Proceedings of the 1996 IEEE International Symposium
on Computer-Aided Control System Design
Dearborn, MI • September 15-18, 1996

Development of Engine Control System using Real Time Simulator

Akihiro Kimura, Iwao Maeda

Engine Engineering Div. II Component & System Development Center, Future Project Div. No. 1
Toyota Motor Corporation

1200 Mishuku Susono Shizuoka 410-11 Japan

Abstract

In automotive manufacturing, the use of precise control is increasing to satisfy customers' needs, such as fuel economy, air quality. To provide sufficient utility to customers, engine control systems are becoming complicated. For this reason, engine control system development requires long periods and many efforts.

We applied two types of real time simulators to the development of an engine control system. One is the engine and vehicle simulator which receives the actuator signals, calculates engine operating conditions such as engine output torque, air fuel ratio, engine speed and vehicle behavior, such as vehicle speed, reaction force and outputs sensor signals. The other is the control logic simulator substituting for a part of the engine control logic on production CPUs.

An application of fuel injection control system was conducted and the simulators have the potential to efficiently improve the development process of engine control systems.

1. Introduction

In engine systems, the engine power and exhaust gas emissions are greatly dependent on the accuracy of engine control. To satisfy customers' needs and to meet strict exhaust gas emission standards, engine control systems have become so precise and complicated that long periods and many engineers are necessary to ensure sufficient quality and reliability. In addition, the complexity of control systems makes it difficult to change the control logic because the side effects of revisions are difficult to identify through experimental and empirical methods.

Computer simulation technologies are strongly expected to lead the rapid development and high quality prototyping. For control logic developments, it is necessary to know characteristics of the controlled objects. The control logic can not be fixed until the hardware is completed. However, all parts of the engine system may not be completed for the development because it is difficult to provide hardware prototypes timely. This situation will not be improved in near future, although a number of efforts have been done.

It has been proposed that incomplete parts could be replaced by real time simulators. This method is called Hardware In the Loop Simulation (HILS). In this study, two types of real time simulators have been developed. One is the engine/vehicle simulator which we call "Virtual Engine and Vehicle" (Virtual Engine). And the other is the control logic simulator substituted for a part of the engine control logic on the electronic control unit (ECU). We call this the "Rapid Proto ECU" (RPE).

Many HILSs have been reported in the last 10 years. We also developed HILSs in the past but they have not been propagated among engine control system development engineers. The major reason for this is lack of flexibility. It was difficult to adapt HILSs to new hardware systems because the programs were written by FORTRAN or C language. To solve the problem above, we have adopted general purpose tools with graphical user interface (GUI) and C code generation to easily install required simulation models and engine control logic on the real time simulators.

2. Virtual Engine and Vehicle

Figure 1 shows the schematics of the Virtual Engine. The Virtual Engine is connected to the ECU to estimate closed loop characteristics. It receives the control signals from the ECU and returns the sensor output signals. The input/output signals of the Virtual Engine which are

0-7803-3032-3/96/$5.00 © 1996 IEEE

also the corresponding output/input of the ECU are listed in Table 1. The engine simulation model has been developed using SIMULINK, a general purpose modeling tool with GUI [1]. The model is transformed to the C code by Real-Time Workshop [2] and implemented on DSP-CIT composed of the high speed calculation board and I/O boards [3].

The Virtual Engine can be used for the following purposes in the engine control system development.

- simultaneous development of hardware/software
- evaluation of control algorithm
- debugging ECU hardware/software
- reappearance of problems in market
- calibration

The Virtual Engine makes it easy to analyze complicated interference caused by many interrupts among the tasks. In addition, it can reduce the high cost of experiments and repeat the same operating conditions, for example cold start and warm up. Moreover, the Virtual Engine can estimate the behavior under even unrealistic conditions of outputs from the ECU, so that the ECU bugs appear clearly.

