2005-01-0027

ECU Development for a Formula SAE Engine

Mario Farrugia, Michael Farrugia and Brian Sangeorzan
Oakland University

ABSTRACT

Motivated by experiences in the Formula SAE® competition, an engine control unit (ECU) was designed, developed and tested at Oakland University. A systems approach was taken in which the designs of the electronic architecture and software were driven by the mechanical requirements and operational needs of the engine, and by the need for dynamometer testing and tuning functions. An ECU, powered by a 68HC12 microcontroller was developed, including a four-layer circuit board designed for EMC. A GUI was written with Visual C++® for communication with a personal computer (PC). The ECU was systematically tested with an engine simulator, a 2L Ford engine and a 600cc Honda engine, and finally in Oakland's 2004 FSAE vehicle.

INTRODUCTION

The Formula SAE® (FSAE) competition exposes students to the tuning and mapping of engines, primarily due to the intake restrictor required by FSAE rules. The engines are controlled by programmable Engine Control Units (ECU's) that handle spark timing, fuel metering and also provide other auxiliary features. Development of an in-house ECU provides various advantages over the use of commercially available ECU's [1,2], namely: the use of stock crank and cam sensors (no missing teeth on the crank wheel); user-settable look-up tables for MAP, coolant and air temperatures, and oxygen sensors; an injector flow bench utility; ability to modify the system to suit individualized needs; and the ability to repair the unit rather than buy a replacement if damaged.

The microcontroller used was the Motorola 68HC12A4, which is a 16 bit MCU that runs at a clock frequency of 16MHz. It has an 8 channel, 8 bit analog-to-digital converter. Spark timing and fuel timing are the priority events, established as interrupts to the CPU. Highest priority is given to the signal from the crank teeth, lower to the cam tooth signal. Tasks, such as handling of slow sensors like temperature or pressure, are given much lower priority. Communication to the PC is done during CPU idle periods when no important interrupts are being serviced. Testing was performed in stages to systematically validate microprocessor function, hardware operation, algorithm performance, and finally the control strategy and interface software. Testing was done using: an electronic engine simulator, a mechanical simulator using motor driven crank and cam wheels, an unloaded 2L Ford engine, the FSAE 600cc Honda engine on a dynamometer, and finally in the 2004 FSAE car.

This paper will discuss the hardware architecture, the software, engine control strategy and testing.

THE MICRO CONTROLLER UNIT

The ECU board carries all the electronic hardware required for engine control: the microcontroller; signal conditioning and power output modules.

HARDWARE ARCHITECTURE

The PCB is four-layer and was designed for EMC. For this reason the digital, power and analog sections were kept separate, while the harness wires are shielded where appropriate. The ECU uses 512K of flash memory and 1Mbyte of battery-backed RAM. The program resides and runs in the flash memory. The flash is also used for storing all the ECU look-up tables and engine parameters, for safe keeping. 32 Kbytes of RAM are used for program variables, while the remaining 992Kbytes are used for logging engine operating parameters. This amounts to more than 1.5 hours of data logging at a rate of 10 readings per second for eight, 16bit-wide parameters. The contents of the flash and RAM are checked for consistency using CRC64 *check sum* at boot-up. The ECU has an on-board, battery-backed real time clock chip that keeps the time of day with an accuracy of 10 milliseconds.

Input / Output

The ECU has an 88 pin Tyco AMP® hybrid harness connector that provides ample contacts at two current ratings. Signals and small power output lines use the lower current contacts, while power devices such as coils and injectors use the higher current contacts.

The ECU features the following **analog inputs**:

- Coolant temperature
- Air temperature
- Throttle position
- Manifold absolute pressure
- Exhaust Oxygen
- Battery voltage
- Auxiliary inputs

The raw signals from the inductive type pickups are conditioned into TTL pulses for interfacing to the MCU. The following are the **digital inputs**:

- Crank Position – from toothed crank wheel
- Cam Position – from cam wheel, used to determine position in four stroke cycle
- Wheel Speed – for use in traction control
- Selected Transmission Gear

The following **outputs** are available:

- Fuel Pump Relay
- Radiator Fan Relay
- Alternator Relay
- Up-Shift Indicator Light
- Down-Shift Indicator Light
- Tachometer
- 8 Ignition Coil Driver outputs
- 8 Injector Driver outputs
- Auxiliary outputs

The ECU has a serial RS232 port for communication with a host PC. Another serial TTL port is also available for future expansion. The TTL port could be used as a communication channel with another processor working in parallel.

