline, a four (4) percent change in the ADI will usually necessitate a change in jet area in order to maintain the same level of performance. Holley jet sizes are about 4 percent apart in flow area. If the ADI goes up four (4) percent, for example 100.0 to 104.0, increase all jet sizes by one number. And, if the ADI goes down four (4) percent, for example 100.0 to 96.0, decrease all jet sizes by one number.

Density Alt (Density Altitude) – Density Altitude (DA) is the current air density expressed in feet of elevation. DA is also known as the corrected or relative altitude.

Effects of Weather from Dragnbreath.com (4)

Air Density and Performance

(Courtesy: Roger Copeland) (5)

SAE Testing

The Standard Day (SAE test standard J1349):

Temperature: 60 deg. F

Humidity: 0%

100% Air Density: 29.92 inches (barometric)

Barometer

Barometer readings have a linear effect on air density. If the barometer goes up or down 1% from the standard reading, so will air density.

Example:

30.22 (New) ÷ 29.92 (Std.Day) = 1.01 or 101%

29.62 (New) ÷ 29.92 (Std.Day) = .989 or 98.9% (Round up to 99%)

A barometer of 30.22" gets you to 101% air density, 29.62" nets to 99%, etc.

Each 5 deg. F change in temperature (away from 60 deg. F), you get about a 1% change in air density. An 80 deg. F day means 96% air density, and a 40 deg. F day means 104% air density.

Temperature goes up... The air gets thinner. Temperature goes down...Air gets thickerer. (Air Density) This is not linear, but it's close enough for racing considerations.

Ever notice how much better an engine runs on a cool damp day?

At 60 degrees, jumping from zero to 50% humidity will cost you roughly 1% in air density. That same 50% humidity at 90 degrees will cost you more than 2% in air density. The reason for this is that it

requires much more water vapor to get you to 50% humidity at 90 degrees than it does at 60 degrees, since air can hold more water as it's heated.

Relative Humidity

"Relative humidity" is expressed as a percentage of water in the air compared to how much it can hold at a given temperature.

All 3 figures need to be considered. A 50 degree day with 50% humidity and a 29.62" barometer nets you to just about 100% air density, since the 2% you pick up in temperature is offset by the losses from the barometer drop of 1% and the humidity loss of 1%.

Power

Figure around 75-80% power change compared to the air density change. An 8% drop in air density will cost you about 6% in power.

Barometric changes give you about a 1.2:1 change in power.

Temperature changes give you about .7:1 change in power.

Humidity is about 1-1 change in power.

The effect on quarter-mile speeds.

Horsepower = (Car Weight with Driver) X (Quarter-Mile-Speed over 232.3)

cubed.

HP = Multiply Car Weight by

$$\frac{(\text{quarter mile speed})\ 3}{232.3}$$

Assume a car (with driver) is 3450 pounds:

96 ÷ 232.3 = .4132587 (where 96 is the 1/4 mile speed)

.4132587 X .4132587 = .1707827

.1707827 X .4132587 = .070577

.0705774 X 3450 (Lb.) = 243.49 HP

Formulas and Equations used in Drag Racing

- Conversion 1 Liter = 61.02 Cubic Inches
- HorsePower = TORQUE * RPM / 5252
- HorsePower from MPH (1/4 mi.) = (.00426 * MPH)^3 * WEIGHT
- HorsePower from MPH (1/4 mi.) = WEIGHT * (MPH/234)^3
- HorsePower from ET (1/4 mi.) = WEIGHT * (5.825/ET)^3
- ET from HorsePower (1/4 mi.) = (WEIGHT / HP)^.333 * 5.825
- TORQUE = (5252 * HP) / RPM
- CFM = (CID/2)*(RPM/1728)*VE
- VE = volumetric efficiency
- VE is approximately equal to 0.78 for stock engines, 0.85 for hi-perf street engine
- and approx. equal to 0.97+ for full race engines.
  (This gives rather low values for CFM but is probably all you really need.)
- Tire height (inches) = {[(section width * profile) / 2540] + (wheel diameter / 2)} * 2
- 60 foot time from ET = (.12 * 1/4 mile et) + .17
- MPH = Tire Radius / 168 * RPM / Trans Gear * Rear Gear
- RPM = 168 * Trans Gear * Rear Gear * MPH / Tire Radius
- GEAR RATIO = Tire Radius * RPM / 168 / MPH
- Tire Radius = 168 * MPH * Trans Gear * Rear Gear / RPM
- CID = Number Of Cylinders * Bore * Bore * Stroke
- Volumetric Efficiency = Actual Engine Air Intake (CFM) / CID
- Compression Ratio = Cyl. Volume @ BDC / Cyl. Volume @ TDC = 1 + (Swept Volume / Volume @ TDC) = 1 + (0.7854 * Bore * Bore * Stroke) / (CCV + HGV / PDV)
- CCV = Combustion Chamber Volume (in inches) = CC's / 16.4 HGV = Head Gasket Volume (in inches) = Compressed
- thickness * 0.7854 * Bore * Bore PDV = Piston Deck Volume = Volume of Piston Dish + Volume of Piston Dome
- HorsePower = Atmos. Press. * CR * VE * CID * RPM / 5252 / 150.8

References

1. NHRA. “National Hot Rod Association Web Page”. http://www.nhra.com**.** 1995-2001.
2. Beard, Michael G. “The Staging Light Web Page”. http://www.staginglight.com. 1996-2000.
3. SAE. “Society of Automotive Engineers Web Page”. http://www.sae.org**.**
4. Cascade Graphics, LLC. “Drag’nBreath for serious racers Web Page”. http://www.dragnbreath.com/**.** 1997.
5. “Visual Basic 6 Database Programming”. John Connell, Wiox Press, September 1998