In developing Virtual Engine we considerd following I/O performance. The voltage level of battery is 0 to 20 volts. So, we dropped down the voltage level by the additional voltage divider in order to be accepted by the AD converter. We used AD and DA converters with a resolution of 20 volts per 12 bits because AD converters installed on the ECU have a resolution of 5 volt per 10 bits. To detect the pulse width of input signals, such as fuel injection and idling speed control signals, the Virtual Engine must measure the time when a signal crosses the threshold level. The required resolution is less than 4 micro seconds, which is the resolution of signal generation of the ECU. We used a wave form capture board with 25 nano seconds resolution to meet our requirements. In engine controls, the Virtual Engine should simulate two crank angle sensor outputs. The former pulses at every 10 degrees of crank angle except for a teeth missing angle. The latter pulses once every two revolution of crank. The required resolution of the output pulse is less than 4 micro seconds which is the resolution of signal capture of the ECU. We used an additional board to generate the pulse outputs which has 6 DSPs (Texas Instrument C31) attached to 6 DA converters, respectively. However , the programs on C31s check the crank angle and output the voltage only every 8 micro seconds. In another words, we could not achieve the required resolution. We need pulse generators with higher resolution.

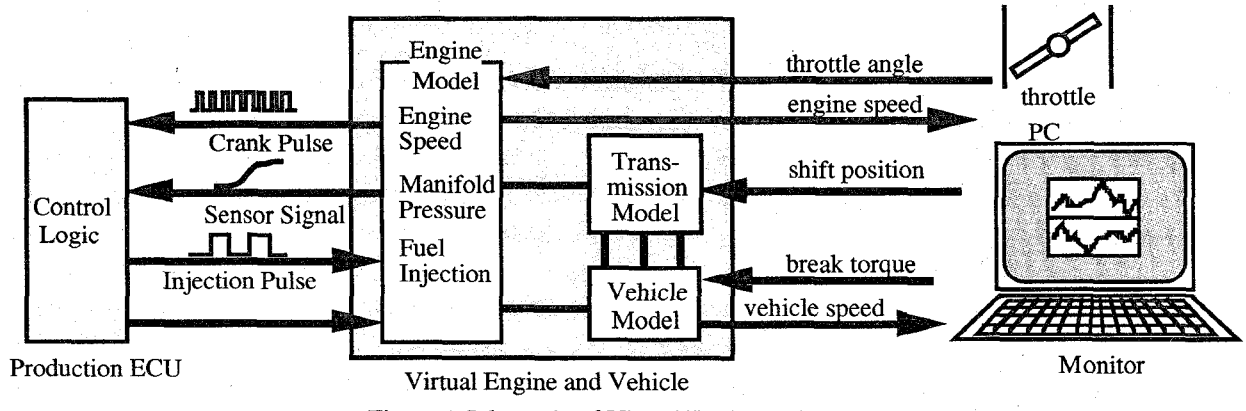

Figure 1 Schematic of Virtual Engine and Vehicle

Table 1 Input and output signals for Virtual Engine

Virtual Engine input signals	type	Virtual Engine output sensor signals	type
Acceleration pedal position	voltage	Manifold absolute pressure	voltage
Engine start key switch	voltage	Engine coolant temperature	voltage
Shift lever position	voltage	Atmospheric temperature	voltage
Exhaust gas recirculation	voltage	Air fuel ratio	voltage
Fuel injection	pulse width	Engine crank angle	pulse edge timing
Ignition	pulse edge	Vehicle speed	pulse edge timing
Idling speed control	pulse width		