SOFTWARE

ECU EMBEDDED SYSTEM

The software for the 68HC12 was written in ANSI C and assembly language, using the ICC12 compiler from ImageCraft [3]. The Embedded software does not make use of an operating system. An operating system was seen as an overhead that would not be affordable when such a humble MCU has to cope with an engine running up to 14000 rpm!

The program is made up of two parts: the Boot Loader and the Application. The Boot Loader can be updated via the BDM interface while the application program can be updated via the serial port using the standard RS232 interface.

The 68HC12 can only address 32Kbytes of code at any time. The source code occupies about 64Kbytes and so portions of the code have to be paged in and out as appropriate. The upper 16Kbytes of program memory are always *visible* to the processor.

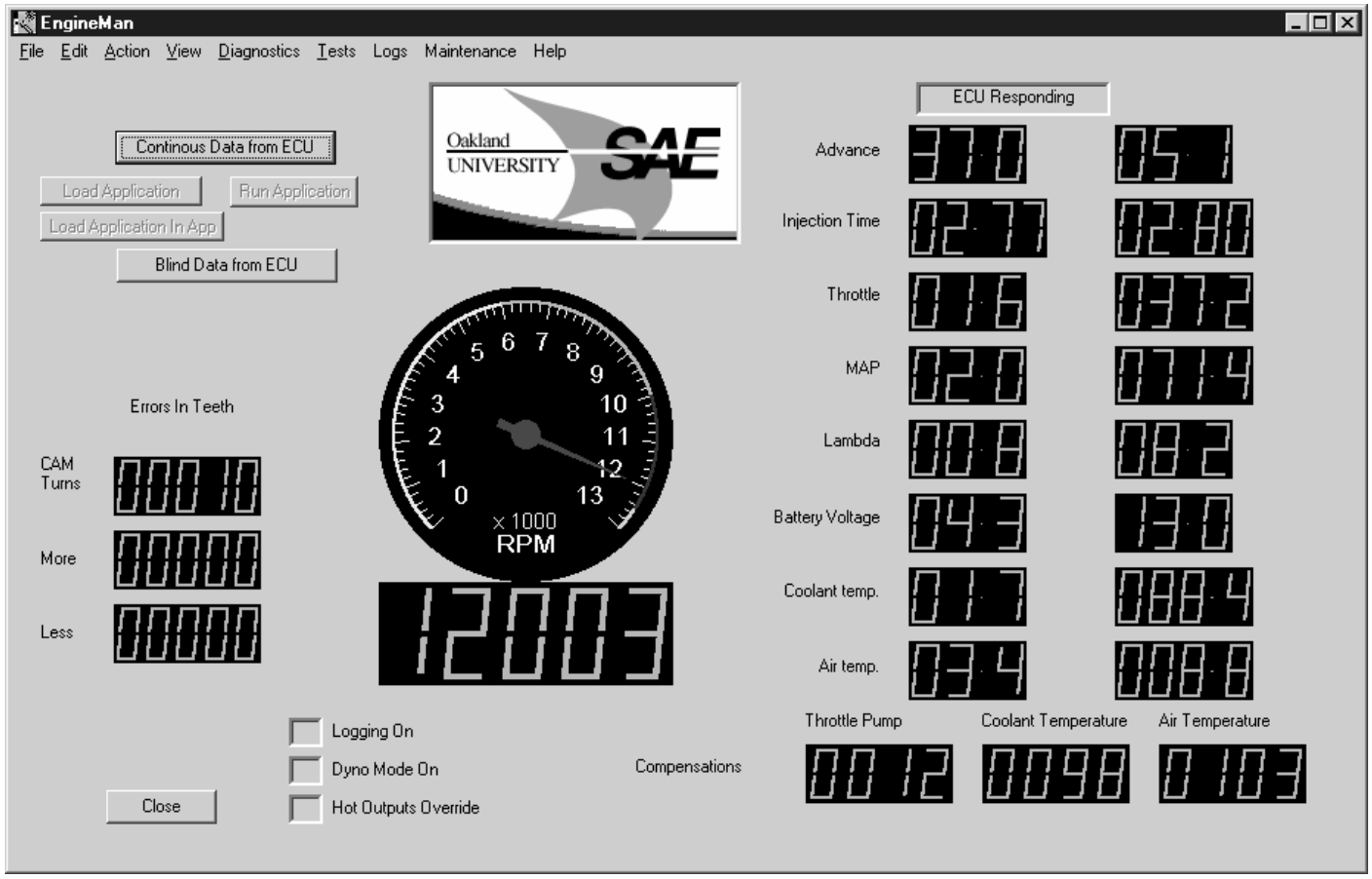


Figure 1: Main Dialog Screen: Engine Manager GUI (Shown here in grayscale for publication)

