

Calibration

Calibration is a process of optimizing or “tuning” a control algorithm to get the desired response from the system. A calibration tool is a combination of a hardware interface and a software application that enables the engineer to access the “calibration variables” in an ECU and change them. Typical control algorithm components that need calibration are look-up tables, gains, and constants. The *structure* of the control algorithm is not changed during calibration, i.e. you cannot add a “D” loop to a PI controller to form a PID loop during calibration - you can only change the P, I, and D constants.

Calibration is a downstream process of algorithm design tools such as [Rapid Controls Prototyping \(RCP\)](#) and [Automatic Code Generation](#). The structure of the control algorithm and initial calibration is established in RCP. The algorithm is then coded manually or with auto-code tools. The algorithm is then optimally adapted via the calibration process. Some examples of calibrations follow:

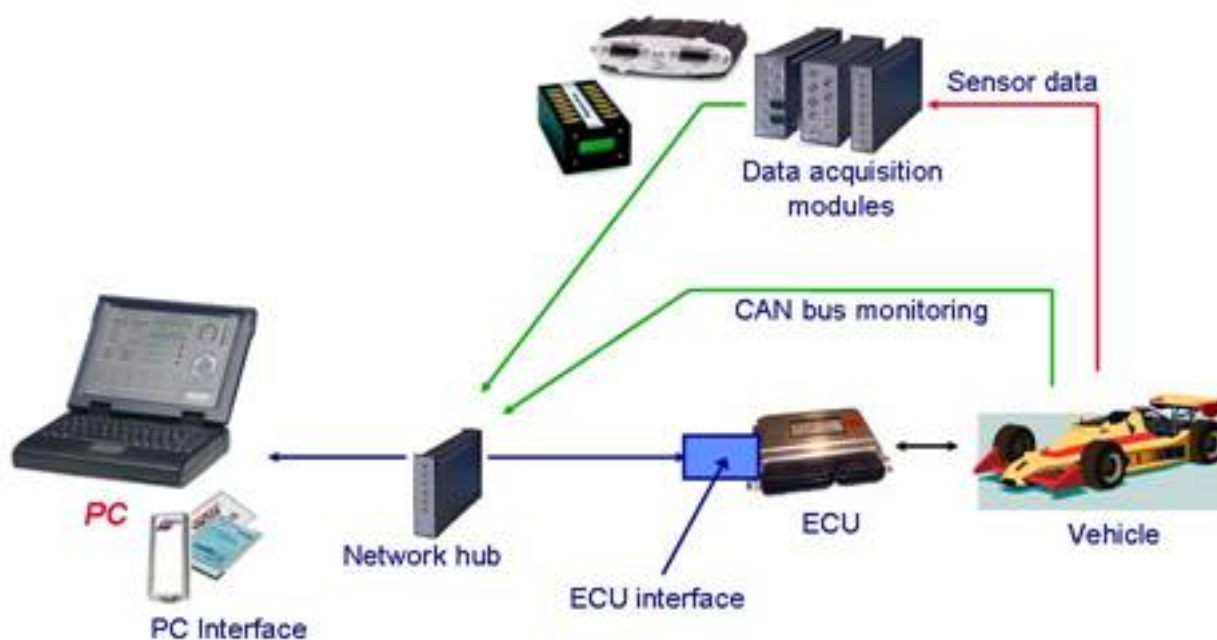
1. Sensors: Different vehicles may use a different sensor suite such as variations of mass-air-flow, air pressure, temperature, etc. These different sensors will all have unique responses. The calibration of a look-up table for linear-izing their response will be needed depending the on vehicle and sensor suite used.
2. Performance: A vehicle can be tuned for varying performance such as power output vs. fuel economy.
3. Constants: A different set of parts on a vehicle may require the change of algorithm constants. For example, the use of larger tires may necessitate a change in the calculation of vehicle speed.

A powertrain control algorithm may have hundreds of calibrate-able parameters. The more parameters that are used, the more difficult the task of finding an optimal calibration. The calibration tool helps the engineer arrive at an acceptable calibration parameter set. All of the calibrate-able parameters are grouped into a special section of ECU memory called the calibration memory. Calibration tools give the user access to this memory to allow “tuning.”

