

Microchip/Circuit Cellar 2007 design contest

## Universal Automotive Racing Performance Monitor

Real time performance data for your car

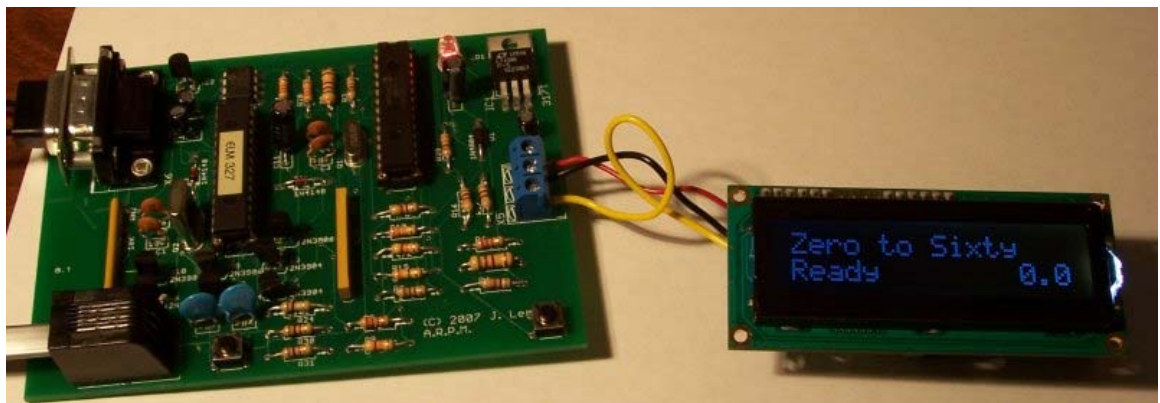
**Registration Number: MT2294**

For hobbyists interested in automotive performance, accurate collection of vehicle performance data has always been a challenge. Traditionally, measuring of data such as 0 to 60 times has required expensive RADAR equipment, and accessing data such as “boost” has required modifications be made to access the information. With the growth of electronic control in modern automobiles, a number of pre-installed sensors are available to record this performance data. The question is how to present this data in human readable format. Luckily for consumers, all of the data we need is available on the government mandated On Board Diagnostics II (OBD-II) port.

The goal of this project entry is to take the data available on the OBD-II port, rationalize it, and make the following information available to the driver:

- 0 to 60 timer
- Real time Miles per Gallon (MPG) fuel economy
- 1/4 and 1/8 Mile timers
- Real time “Boost” (Intake Manifold Pressure)

The “brain” of this project is a Microchip dsPIC 30F3013 16-bit digital signal processor. The dsPIC 30F3013 was chosen for its floating point support and the availability of 2 UARTs.



### Functional overview of the device

The Universal Automotive Racing Performance Monitor (UARPM) consists of a single device which plugs into the OBD-II jack of an automobile. The monitor itself consists of two main components, a main board which houses all of the logic and data acquisition hardware, as well as two user push button switches, and an LCD screen to display information.

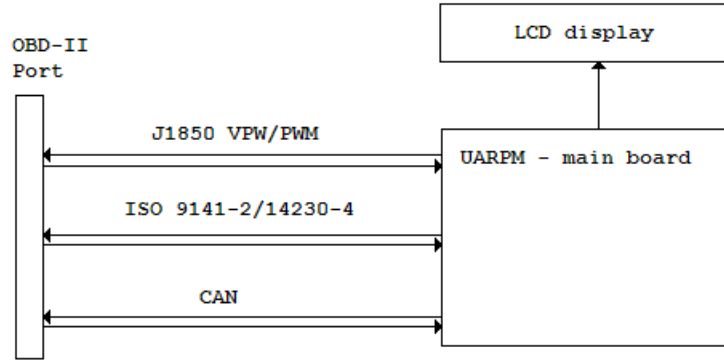


Figure 1: Top level block diagram.

### Operation of the device

To install the UARPM, the user simply plugs in the device to the OBD-II jack of their car. The unregulated 12 V supply pins of the OBD-II are always active, so the device will power up immediately upon being plugged in. The automobile will not send data via OBD-II (the “bus”) unless the key is in the RUN or START state, so the displays will remain in the default state until the device begins receiving data.

The device has two push button switches for user control. The right button is for toggling between the 5 display modes, and the left button is held down to display the best recorded data for the current active screen. The operation of each individual screen is given below

### Zero to Sixty Timer



Figure 2: Zero to Sixty display.

