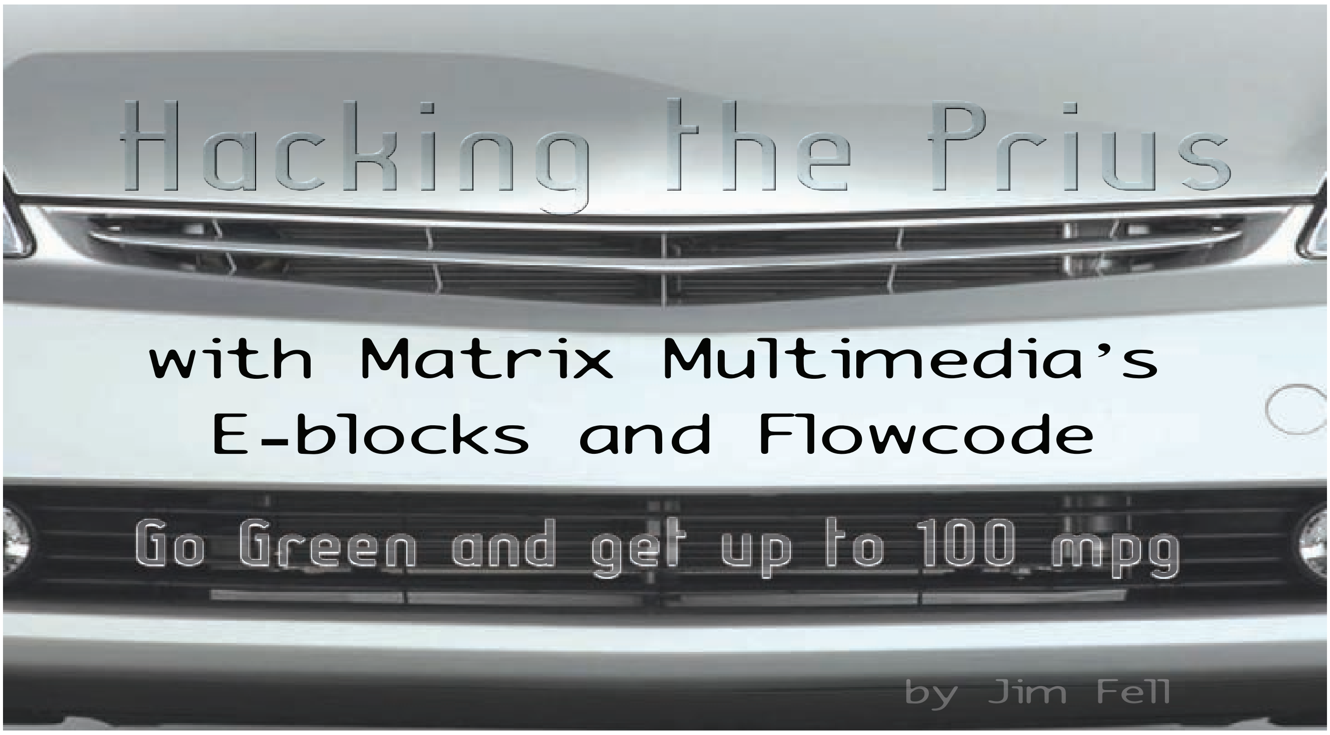


Hacking the Prius

with Matrix Multimedia's E-blocks and Flowcode

Go Green and get up to 100 mpg

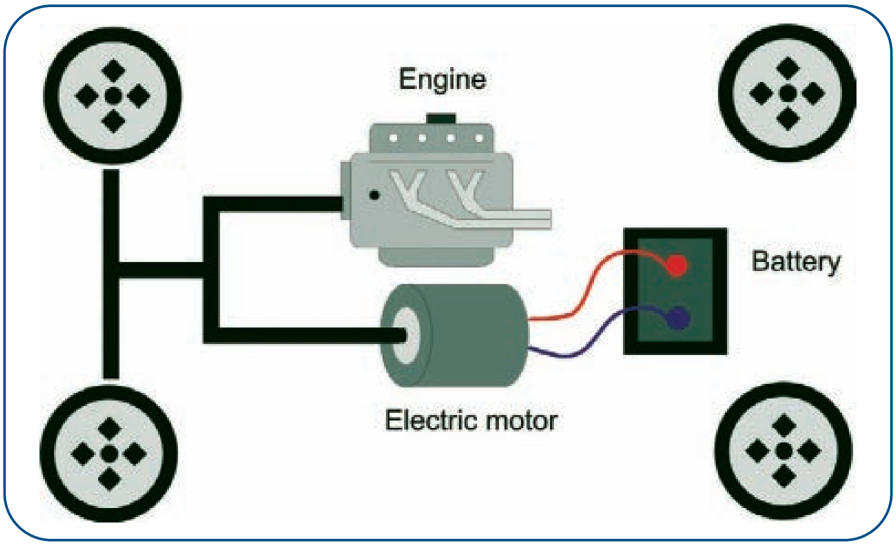
by Jim Fell

I first converted a car to purely electric operation in 1999 and after several improvements — particularly to the battery pack — the car was moderately successful. I was generally able to travel about 50 miles on a charge and considerably more if care was taken. The car completed the London to Brighton Electric Vehicle (EV) Run in 2005 and 2006.

Unfortunately, the achilles heel of the EV is still the battery pack. With low cost, traditionally lead-acid batteries the range is severely limited and a long, cross-country run must be planned like a military campaign. There must be charging points every 50 miles or so, and you need to stop for a couple of hours at each to restore some charge.

In 2005, I started looking at the hybrid cars that were available and the Toyota Prius in particular. The interesting thing about the Prius was that it could run for a limited period as an EV, however with the NiMH battery pack the electric motor can take the car only about one mile at less than 31 mph.

■ FIGURE 1. The basic topology of the Prius drive system.



I wanted to reduce the fuel consumption of the Prius from 60 mpg to 100 mpg — a massive cost savings — by the addition of a large Li-Ion pack. This article described how I achieved this using E-blocks and Flowcode from Matrix Multimedia as a control system.

How It Works

Figure 1 shows how the Toyota Prius works. Essentially, it is a normal car with the addition of an electric motor/generator in the drive train. When the driver needs to slow down, the brake pedal puts the 'motor' into generator mode which charges the battery up. Conversely, at low speeds the motor is used to assist the conventional petrol engine which decreases fuel consumption.

When I started the project, a few groups in the US were experimenting with supplementary battery packs to increase the range of the Prius. The Toyota — along with most modern cars — has a very complex electronic control system. The part that deals with drive and battery management uses CANbus for communication. The operation of the drive among other things is based on the State Of Charge (SOC) of the battery pack. If the SOC is low, the management system will recharge when descending a hill, braking, or use any surplus energy from the engine. If the SOC is high, then the battery pack will be used to drive the car at low speed or to supplement the engine when driving, climbing hills, or overtaking. In practice, the SOC is moving about the entire time, dependent upon traffic and driving patterns.

I saw two main problems in adding a large battery pack in parallel with the existing battery. The first was what would the reaction be from the Toyota management system if — out of the blue — the existing battery started receiving charge from an outside source — the second battery! The second problem was how to control this external charging source.