A basic calibration system consists of an ECU interface, a link to the host PC, and a PC application. A more capable system will add a vehicle network link as well as analog data acquisition modules. The ECU interface is typically a CAN interface when a CAN-

based calibration method is used, or a Read-Only-Memory (ROM) emulator when a direct memory access method is used. The link back to the host PC can be CAN, USB, Ethernet, or other method. The PC application is typically a MS-Windows® application. Figure 1 shows a typical calibration system.

Calibration System Definition



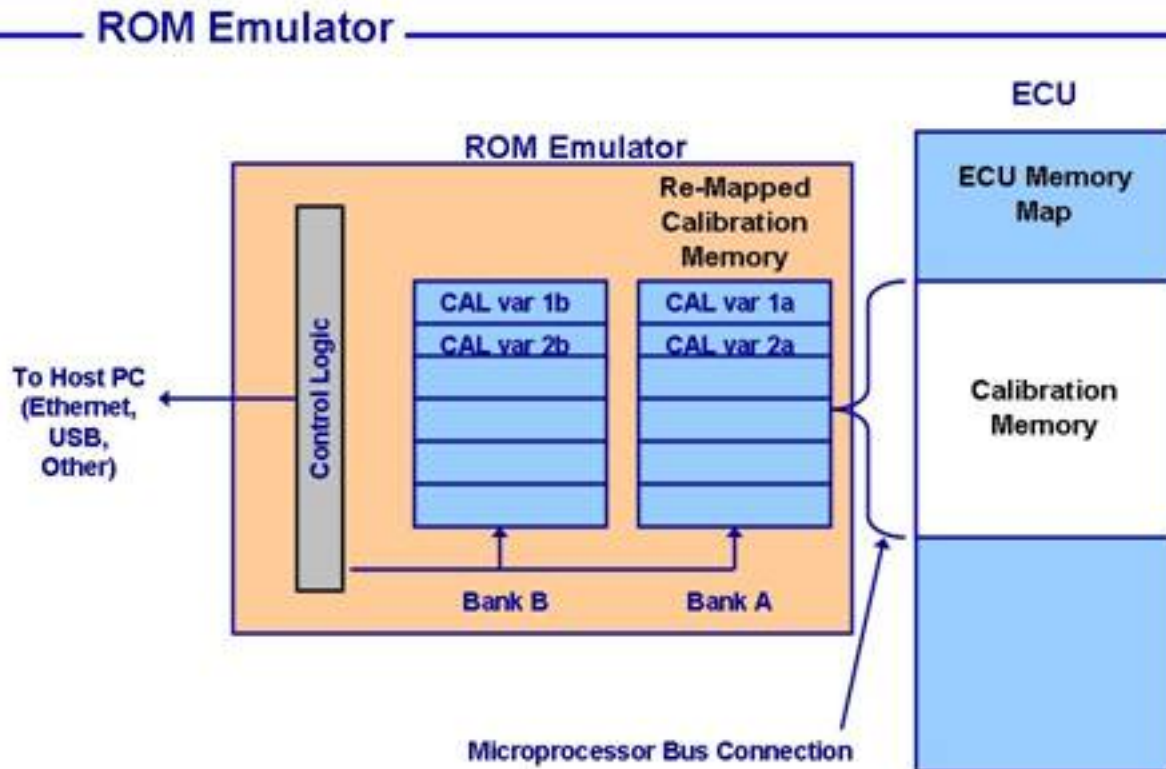
©2005 PrecisionMBA, LLC

Figure 1: A typical calibration system

Let's discuss the ECU interface first. The ECU interface is either a CAN link or ROM emulator. Each method has benefits and liabilities and the selection of the right method is dependent on the intended application. A description of each follows:

ROM Emulator Interface: This interface is chosen for the most difficult ECUs, i.e. the ones with the most calibration memory such as engine control units or transmission control units. The reason is speed. The ROM emulator is physically attached to the bus of the microcontroller on the ECU. The calibration memory is mapped off the ECU and onto the ROM emulator. By virtue of now containing the calibration memory, the ROM emulator device can log and manipulate the variables. Speed in switching parameter sets is achieved by "bank switching" the calibration memory. Also, the processor is completely unaffected by the calibration tool - it doesn't even "know" it is there. The downside of ROM emulation is that it is invasive, that is, it needs to

connect to the ECU data bus. This can be difficult and costly as it can require Non-Recurring Engineering (NRE) work to “stuff” a connector onto the ECU, or a custom designed ROM emulator adapted to the ECU. The tool manufacturers try to minimize the adaptation costs, but they are still there. Figure 2 shows a basic ROM emulator architecture.



©2005 PrecisionMBA, LLC

Figure 2. ROM Emulator ECU interface architecture.

CAN ECU Interface: The CAN ECU interface is connected to a CAN port on the ECU. Most transportation ECUs use a variant of the CAN bus for vehicle ECU connection. The CAN ECU interface is dependent on an ECU resident calibration service routine to manipulate the calibration memory. The host PC calibration application communicates to the ECU and service routine via a special protocol called CCP or CAN Calibration Protocol. There is also a new variant of this protocol with functional extensions called XCP. The advantages of a CAN calibration interface, often called CCP calibration, is that it is minimally invasive to the hardware of the ECU. It is just a matter of connecting to the CAN bus. Often a dedicated CAN port is necessary for best performance. It is also the least expensive calibration method since an expensive ROM emulator and NRE is not needed. The downside is that the CCP protocol and calibration service routine are executed by the ECU and thus stealing computing

cycles from the ECU. There is also a bandwidth limitation on acquiring and logging variable data since the ECU must send this information out on the CAN bus. Figure 3 shows CCP calibration.