The Boot Loader is placed in the higher 8Kbytes of the memory map, which also contains the vector table. A portion of the application, which contains mainly the interrupt servicing routines, is put into the other 8Kbytes that are always visible. The remaining parts of the application are split into 16K pages in the flash.

The ECU application program governs the spark ignition advance angle and the fuel injection duration for the engine. It does so by using internal look-up tables for each of these two functions. The tables can be chosen to have TPS or MAP as the Load parameter.

The internal tables are made up of 16(Load) x 64(RPM) cells. The values inside these tables are determined, by means of a spline interpolation, from the smaller array of values entered by the user. At run-time, two dimensional linear interpolations are performed to determine the values to use for ignition and injection. The ECU also takes care of the compensations that are needed for temperatures and throttle pump.

ENGINE MANAGER GUI RUNNING ON PC

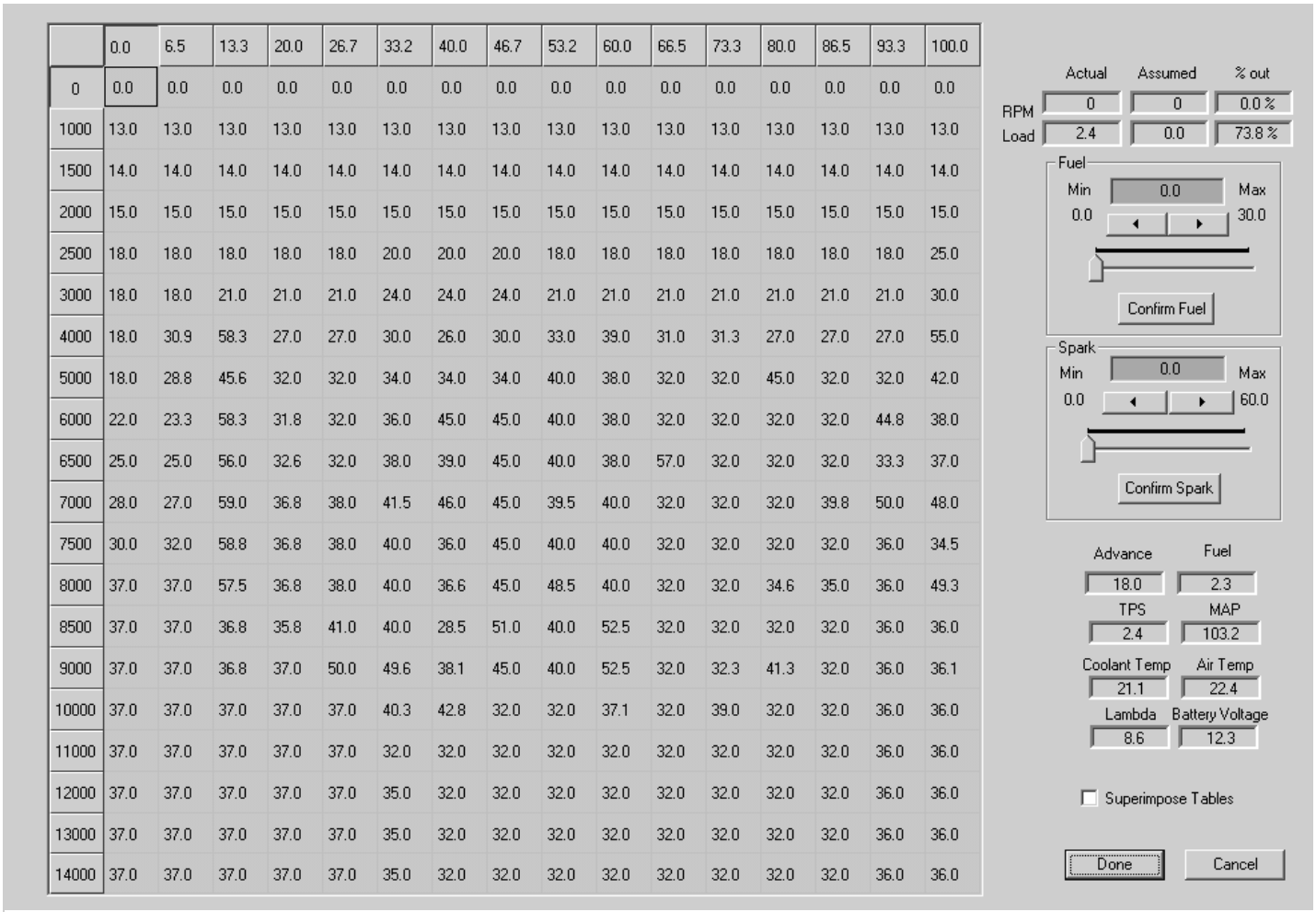
The PC software is written in C++ using Visual Studio from Microsoft [4]. The main dialog screen is shown in Figure 1.