The Zero to Sixty Timer works as follows:

1. When the device sees a vehicle speed of 0 mph, the timer displays the time of the last run, and signifies the ready state by displaying the text “Ready:”
2. When the car begins moving, the “Ready:” text disappears, and the timer begins incrementing in 100ms intervals.
3. Once the car reaches 60 Mph, the device stops counting. The resultant time is the time required to go from a standing start to 60 mph.

- 4. If the run is the fastest time on record, the recorded time is stored as the best 0 to 60 time. The best zero to sixty time can be accessed by holding the left push button down.

### Real Time Fuel Consumption (MPG)



Figure 3: MPG display.

The real time fuel consumption gauge is a non-interactive display which displays a calculated Miles per gallon rating. The formula for this calculation is given in the calculations section of this document. The refresh rate is 100 ms.

### 1/4 and 1/8 Mile Times



Figure 4: 1/4 Mile Time display.

The 1/4 and 1/8 mile timers while two distinct screens, operate in a similar manner:

- When the vehicle is at a stop, the device is in the "armed" state, signifying the device is ready to start recording.
- As soon as the vehicle begins moving, the time portion of the display begins incrementing every 100 ms, and the current vehicle speed is displayed. Once the vehicle has traveled 1/4 or 1/8 mile, both portions of the display stop incrementing.
- The displayed values are the time taken to drive 1/4 or 1/8 mile, and the vehicle speed reached at the end of the run.



Figure 5: 1/8 Mile Time display.

### Boost



Figure 6: Boost display.

The Boost gauge displays the real time absolute value of the intake manifold pressure. This is a display of the intake manifold pressure converted to PSI (the original units of the bus message is KPA). The refresh rate of the display is 100 ms.

## Technical Design of Project

### I. Hardware

#### Main control

The microcontroller used to do the performance calculations is the Microchip dsPIC 30f3013. This microcontroller has IEEE floating point support, which was required for fast calculation of miles traveled from the vehicle speed messages. The other criteria for the microcontroller is the availability of 2 UARTs, which to 30F313 provides. 2 UARTS are needed because both the LCD and the ELM 327 OBD-II transceiver communicate via RS232 at TLL voltage levels.

#### OBD-II transceiver

Due to the desire to create a universal monitor, the ability to communicate over the various flavors of OBD-II buses was a requirement. The ELM Electronics ELM 327 fit this criteria, as it is able to auto-negotiate and communicate via CAN, J1850 VPW/PWM, and ISO 9141-2/14230-4. The ELM 327 is controlled via UART serial connection. To the main microcontroller, the OBD-II bus is a master/slave architecture, where the ELM 327 responds to requests from the dsPIC. The electronics connecting the ELM 327 to the OBD-II port are based on the reference design provided in the ELM 327 datasheet.

#### LCD display

The LCD display chosen is a Spark Fun Electronics white on black 2x16 character display with backlight. This LCD is controlled via RS232 serial communications, simplifying the software design of the project. The backlighting of the display, as well as the great contrast provided by the white on black color scheme, make the display easy to read in the harsh lighting conditions of an automotive cabin.

### II. Software

The software running on the dsPIC has been divided up into 2 sections: a driver layer, which handles the reading data from and sending data to the peripherals, and an application layer, which makes decisions on what data to request and what data to display. The application layer is modeled outside of the microcontroller first, then C source code is auto generated from this model to be compiled with the C driver layer to make a single flashable software module. The structure of the driver/application relationship is shown in figure 7.

## Driver Layer

The driver layer of the software was written in ANSI C for Microchip's C30 compiler. The main functional components of the driver layer are as follows:

### 2 16-bit timers:

Timer1: handles periodic requests of OBD-II bus information.

Timer2: 20ms timer which updates timers used for various time based calculations.

### 2 UARTs:

UART1: connected to LCD, operates at 9600 bps. One way communication.

UART2: connected to ELM 327, operates at 36800 bps. Two way communication.

Code to call the model portion of the software:

The procedure to run the model is:

- A. update input API of model
- B. Run the Model
- C. Handle the output API of the model

## Model Layer

The model layer of the software was designed and tested with a Mathworks Simulink model. The logic of the model is implemented in a Stateflow diagram, decomposing all processing into a collection of state charts. Modeling the core functionality in a Stateflow diagram allows rapid testing of the behavior before it is flashed onto the microcontroller. Once confidence has been achieved in the model, it can be autocoded into C code, and compiled along with the supporting driver layer with Microchips C30 C compiler.