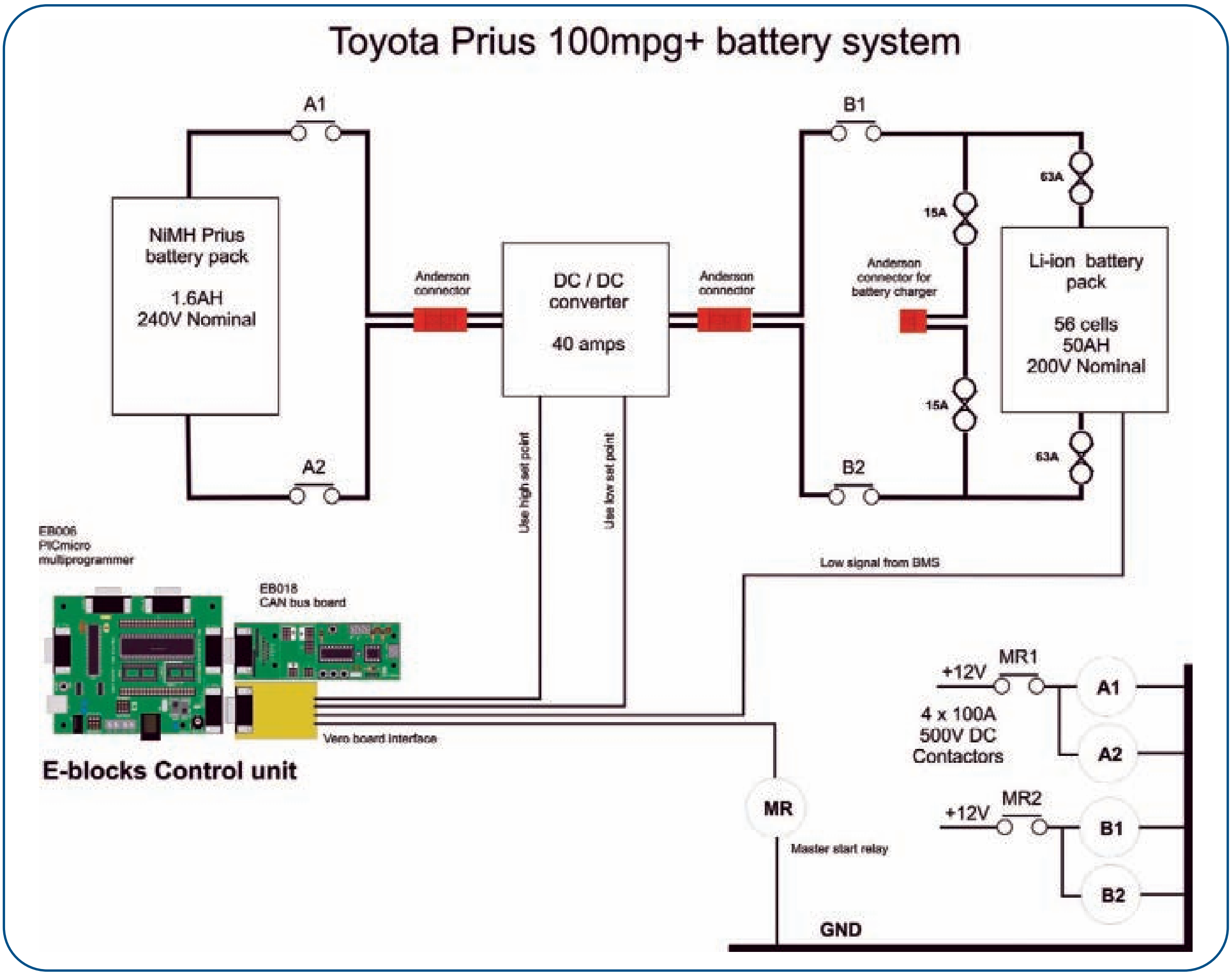
The control system needed to be such that the existing batteries' SOC could be manipulated so that the Toyota management system saw a high SOC and used the battery instead of the engine wherever possible.