The Engine Management program is used to:

- monitor the engine sensors
- update all engine parameters and tables
- setup and collect data logs from the engine
- edit the ignition and injection tables while tuning the engine on a dynamometer

Upon startup, the program first checks to see if the ECU is connected. If so, it downloads all the engine operating parameters to the PC's tables. If no ECU is detected it asks the user to choose a configuration file.

The main screen of the program shows the engine operating parameters that are sent from the ECU, in real time, every 100ms. The user can edit and upload the ignition and injection tables, even while the engine is running. Whereas commercially available ECUs require the use of specific sensors, the OU ECU allows use of the OEM sensor, or any sensor. The user would separately calibrate the sensor, and then enter calibration values into a look-up table for that sensor.



	0.0	6.5	13.3	20.0	26.7	33.2	40.0	46.7	53.2	60.0	66.5	73.3	80.0	86.5	93.3	100.0
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1000	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0
1500	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0
2000	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0	15.0
2500	18.0	18.0	18.0	18.0	18.0	20.0	20.0	20.0	18.0	18.0	18.0	18.0	18.0	18.0	18.0	25.0
3000	18.0	18.0	21.0	21.0	21.0	24.0	24.0	24.0	21.0	21.0	21.0	21.0	21.0	21.0	21.0	30.0
4000	18.0	30.9	58.3	27.0	27.0	30.0	26.0	30.0	33.0	39.0	31.0	31.3	27.0	27.0	27.0	55.0
5000	18.0	28.8	45.6	32.0	32.0	34.0	34.0	34.0	40.0	38.0	32.0	32.0	45.0	32.0	32.0	42.0
6000	22.0	23.3	58.3	31.8	32.0	36.0	45.0	45.0	40.0	38.0	32.0	32.0	32.0	32.0	44.8	38.0
6500	25.0	25.0	56.0	32.6	32.0	38.0	39.0	45.0	40.0	38.0	57.0	32.0	32.0	32.0	33.3	37.0
7000	28.0	27.0	59.0	36.8	38.0	41.5	46.0	45.0	39.5	40.0	32.0	32.0	32.0	39.8	50.0	48.0
7500	30.0	32.0	58.8	36.8	38.0	40.0	36.0	45.0	40.0	40.0	32.0	32.0	32.0	32.0	36.0	34.5
8000	37.0	37.0	57.5	36.8	38.0	40.0	36.6	45.0	48.5	40.0	32.0	32.0	34.6	35.0	36.0	49.3
8500	37.0	37.0	36.8	35.8	41.0	40.0	28.5	51.0	40.0	52.5	32.0	32.0	32.0	32.0	36.0	36.0
9000	37.0	37.0	36.8	37.0	50.0	49.6	38.1	45.0	40.0	52.5	32.0	32.3	41.3	32.0	36.0	36.1
10000	37.0	37.0	37.0	37.0	37.0	40.3	42.8	32.0	32.0	37.1	32.0	39.0	32.0	32.0	36.0	36.0
11000	37.0	37.0	37.0	37.0	37.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	36.0	36.0
12000	37.0	37.0	37.0	37.0	37.0	35.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	36.0	36.0
13000	37.0	37.0	37.0	37.0	37.0	35.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	36.0	36.0
14000	37.0	37.0	37.0	37.0	37.0	35.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	32.0	36.0	36.0

