Trends in engine control

Reuse and standards are altering the engineering of software, which represents one third of the total cost of an electronic control unit.

by ***Terry Costlow***

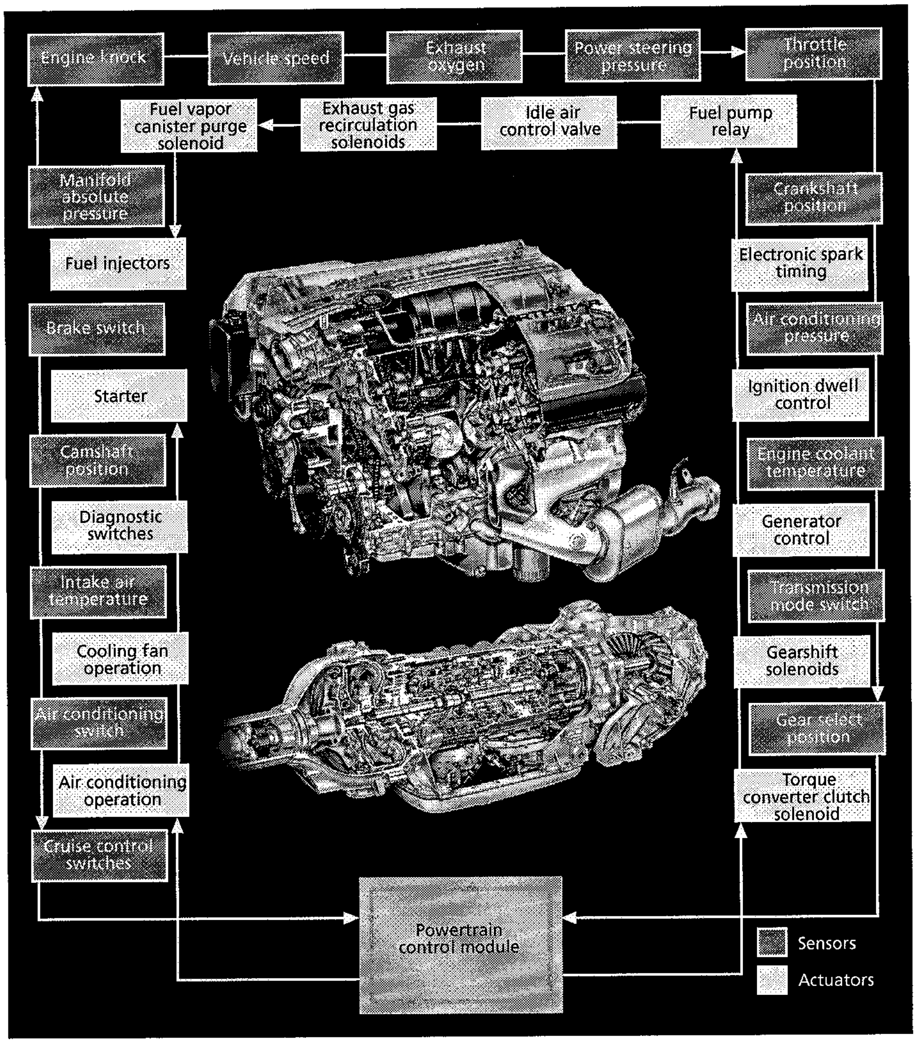
As software becomes an increasingly critical part of vehicle development, engineering managers throughout the automotive industry are shining the spotlight on development costs and strategies. In engine control systems, reusable C modules and standard operating systems are becoming the norm, though changes are coming slowly.

Software was rarely discussed in the past, but it's become such a key factor in automotive design that it can no longer be overlooked. Software costs today are becoming a significant factor in the cost of a vehicle, and the soaring impact of electronic subsystems means software costs will rise sharply in the next few years.

"In the year 2000, software was 4% of the total value of a car. In 2010, it will grow to 13%," said Wolfgang Dehen, President and CEO of **Siemens VDO Automotive**.

When it comes down to the electronic systems, that percentage obviously rises dramatically. "One third of the total cost of a control unit is software," said Hans-Georg Frischkorn, Senior Vice President of electrical and electronics development at **BMW** AG.

In engine controllers, software is the key technology being tapped to meet tightened emissions standards and



Software in GM's powertrain controllers receives input from sensors, determines what actions to take, then sends commands to actuators.

improve engine performance. As designers move from 16- to 32-bit processors, the focus is on strategies that maximize reliability while keeping costs and development time low. Reusable modules, automatic code generation, OSEK-compliant operating systems, and commercial software are among the options being employed. (OSEK is a German abbreviation for Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug, or, in English terms, Open Systems and the Corresponding Interfaces for Automotive Electronics.)