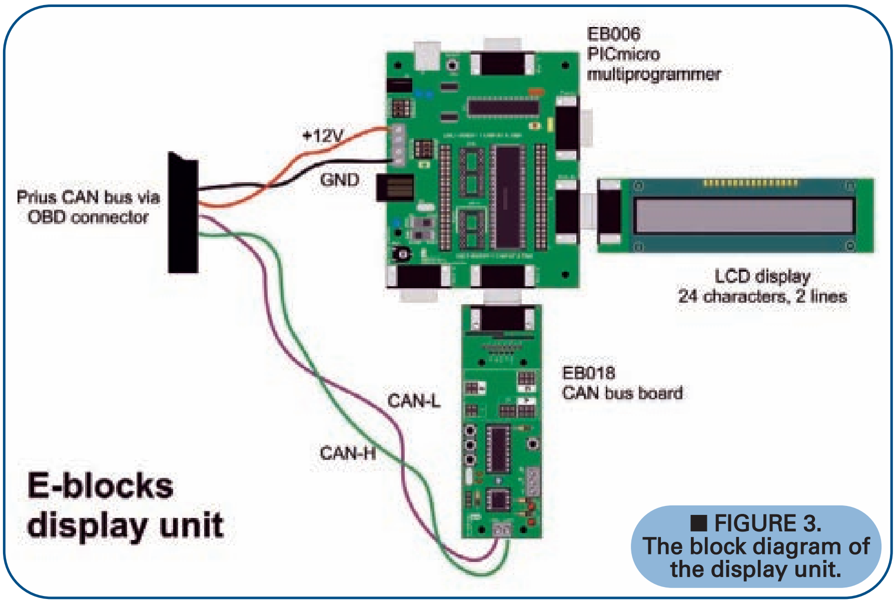
The first problem was simple. I connected my EV charger across the Toyota battery pack and charged the pack. The SOC increased up to fully charged (about 80% SOC). The battery manager took into account the pack temperature and voltage, and computed the SOC quite happily. Solving the second problem — transferring energy to the Toyota's battery — was the main area of work.

Circuit Details

I was lucky enough to have acquired a set of 56 Thunder Sky Li-Ion cells which I could use as a second battery. These are connected in series to give a resulting DC voltage of around 210V and more than 50 Ah. The Toyota's NiMh battery produces around 240V DC, so I knew that I would need an inverter to allow the additional battery pack to charge the Toyota's own battery. I also wanted to be able to recharge the Li-Ion batteries overnight, so I needed a recharge circuit. I needed a circuit to control the flow of charge into the Prius' own battery, as well. You can see the circuit in Figure 2.

■ FIGURE 2. Block schematic of the system.





■ FIGURE 3. The block diagram of the display unit.

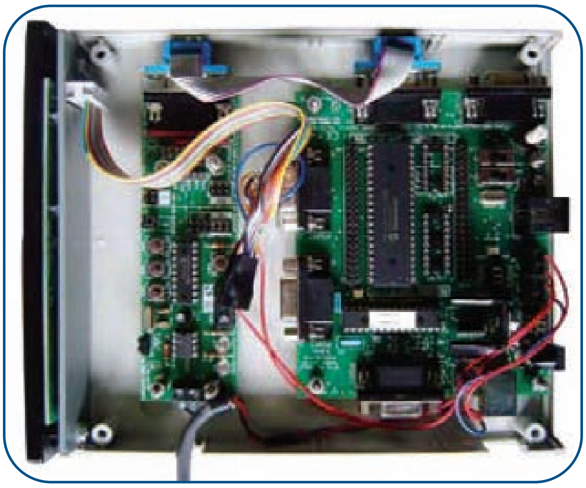
The means of connecting the extra battery pack to the existing pack was by using four single pole, high voltage power contactors and a high power DC-DC converter.

The converter is actually a battery charger which has a bridge rectifier as the first component to convert the normal AC mains input to DC. Of course, you can just feed it with DC. The DC-DC onboard converter is used to charge the Li-Ions if required, but that's another story.

The converter had a two-stage, selectable output. In high, the converter would try to lift the existing pack to a high voltage and thus a high SOC. In low, this voltage was lower and allowed the existing pack to lose charge letting the SOC% to fall back. The output of the DC-DC converter is controlled by switching in one of two sets of points. When the battery is being charged overnight, it is isolated from the Toyota circuit by a second set of points.