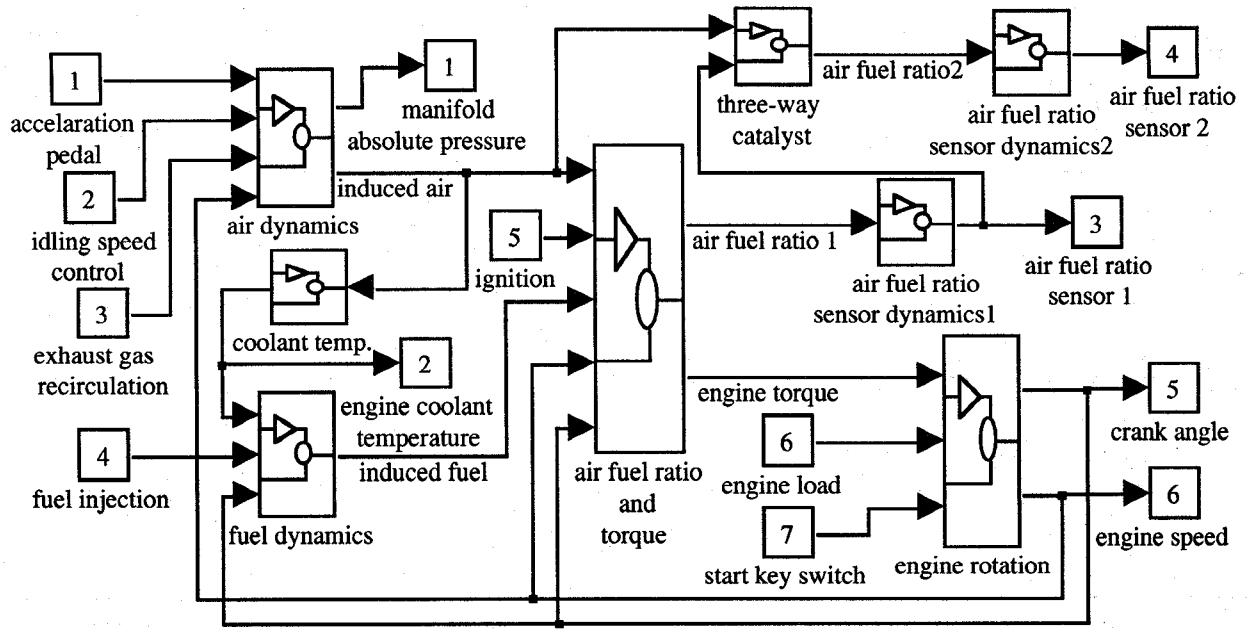

Figure 2 Engine Model described with SIMULINK blocks

The Virtual Engine is composed of the following models as shown in Figure 2.

- Intake air flow model
- Fuel behavior model
- Engine rotation model
- Model for calculating air fuel ratio and engine torque
- Air fuel ratio sensor model
- Three-way catalyst model (only calculating oxygen molecular behavior)
- Model for calculating engine coolant temperature

The simulation programs were written with SIMULINK. The SIMULINK diagram of the engine model was transformed into the C code automatically in order to perform the simulation on the Texas Instrument DSP C40. The C40 can calculate 1 milli second engine behavior within about 800 micro seconds.

We encountered some difficulties in developing the simulation program. Intake air and fuel are induced into the cylinder intermittently and engine torque varies at every spark ignition. SIMULINK does not provide block which has a function to represent such intermittent behavior. So, we must use S-Functions and MATLAB Functions written as M-files, which are often needed to describe complex functions. The Real -Time Workshop, however, does not transfer the M-files into the C code, so we coded them by C language.

The simulation program described with the block diagram is easy to understand and revise. Actually, we have developed the Virtual Engine in only one month. Only a few software bugs appeared in the parts written by blocks supported by SIMULINK, while a large number of bugs occurred in the parts written by C code. This indicates that the block diagram approach can be very helpful to develop control software and maintain the simulation according to the development stage.

3. Application of Virtual Engine

We applied the Virtual Engine to the fuel injection control development. In order to reduce exhaust gas emissions and attain good drivability, it is very important to control cylinder gas mixture for good combustion, especially from engine start to warm up. In fuel injection engines, injected fuel adheres on intake ports and cylinder walls. The amount of the wetting fuel mass varies with engine operating condition and this fuel flow delay makes it difficult to achieve accurate air-fuel ratio control.

Figures 3 and 4 show simulation results of engine behavior during warming up (coolant temperature is 25 ℃) just after a driver attempts to accelerate the vehicle rapidly. Figure 3 shows the results at the beginning of the parameter study to calibrate the fuel injection control. In this case, the engine hesitated and stumbled when the throttle valve was rapidly opened. This means that the ECU had inadequate control parameters. Figure 4 shows the results after the parameter study was completed. In this figure the engine speed smoothly rises up which indicates good drivability.