While Simulink's C code generator does not explicitly support the Microchip dsPIC line of processors, it does have a generic 16-bit target which works well with the dsPIC. The auto generated C code is fully compliant with the C30 compiler.

### Calculations:

The formula for the Mile per Gallons is  $MPG=(7107*veh\_speed)/MAF$ , where:

veh\_speed is the vehicles speed in KM per hour

MAF is the Mass Air Flow Rate

MPG is the target Miles per Gallon x 10

The source of this formula is Bruce Lightner's paper available at:

<http://www.circuitcellar.com/advertise/cc-advertising/Lightner-183.pdf>

Complete source code, schematics, block diagrams, and a collection of photos are available in the relevant sub directories.

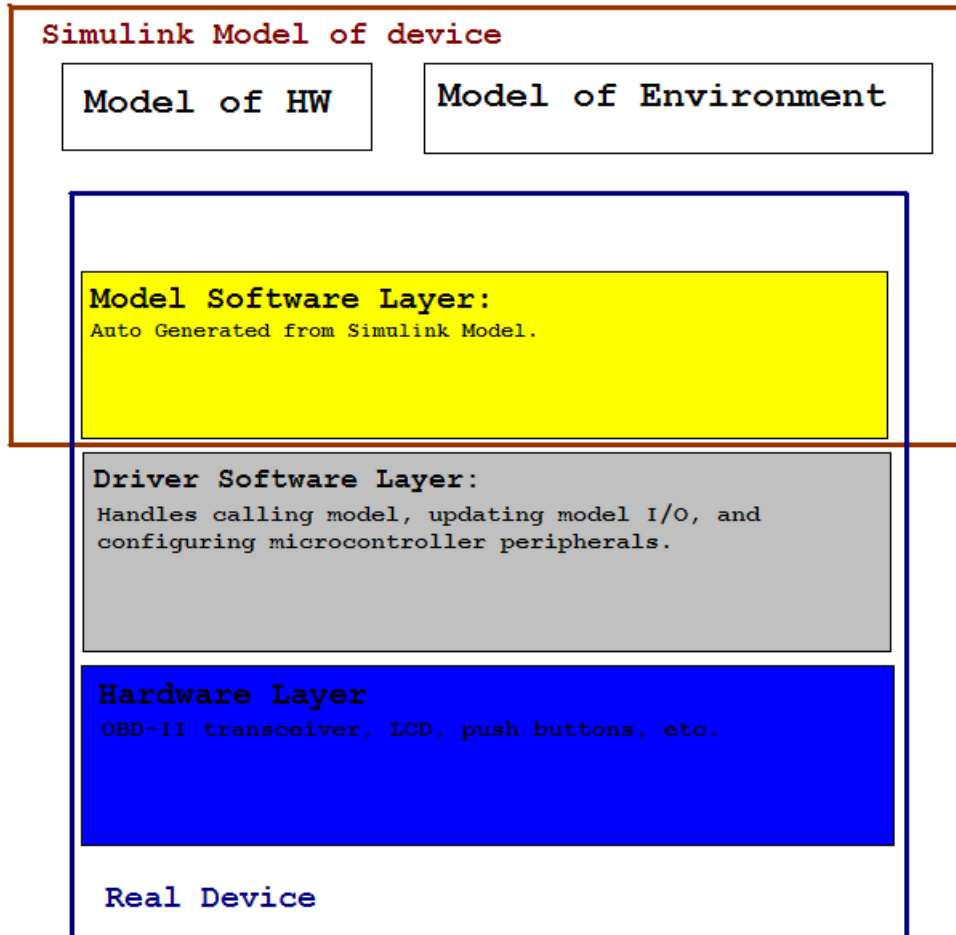


Figure 7: Relation of model SW layer, driver SW layer, and real hardware. The model software layer can be tested both virtually and on the real device.