Figure 2: Ignition and Injection Editing Screen in Dyno Mode (Shown in grayscale for publication.)Uppermost row is the Load Parameter, while first column is engine rpm.

In dynamometer mode, the ECU interpolation and compensations are switched off so that the exact values of spark timing and injection duration are known. While the engine is running, the specific Load (MAP or TPS)/Speed point is identified by a highlighted cell on the GUI. The value in that cell (either spark timing or fuel injection duration) can be modified in real time by dragging a slider with the mouse, Figure 2.

Engine operating parameters are continuously logged and can be downloaded to the PC once a communication link has been established. The data are available in a text file for later analysis.

ENGINE CONTROL STRATEGY

As stated previously, engine control is achieved by the use of look-up tables for spark timing and fuel injection duration. The user can enter values into the look-up tables using available data, or real time during engine operation.

HIGH AND LOW PRIORITY TASKS

Spark timing and fuel injection timing are the most important events. Timing is based on knowledge of crankshaft position (within the cycle) and speed obtained from the crank and cam sensors. These priority events are interrupts to the CPU. Highest priority is given to crank tooth events, less to cam teeth. Control of spark and fuel timing, to one crank angle degree resolution, is achieved by timers that work from the last known physical position of the crankshaft, that is last crank tooth event.

Tasks such as handling of slower sensors, like temperature or pressure, are given much less priority. Communication to the PC is done during CPU idle periods when no important interrupts are being serviced.

SPARK TIMING AND FUEL INJECTION LOOK-UP TABLES

The look-up tables within the processor are 16 by 64, 16 Load divisions and 64 Speed divisions. The number of Speed divisions is necessary for the processor to perform accurate timing calculations for spark and fuel, especially at the higher RPMs. Timing calculations are based on the engine speed, and referenced to the last known crank event. Accurate determination of Speed results in lower uncertainty in the spark and fuel injection timing. In the GUI, refer to Figure 2, a limited number of cells are available to the user to limit the time required to enter values into the tables. The PC software then applies spline functions to the user-specified cells values and fills the extended look-up tables. The extended look-up tables are then sent to and stored in the ECU.

CRANK AND CAM SIGNALS

Some commercially available ECUs require the use of specific crank wheels with missing teeth (non-uniform spacing). Since the Honda F4i engine has 12 evenly spaced teeth (no missing teeth), it was preferred to use the standard OEM setup. Furthermore it is the authors' opinion that having uniformly spaced crank teeth allows more precise knowledge of crank position throughout the whole engine cycle. The software expects a single cam signal that resets the crank tooth counter, thereby restarting the engine control loops. The ratio of crank to cam tooth events is monitored, and any errors are shown on the GUI. Once the sensors are properly setup, no crank-cam errors occur.

TESTING

The testing of the ECU board was done in various steps, first checking microprocessor and hardware functionality, and later the control algorithms.

TESTS ON ENGINE SIMULATOR

Initial testing of the ECU board and software was done with a signal generator that supplied a crank signal, and a divide-by-24 circuit that supplied the cam signal. The load parameter (TPS or MAP) was simulated by a potentiometer. The algorithms were checked for proper operation, and tested for simulated engine speeds up to 14000 rpm. The ECU outputs were supervised by means of a storage oscilloscope.