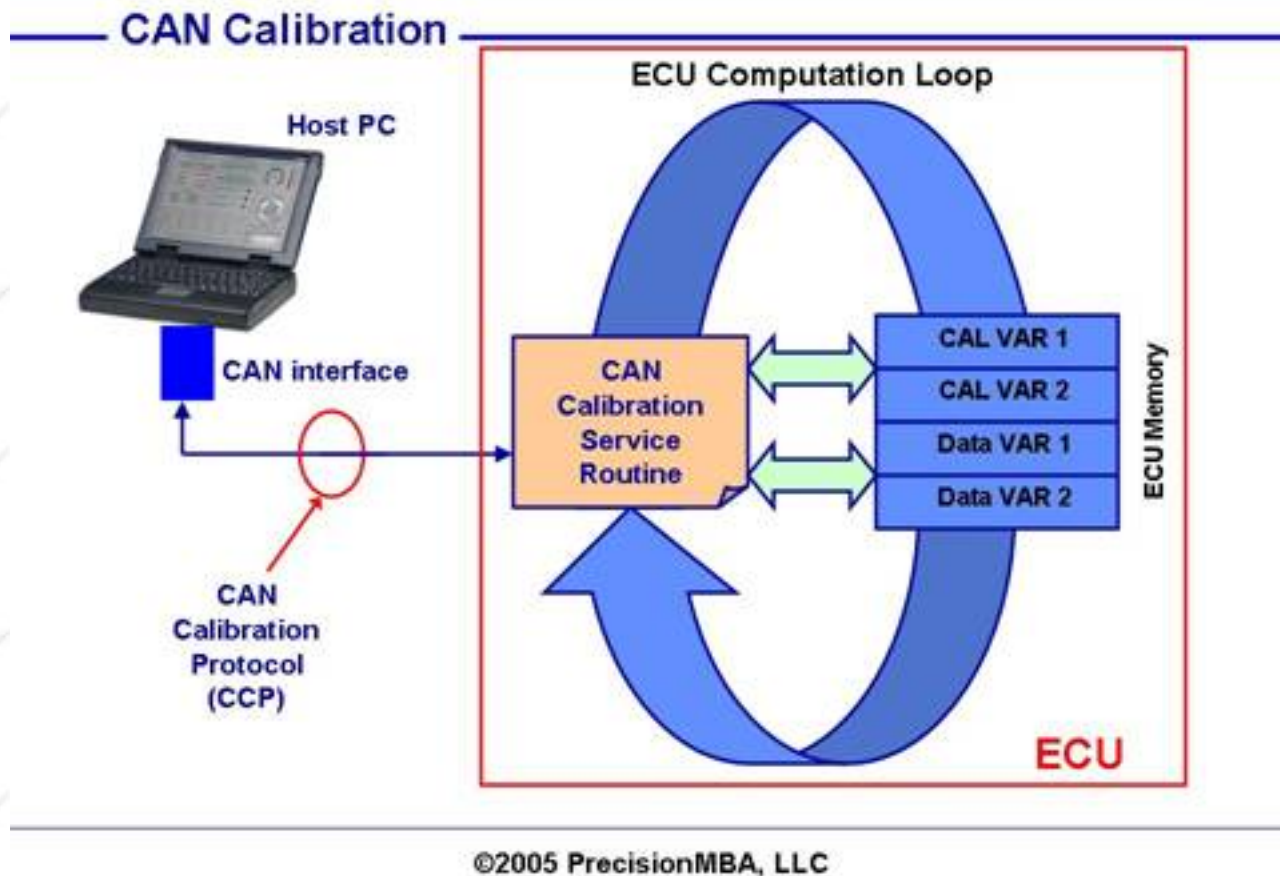


Figure 3. Calibration via CAN Calibration Protocol (CCP)

CAN Network Data, Analog Data, and the Network Hub: The ECU interface is primarily used to change the calibration variables. As mentioned above it is also a data collection tool. During the calibration process, the engineer may want to collect and correlate data from the ECU. An example of ECU data would be RPM, spark advance, or injection pulse width. Thus the ECU interface, regardless of method, is providing data to the host application. In addition to ECU data, the calibration engineer may want to collect data external to the ECU. This data can be on the vehicle CAN bus, such as what gear the vehicle is in, or external "analog" data such as the ambient temperature. To get this data the calibration tool must have a connection to the vehicle bus and/or connection to data acquisition devices. Please refer again to Figure 1.

Adding the connection of the vehicle network and analog data to the calibration

system requires the use of a device to coordinate and time-stamp the data. Time-stamping is needed for the host PC calibration application to be able to meaningfully log and display the data. To attain data correlation from these different sources, a network hub is used. The network hub assures that the host PC will get time correlated data.

Calibration Application: The user's window into the calibration world is the MS-Windows® calibration application. This application does the following:

1. Reads the calibration data file: Every ECU can have a different calibration memory layout - although most OEMs try to standardize this architecture. Thus, there needs to be some way of telling the host application where the variables are in the ECU and what they mean. This is done via a variable database. There is a popular standard for this database called ASAM-MCD-2MC (yikes!) or shorter, ASAP2, or shorter yet, "A2L." This database, also called an "ECU description file" defines the location of the variable in memory, their type, scaling, and units. The data can then be fetched and displayed or stored in a meaningful way by the host application. Systems that read files like this are called "data-driven systems" because they are essentially useless without them.
2. Read, Write, Modify: Self explanatory, the tool lets the user change the calibration memory. More deluxe operations are bank switching for before/after analysis, or movement of "chunks" of memory.
3. Dataset Management: The host application has data browsers that ease the storage and retrieval of calibration datasets.
4. Data Logging: The host can log, correlate, and display data from all sources.
5. Trigger: Some calibration tasks are episodic and thus require trigger sources and actions. An episodic trigger could be coolant temperature, speed, etc. The action could be to log data after the trigger.

In short, the calibration application gives the calibration engineer a tool set for the capture and analysis of relevant data and the ability to react to this data by modifying calibration variables.

Calibration is a large and costly activity for OEMs in the transportation industry. Entire engineering departments and fleets of vehicles are dedicated to the art of calibrating. Why do they do it? Well, right now there just is no other way to do multivariable optimization on these control algorithms. Some day there may be more automated methods, plug-and-play sensors, or self-optimizing algorithms, but until

then calibration is a needed process.

If you are considering calibration tools and need advice, please [contact](#) us.

[Home](#)

©2005 PrecisionMBA, LLC