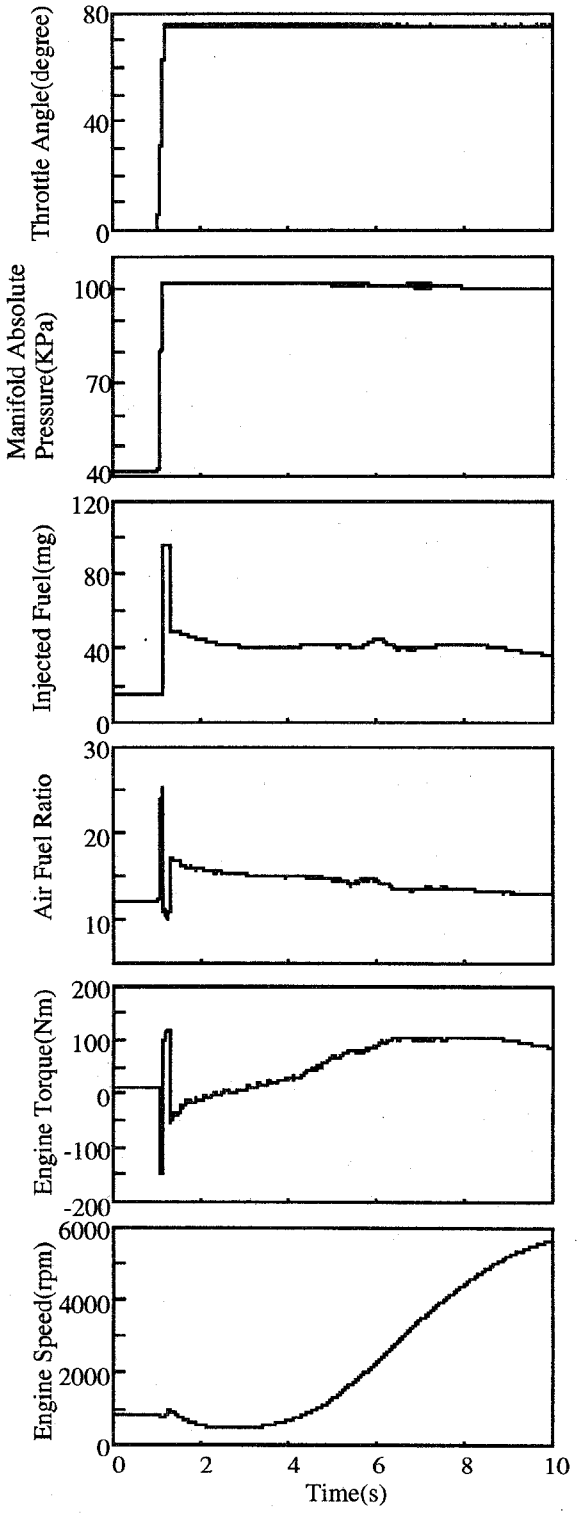

Figure 3 Engine behavior before parameter study

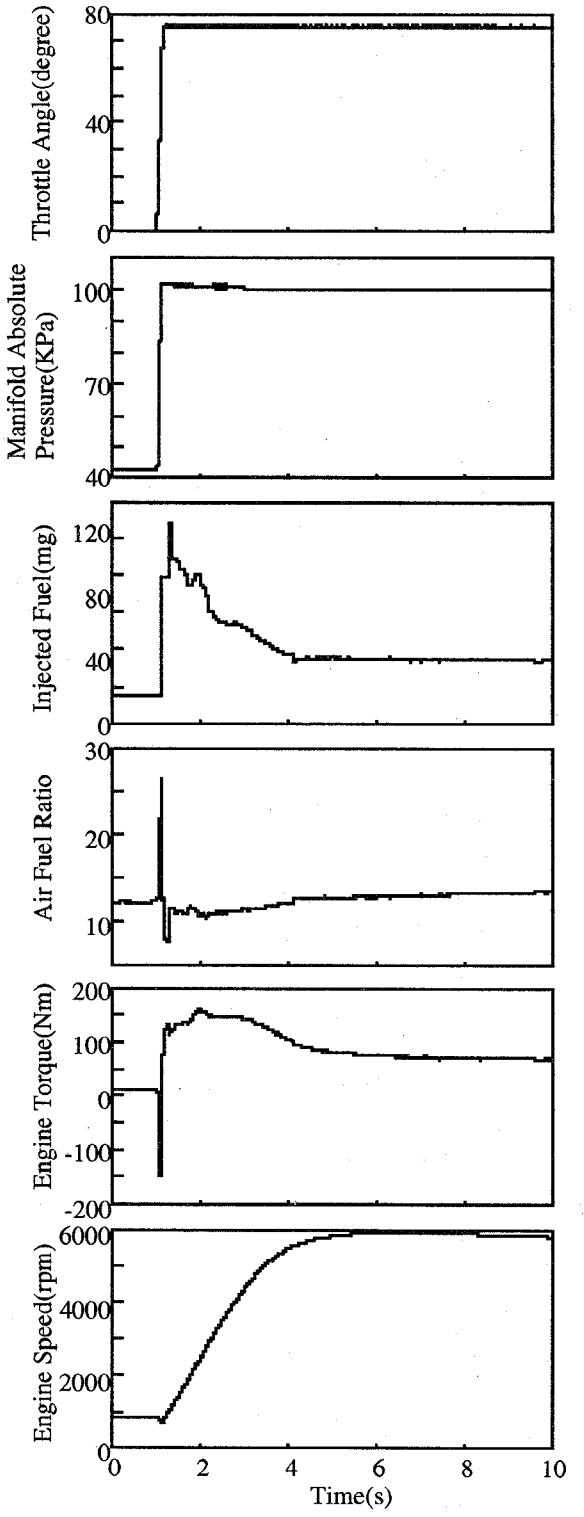

Figure 4 Engine behavior under suitable parameter

The Virtual Engine has limited capability for precise parameter tuning at this time, but it can be useful to investigate fundamental problems in control logic and roughly calibrate control parameters. The Virtual Engine will play an important role for vehicle control system development in the near future.

4. Characteristics of Production ECUs

ECUs installed on cars must have characteristics different from these used in general purpose computers or controllers.

First, they must ensure high reliability. For safety reasons, high reliability is required because even a small error of the engine control could cause an unexpected

accident. Therefore, engineers must take care so that nothing escapes their eyes.

The second requirement is to provide high quality vehicles to customers at low cost. This limits the performance and memory size of the CPU. Currently, fixed point CPUs with 8 bits and/or 16 bits are used by almost all automotive manufactures. It can be said the margins for calculation time and memory are small.

The third characteristic is the ability to handle frequent interrupts. These include internal timer, I/O and software interrupts. The internal timer interrupts schedule tasks. They include decision of actuating time and measurement of input signals, for example , air flow rate, coolant temperature and throttle angle. The I/O interrupts are used to get engine speed from crank pulses, to control fuel injection quantities, spark advance timing and various engine device actuators. Engine control logic is composed of various event driven tasks.

In order to develop such a highly reliable and complicated software, the period and cost for the development are rapidly increasing.