The basis for all these trends began a few years ago when engineers started moving away from 8-bit microcontrollers. That prompted a move to C, the high-level language used practically everywhere, with assembler code remaining in only a few rapidly fading legacy programs.

"With 8 bits, almost all the code was in assembler language. When the move to 16- and 32-bits began about five years ago, then applications had to be written in C," said Ashok Ramaswamy, Manager of software and forward engineering at **Delphi**.

Reusable code

Using C makes it simpler to port code from one processor to another, so programmers are now focusing on the creation of reusable modules. Modules can be moved from one processor to another, and they are much easier to manage because they can be grouped into functional blocks.

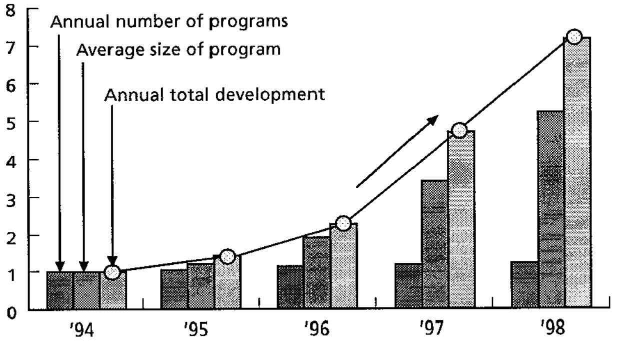
Juergen Wiesenberger, Director of gasoline systems at Siemens VDO Automotive, noted that when Siemens first switched to 16-bit chips and began using modules, there were 400 discrete modules. When the transition to 32 bits was made, Siemens adopted an architecture that combined those modules by functional groups.

"Instead of 400 modules, we have 60-70 aggregates, which we define as a container of modules grouped by function," Wiesenberger said. Reuse helps lower costs and development time by using code that's already in use. That also means that the code has been tested thoroughly, which should improve long-term reliability.

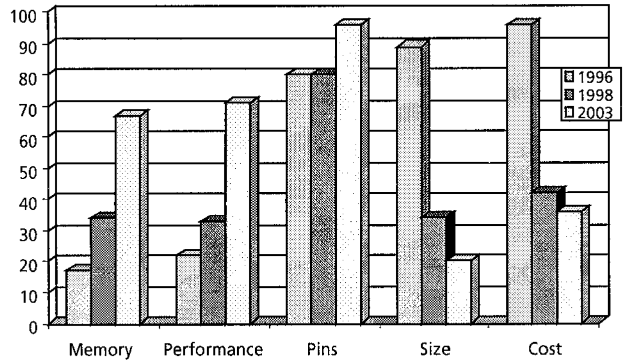
"We're seeing a real paradigm shift from proprietary software to reusable code," said Martin Duncan, Technical Marketing Manager at **STMicroelectronics**' Digital Segment in Livonia, MI. "The revolution is not happening yet, but the tendency is definitely there."

The speed of this transition to reusable code depends on many factors. Foremost among them is the target processor. Those who are now shifting from 16- to 32-bit architectures may find that it is difficult to port software to the newer devices.

"In practice, only a small percentage of the code can be reused—about 20%—when you go from 16- to 32-bit controllers," said Andrew Noble, Senior Manager of controls and electronics at **Ricardo**, PLC, a Shoreham, England design house. However, when chip architectures are the same and designs are simply moving to successive models of the CPU, the benefits can be much greater. Often, substantial portions of the applications package can be retained.



At Toyota, the number of engine-control software programs has not risen, but their size has skyrocketed.

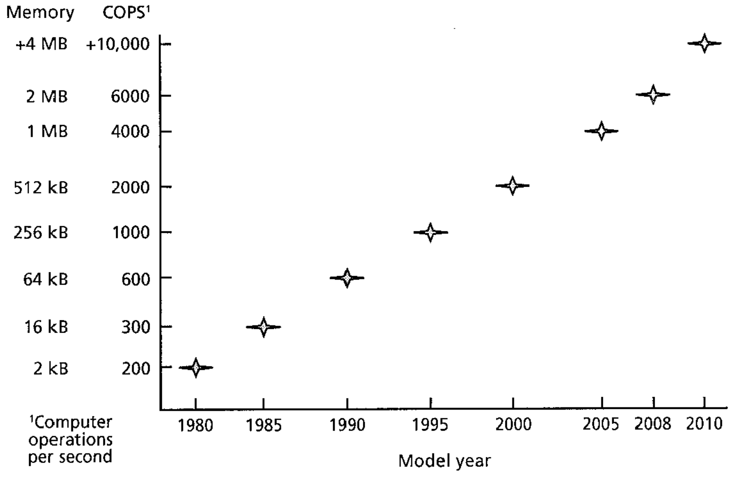


The dramatic increases in memory size underscores the growth of software in GM's engine controllers, which, though smaller and less expensive, offer much higher performance.