The NiMh to Li-Ion battery contactors would be energized the whole time during vehicle operation, until the extra battery pack was fully discharged and no longer able to contribute — at which time the batteries were disconnected.

Controlling this system meant hacking into the Toyota CANbus system. The car has many devices on the CANbus network and fortunately they all broadcast their data onto the bus. The devices that need the data read it and react accordingly. As far as I am aware, no device solicits information from another device (*CANbus is arranged as one master and N slaves -Ed*). What was needed was a custom CANbus device that could read parameters on the system and move charge into the existing battery pack at the right time.



■ FIGURE 4. The display unit with multiprogrammer, CAN board, and LCD on the front.

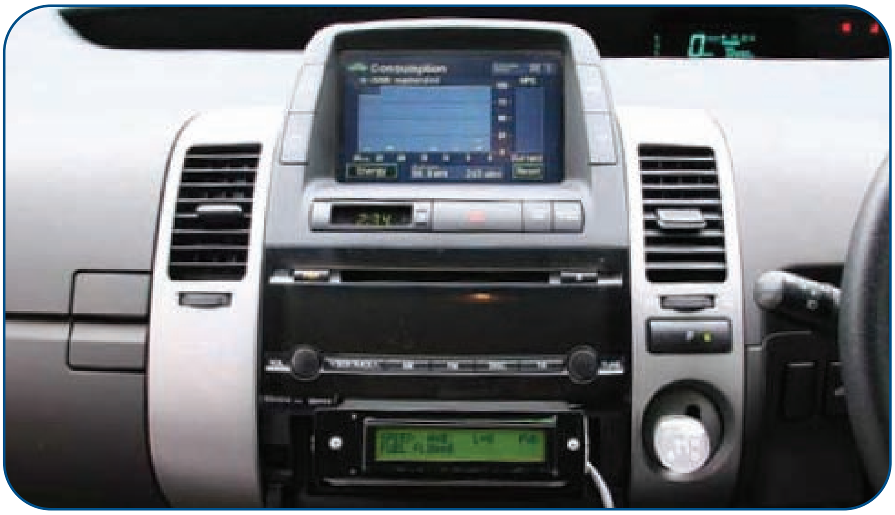
At this point, I read an article in *Elektor* on Flowcode (February '06); this referred to a CANbus system consisting of two nodes of a network. From past experience with other bus systems, it can take a long time to get a system up and running. I have a bit of experience with Microchip PIC devices and there is a wealth of information on their website on CAN networking. The datasheet on the CAN interface chip (MCP2515) runs to 81 pages.

I ordered the Flowcode CAN system and saw immediately all the hard work of using the CANbus had been done already. Setting up the parameters for the bus and reading specific messages is carried out by prewritten macro commands. Getting the communication between two points was very straightforward.

In order to read the messages emitted containing data on the SOC, a Kvaser Light CAN to USB unit was used to look at the bus traffic. There is a convenient OBD-II connector with 12V power located just under the steering wheel in the Prius. There is some documentation regarding the messages on the bus on the Internet. The format of the data varies and a bit of manipulation is needed to convert the data to a form which can be displayed on an LCD.

With some idea of what I wanted initially from the bus, I set up a system in the workshop mimicking the function of the CAN bus in the Prius. One of the E-block systems continuously transmitted an SOC message in the same format as the Toyota message; the other system was set up as a display unit which showed the system parameters on an LCD display. This was used in the development and commissioning phases of the project on the bench, and fitted into the radio compartment of the car as shown in Figure 5. The display shows the Battery Current, Battery Voltage (charging/discharging), State of

■ FIGURE 5. The display unit in the Prius dashboard.



Charge %, Charge Current Limit, Discharge Current Limit, Max Battery Temperature, and Min Battery Temperature. In this way, the whole system could be built up and tested away from the car.