Additional engine simulator tests were performed using a variable speed DC motor rotating a 16 tooth crank wheel, and a single-tooth cam wheel rotating at half speed by means of a toothed-belt reduction. This mechanical apparatus provided a means by which the signal conditioning for the inductive-type pickups was configured and checked without having to run the engine.

UNLOADED ENGINE TESTING

Once testing with the simulator was complete, engine testing began using an unloaded 2L SI engine. This engine had an upper speed limit of 5500 rpm and so it was not able to take advantage of all the speed capability of the controller. However, it did allow testing of the control strategies on a running engine. The main purpose of these tests was to check starting capability and immunity to noise.

This engine was setup with a 16 tooth crank wheel that provided the opportunity to test the software's ability to properly read crank wheels having differing numbers of teeth. The load parameter in this case was MAP, rather than TPS, to test its use. The duration of injector open-

time was quite different than the injector open-time on the FSAE engine.

LOADED ENGINE TESTING

The controller was then tested on the FSAE Honda F4i 600 engine that was connected to a Clayton water-brake dynamometer. The engine was left with its stock crank and cam sensors. The stock cam wheel, with three teeth, was modified to have just one tooth, but the crank wheel was left in the original configuration of 12 teeth (none missing). Testing was performed to confirm proper triggering from these sensors. The variable reluctance sensor drivers were configured to trigger on the middle of the tooth, not the sides, as this remains consistent over large variations in engine speed. The crank sensor is always responsible for the information on engine position while the cam sensor only resets the algorithm counters.

The ignition timing was checked against the values specified in the ignition table by strobing a protractor setup on the crank. The ignition timing values in the table were visually confirmed, from cranking speeds up to a speed of 10,000 rpm, with an accuracy of one crank angle degree.

The dynamometer tests were performed using the stock F4i intake and exhaust manifolds. The Load parameter used during the FSAE engine testing was TPS. The ability to use the GUI to modify the look-up tables for ignition timing and fuel injection duration was verified.

Finally, the ECU was tested on Oakland University's 2004 FSAE car. The FSAE car has an intake restrictor of 20mm as per FSAE rules, which necessitates very different ignition and fuel look-up tables. The look-up tables for the restricted configuration were compiled based on past experience with restricted FSAE engines and the car ran quite well. The next objective is to put the engine with the restricted intake on a dynamometer and adjust the look-up tables by experiment.

CONCLUSION

An engine control unit (ECU), with the associated embedded software and a GUI (for MSWindows®) was designed and developed. The engine controller's ability to start and run the engine from the stock crank and cam sensors (two wire) was tested. Proper operation of the controller at high engine speeds was also tested; 12500 rpm rev limited on the real engine, and 14000 rpm on the simulator. Loaded operation was tested on the dynamometer using an unrestricted intake and also using a restricted intake on the FSAE 2004 car.

ACKNOWLEDGMENTS

We would like to thank Tyco AMP®, ST Microelectronics, National Semiconductor, Infineon and Macronix for providing free samples of their products.

REFERENCES

1. Haltech E6S Instruction Manual, Haltech Engine Management Systems Sydney Australia, www.haltech.com.
2. DTA P8 PROUser Manual, DTA Competition Engine Management Systems Bradford England, www.dtafast.co.uk
3. ImageCraft Creations Inc., www.imagecraft.com
4. Microsoft Visual Studio®, www.microsoft.com

CONTACT

More info on the Engine Management Software and ECU at www.oakland.edu/~mfarrugi/FormulaSAE and www.reataengineering.com

ACRONYMS,

BDM Background Debug Mode
CPU Central Processing Unit
ECU Electronic Control Unit
EMC Electro Magnetic Compatibility
GUI Graphical User Interface
MAP Manifold Absolute Pressure
MCU Micro Control Unit
PCB Printed Circuit Board
RAM Random Access Memory
TPS Throttle Position Sensor
TTL Transistor-Transistor Logic