"In one application, we helped port from one processor to another. In the application software—about 80% of the total software—we found 98% was reusable and 2% was platform-dependent," Duncan said. He noted that, with device drivers, which account for about 9% of the total code, 80% of the code was reusable. However, in the 11% of the program that included the operating system, there was no reusable code.

While reusability is seeing solid acceptance, it is not something that will happen overnight. Most designers say that fully utilizing the power of the 32-bit chips requires writing code far beyond what was needed for 16-bit designs. That is evident in the amount of memory allocated to different types of processors. Ramaswamy noted that today's 32-bit chips have from 512 kB to 1 MB. That is far more than the 128 kB of ROM generally needed for programs on 8-bit microcontrollers. Even 16-bit microcontrollers required only 256 kB.

Though suppliers at all levels are looking at ways to reuse code, others note that it is not always possible. Some carmakers say that minor differences in the devices can bring major changes in software, so each line must be closely examined, or written from scratch, to make sure it is compact and runs efficiently.



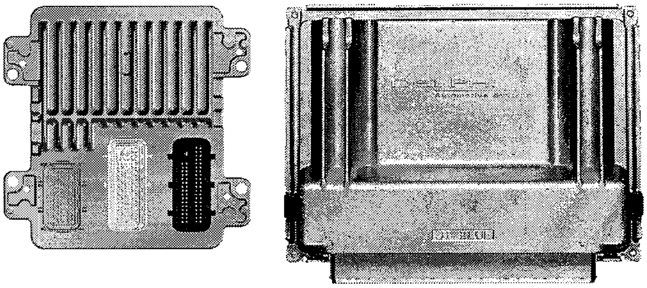
GM expects memory allotted to powertrain control software to continue its steady growth as faster processors do more tasks with increased precision.

"We've found that, when you really get way down in the processor and its support peripherals, there can be timing differences when you're running software on a **Motorola** part versus an **Infineon** part. We have to go through the code to understand how the processor uses it and supplies commands to the peripherals," said Dennis Bogden, Director of powertrain electronics engineering at **General Motors**.

Most observers feel that, though it may not work everywhere, software reuse is going to become increasingly popular. "We hear from customers that they have made a significant investment in software, and they're very keen on maintaining that software," said Ross McOuat, 32-bit Operations Manager for Motorola's 32-Bit Embedded Controller Division in East Kilbride, Scotland.

Another way to reduce the time and money spent on software development is to automate the process. In addition, the resulting code has higher-level descriptions, so it is usually much easier to alter and repair long after it is written. That resolves a common problem that arises when the author of a program is no longer at the company and no one can figure out what to do.

Today, autocoding is used primarily for prototypes and other quick-turnaround applications that are not going into production. But at least one observer sees a strong future for it. Ricardo's Noble does not expect to see automatically generated code in engine controllers for another five years, but he predicts that its use eventually will be become widespread. "Around 2008-09, you will see hand-written C going the same way as assembler language," Noble said.



Delphi is moving engine controllers to 32-bit processors, prompting a greater emphasis on simplifying software development.

Most designers feel that computer-generated software just does not meet the stringent demands of engine controllers, which must operate smoothly for decades. "It comes down to code optimization, memory, and throughput. Humans still generally write more efficient code," Wiesenberger said.

Others question just how much can actually be gained by letting a computer generate software. At GM, Bogden figures that only about 10% of a common design effort is spent coding software.

"Ninety percent is spent on the algorithm, putting it into the right form for our architecture. Doing the coding is a very small portion of the work, so the benefits of autocoding are limited," Bogden said.

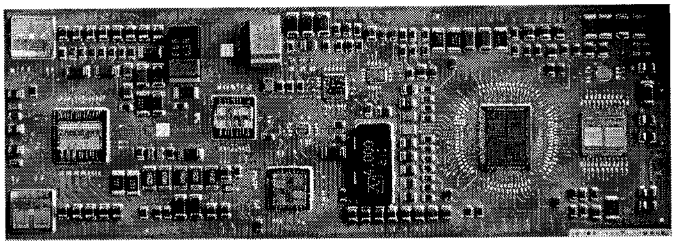
To Bogden, those algorithms, which define how things are done, are much more important than the software, which is the implementation that actually goes into the computer. "What we consider the most important part of our intellectual property is our algorithms," Bogden said.