The second stage of the program used only one of the items (SOC%) and gave out one of two outputs — high or low — depending on the value of SOC. In order to maintain the existing battery SOC at around 70%, a simple pair of decision instructions in Flowcode are put on the low output if SOC% >70 (and disconnected the Li-Ion cells from the charge circuit) and put on the high output if SOC%<65 (which switched the Li-Ion cells into the circuit and charged the NiMh Prius battery). In each case, the opposite output would be turned off.

One additional output was used to drive a relay which, in turn, energized the four main contactors. This output would come on five seconds after the system powered up and would go off in response to the additional battery pack becoming discharged.

There was no need for a display on the final controller and this now lives in an enclosure in the boot (*or trunk for us Yanks -Ed*) next to the extra batteries and power contactors.

The additional battery pack is a set of 56 Thunder Sky Li-Ion cells. These cells are about two years old and vary in capacity; the worst being about 50 Ah at 20 degrees C when discharged at 25 amps. The worst cell defines the pack capacity, so with the current limit set to 25 amps the car will run for two to three hours in assist mode until the battery pack switches off. The car then runs in normal hybrid mode as before.

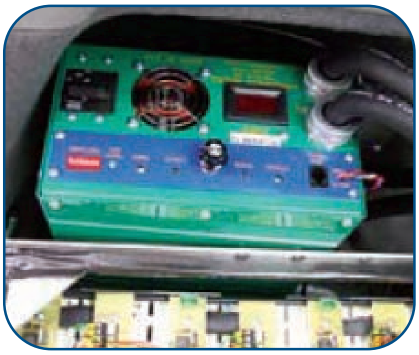
One drawback of the system is these batteries are still very expensive and physically quite large. Another is that the batteries take up some of the boot space as you can see in Figure 6.

Figure 8 shows the Flowcode program. Flowcode allows users to develop a program by dragging standard flow chart icons onto the workspace and then clicking on them to enter the properties. Users can simulate this on screen, and when the program is debugged can compile the flow chart to machine code which can be downloaded to a PIC microcontroller. Flowcode includes built-in routines and dialogs for CANbus communication which allows you to easily transmit and receive packets of CAN data.