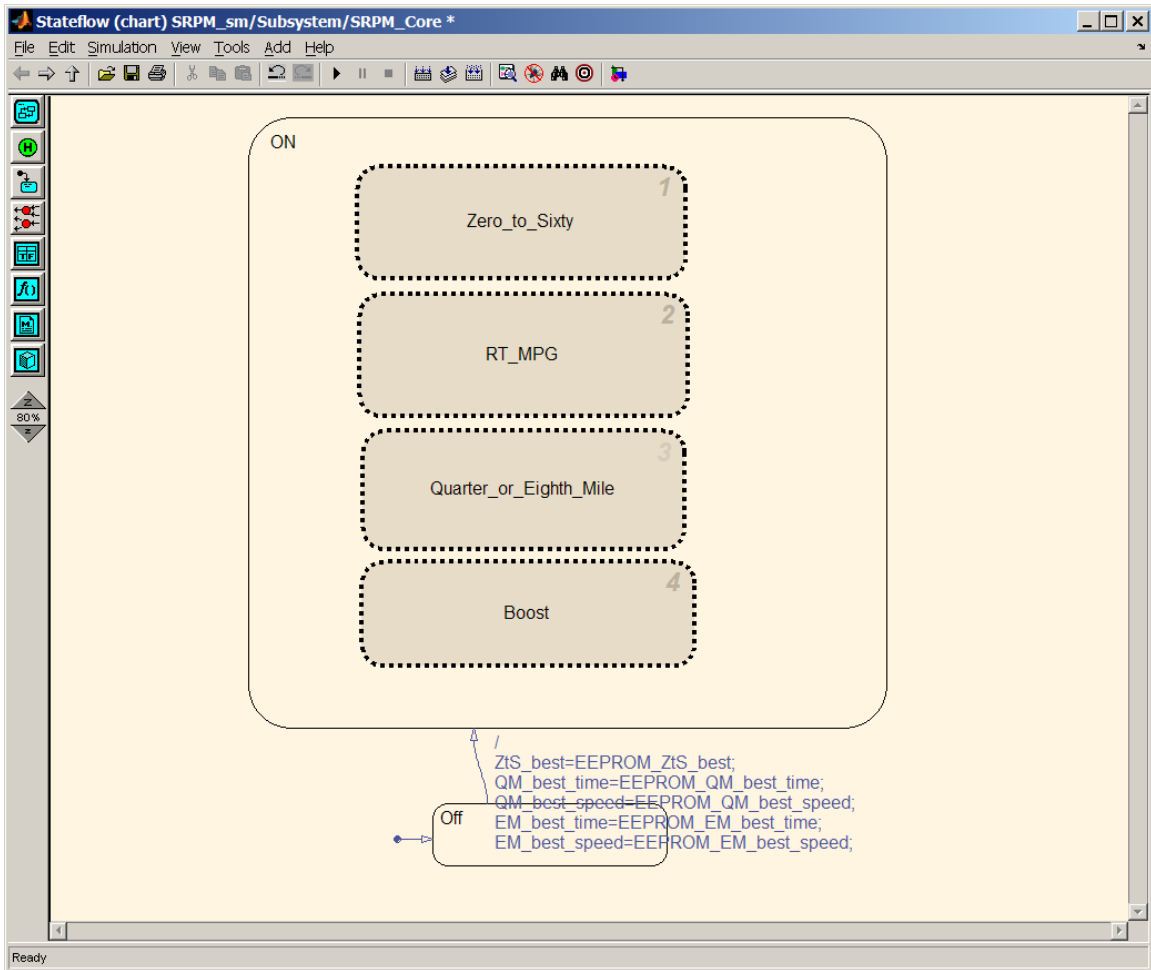


Figure 8: Top level of Simulink Stateflow Model

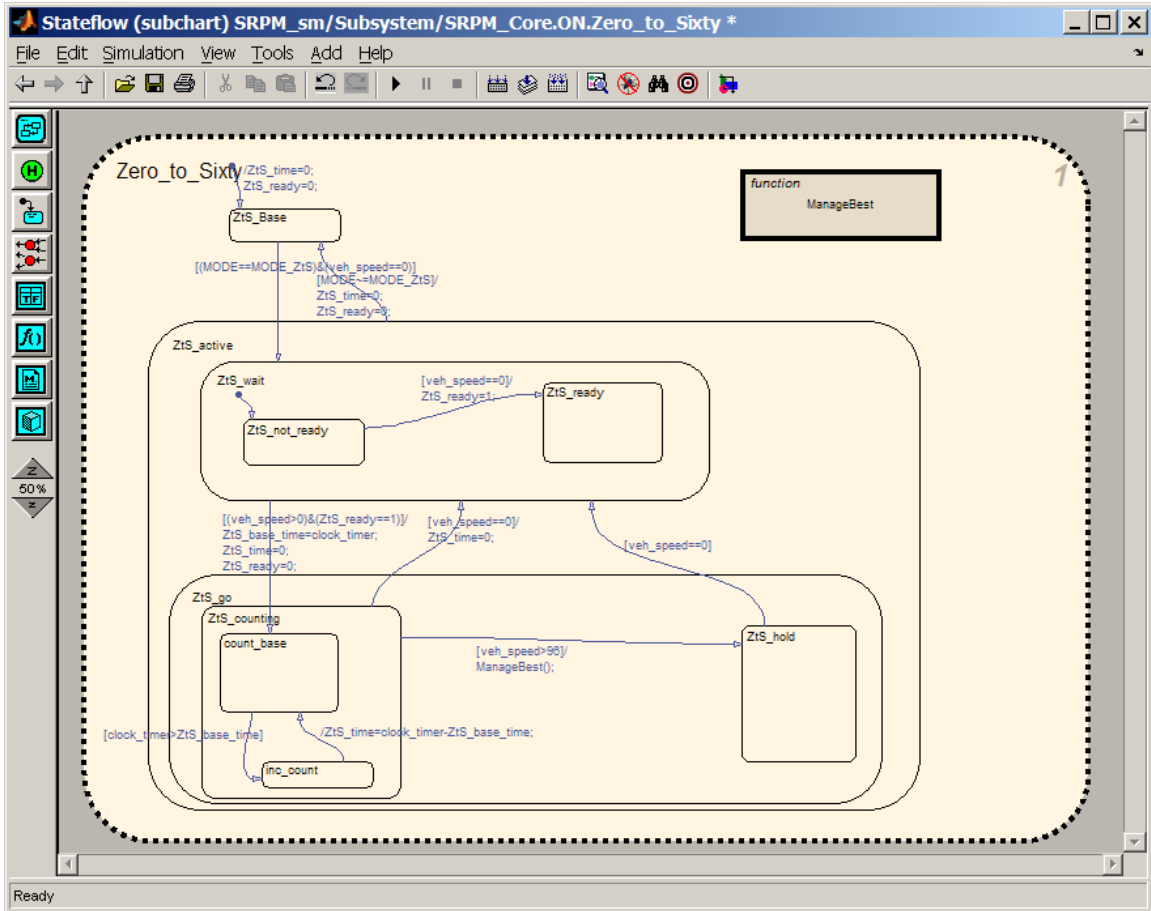