OS standards

Reusable modules and autocoding dominate the applications area, but there is an equally noteworthy change occurring in operating systems. The OSEK operating system is taking hold, and that's opening the door for commercial operating systems.

OSEK was created in Germany in the early 1990s by BMW, **Bosch**, **DaimlerChrysler**, **Opel**, Siemens, **VW**, and the IIIT (Institute for Industrial Information Technology) of the **University of Karlsruhe**. The open architecture covers three areas: the operating system, communications, and network management. Many automakers are adopting it, with some saying they eventually will eschew other alternatives.

"OSEK will be the standard for all products developed by BMW," said Frischkorn.

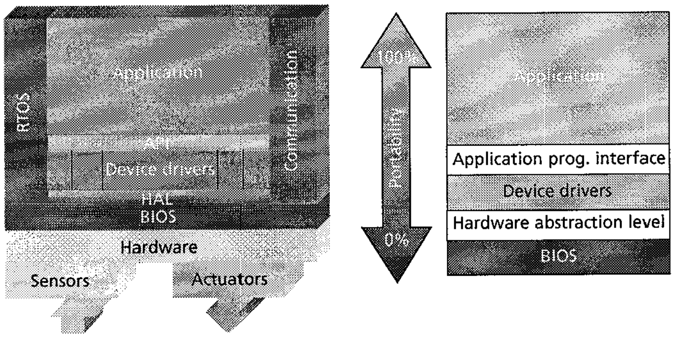


A Magneti Marelli controller for the Fiat Punto houses power, smartpower, and Flash memory chips, with most of the Flash dedicated to program software.

All of the new programs at Siemens VDO employ OSEK, and the company hopes to extend usage in the future. "Now, it's only a small part of the solution, but we hope it will grow," Wiesenberger said.

Though OSEK is making inroads, its acceptance is likely to be slowed by vendors who want to hold onto proprietary technologies. If companies feel their internal operating system gives them a competitive edge, the benefits of standards will be ignored, most observers agree.

That is equally true in a related move towards commercial operating systems. In the past three years, OEMs have moved towards commercial operating systems. **Visteon** has commercial real-time operating systems (RTOS) in production, feeling that these products are more cost-effective than its proprietary operating systems.



Software reuse is critical in applications packages and the applications program interface, but declines on layers that are closer to the hardware.

Diagnostic software adds capabilities

Diagnostics are becoming a central part of engine control, and the trend shows no sign of abating. Tighter emissions standards and a desire for more information are putting more importance on diagnostic software.

It's already a significant portion of the job of controlling an engine. "Today in our engine controllers, about 40% of the software is dedicated to running the engine. About 30% is used to run diagnostics," said Dennis Bogden, Director of powertrain electronic engineering at **General Motors**.

About 20% of the code is dedicated to interfaces for subsystems like anti-lock braking, and the remainder goes to overhead: the operating system, math libraries, and other things, Bogden said.

That makes diagnostics the second largest part of the design equation. But increased emphasis on system monitoring means an expanding role for diagnostic software. As auto designers and government regulators continually want more information, the importance of diagnostics is growing faster than other applications.

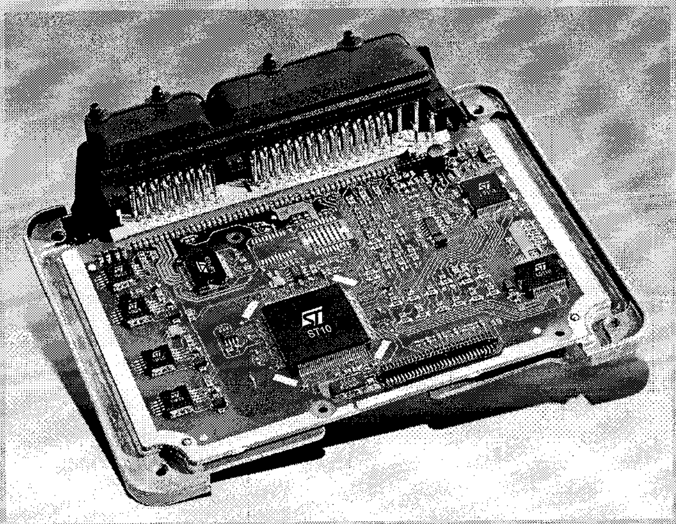
"We're now at the point where the diagnostic part in some areas is bigger than the other parts of a system," said Andreas Greff, President of **IAV** Inc. That will not be unusual in the future. Emissions standards are being tightened in many countries, putting more strain on diagnostic controls.