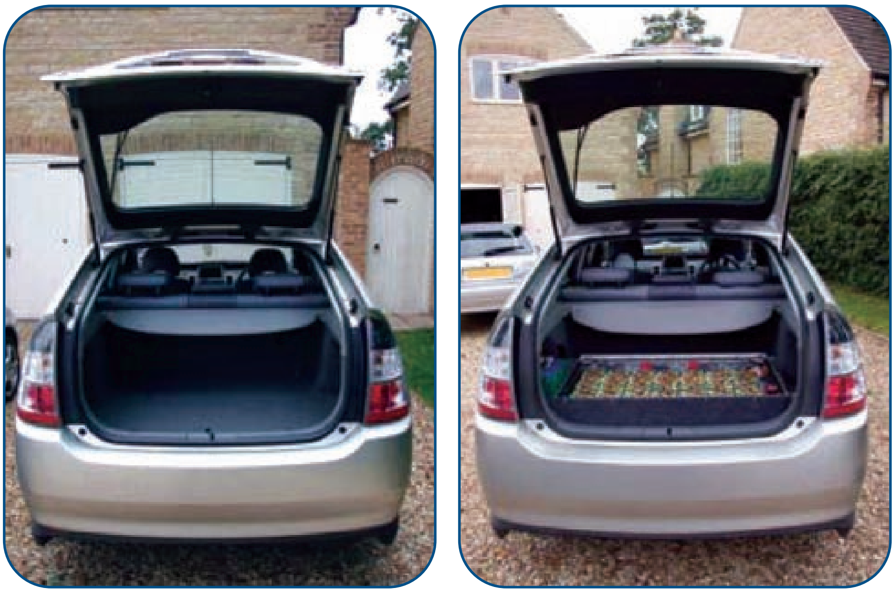
Conclusion

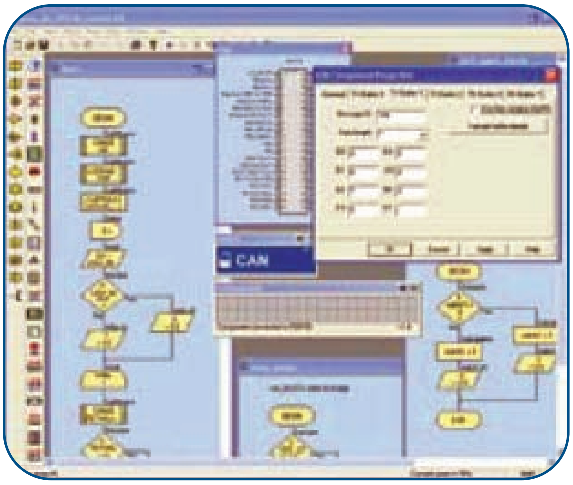
In summer, the car will return

■ FIGURE 7. The inverter in the boot.



■ FIGURE 6. Photos showing before and after pictures of the battery system.





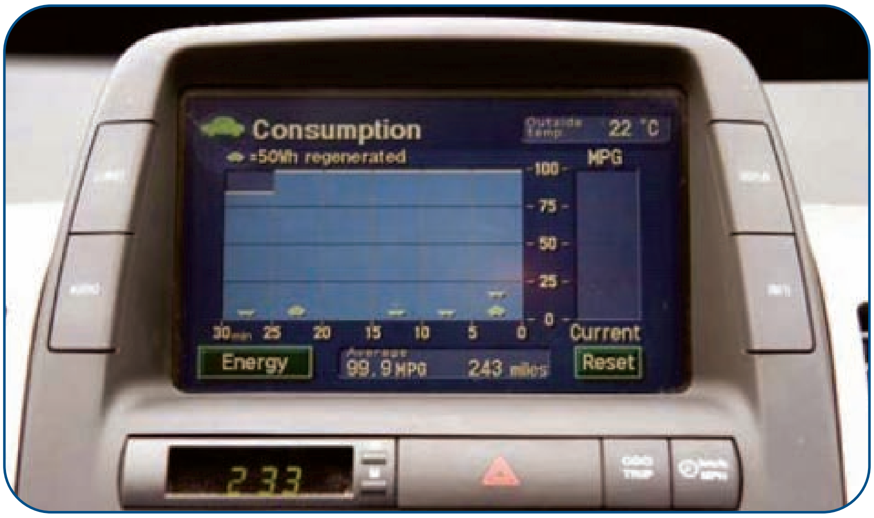
■ FIGURE 8. The Flowcode software used for developing the control system.

about 60 mpg in normal hybrid mode and about 100 mpg in battery boost mode. Unfortunately, the Prius' read-out only goes to 99.9 mpg so you are a bit blind as to how well it's really doing. In Figure 9 you can see a photograph of the Toyota's display which shows that the car has achieved at least 99.9 mpg.

Another job for Flowcode will be to read the instantaneous fuel flow from the bus along with the speed and compute the real fuel consumption.

Further gains can also be made by reading the bus speed signal and pulsing a relay when the speed drops below 30 mph to force the car into EV mode. The relay would be pulsed again on the speed rising to 30 mph to take the car out of EV mode — another job for Flowcode. **NV**

■ FIGURE 9. A close-up of the Prius display showing the 99.9 mpg achieved.



References

Elektor February 2006 Easy CAN
Microchip **www.microchip.com**
Plug-in Prius Wiki group at: **www.eaa-phev.org/wiki/Main_Page**
Follow links to plug-in hybrids, then Prius.

The E-blocks Easy CAN bus pack includes a copy of Flowcode, two PICmicro multiprogrammers, two CAN bus boards, an LCD display, switch board, LED boards, and accessories.

Matrix Multimedia **www.matrixmultimedia.com**