Figure 9: Example Subchart – Zero to Sixty Timer

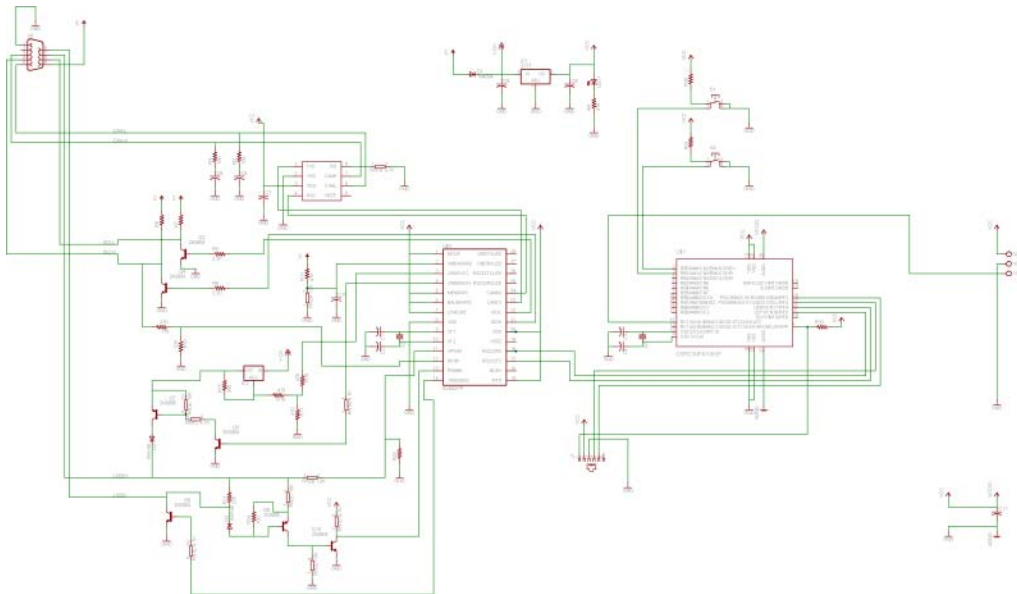


Figure 10: device schematic