"Every time emissions standards get tighter, diagnostics must become more robust," Bogden said. For example, he explained, if timing drifts in a system today, it can go 15% away from mid point before it is flagged. As emissions standards tighten, that 15% also becomes tighter, forcing programmers to make their diagnostic algorithms more precise.

Engine diagnostics sometimes extend far from the engine compartment. Legislation now requires notification when fuel system leaks are identified, posing big challenges to designers and programmers.

They must look at everything from the fuel pump to the gas cap, figuring out one approach for all makes and models. That

Improved diagnostic software helps engineers test performance via laptops that store data continuously while the engine is being driven, according to IAV.



Diagnostics are a central aspect of the Magneti Marelli controller used by Fiat, Renault, Volkswagen, and PSA.

means accounting for variations in fuel tank geometries, fuel lines that run next to exhaust pipes that dramatically alter temperatures, as well as detecting loose gas caps.

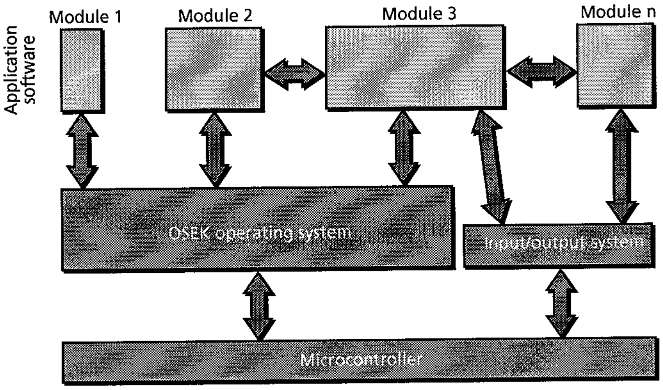
Software is once again the technology that solves the problem. "When the engine is off, there's a natural vacuum. When the car is shut off, we measure pressure in the fuel tank. By determining the starting pressure and how fast it decays, we can determine whether there's a leak or not," Bogden said.

Reaching across the entire fuel system is only one way that diagnostic programs are doing far more work. Many engineers are starting to design programs and hardware that give technicians and drivers more information. Simply lighting a warning light on the dash is becoming passe. Check engine lights or warnings are starting to provide more information.

"Certain manufacturers require systems that warn you not just if the temperature is going up, but why it's going up," said Ashok Ramaswamy, Manager of software and forward engineering at **Delphi**. "There's definitely a trend for them to ask for more."

Looking a bit further out, it's somewhat likely that cars will soon warn drivers or technicians when it seems that a component or subsystem is close to failure. "We have discussed doing predictive diagnostics, like some of the airlines do, warning that something seems to be nearing a problem," said Juergen Wiesenberger, Director of gasoline systems at **Siemens VDO Automotive**.

Eventually, the car may call a repair shop when it has problems, giving employees there a chance to order necessary parts or set up an appointment. "We think there will be more and more remote controls, or telematic diagnostics," IAV's Greff said. When the diagnostic system detects a fault or notices that a component is drifting beyond tolerances, a signal will be sent to the repair center. Greff added that this technology is beginning to see use on heavy-duty trucks, and he expects the same for passenger cars.



OSEK facilitates modular software usage, providing a standard link to various processors.

"Many RTOS vendors now offer solutions that are extremely efficient in both memory size and execution speed. In addition, some vendors have developed unique licensing agreements, making commercial RTOS systems very attractive for production powertrain applications," said Dan Presidio, Visteon's Manager of control electronics/software development.

However, most observers note that the move is being made judiciously. "It's less than a quarter commercial at present across all the OEMs in the world. Significantly less," Ramaswamy said.

Automotive firms that have done their own software for years are leery of turning the technology over to an outside firm. One key question is how much access to source code the RTOS vendor is willing to provide. Other concerns include support, reliability, and the stability of the vendor. The latter point is especially true in the real-time operating system world, where many of the suppliers are fairly small companies.

In yet another move that will simplify software development, automakers are developing software layers that sit between the engine and programs written by third-party providers. Instead of looking at specifications that will vary from project to project, this corporate specification provides a single approach to some of the common tasks of engine control.

"We have a piece of software we call the hardware I/O; others call it a BIOS," Bogden said. "We tell our development companies when you're reading, say, the MAP (manifold absolute pressure), deliver the information to us in this format."

As in other fields of computers and software, these interfaces give programmers a more uniform format than when they needed to directly address the system being controlled. Although each OEM creates its own specification, the layered approach greatly simplifies the task of writing code.

"We don't worry about the internals of the engine anymore; the interface specification becomes the important part," Wiesenberger said. ***aei***