5. Rapid Proto ECU

The RPE is a support tool to control an actual engine and vehicle through communication with the production CPU. The purpose of the RPE is to calibrate the control parameters and evaluate the designed control logic efficiently and rapidly. The following advantages are expected for the RPE.

- reduce experiments
- speed up test cycles
- generate production C codes rapidly

In this system, the control logic can be designed on MATLAB and evaluated by simulation on SIMULINK to screen specifications of control logic in advance. This effectively reduces the number of experiments. Candidates are transformed to the C codes by Real-Time Workshop and transmitted to the RPE. In this way, the evaluation of how the new design controls actual engine can be done immediately. Although this C code (which use floating point CPUs) cannot be applied to current production CPUs because they use fixed point processors, the development period is expected to be reduced considerably.

Figure 5 shows the schematic of the RPE. We used a high speed controller board (DSP-CIT) with the same C40 as in the Virtual Engine. Only specific parts those affected by new design changes of engine control logic run on the C40 and the other parts run on the production CPU. The C40 communicates with the production CPU through dual-port memories. The developed logic on the C40 starts after the production CPU writes the required data on dual-port memories and outputs the interrupt request to the C40. The production CPU is waiting until C40 returns the calculation result. This process is shown in Figure 6. The advantage of this method is that it is able to concentrate the effort on the specific or target parts (in program B), while the other parts (in program A) can be used without changing the production CPU.

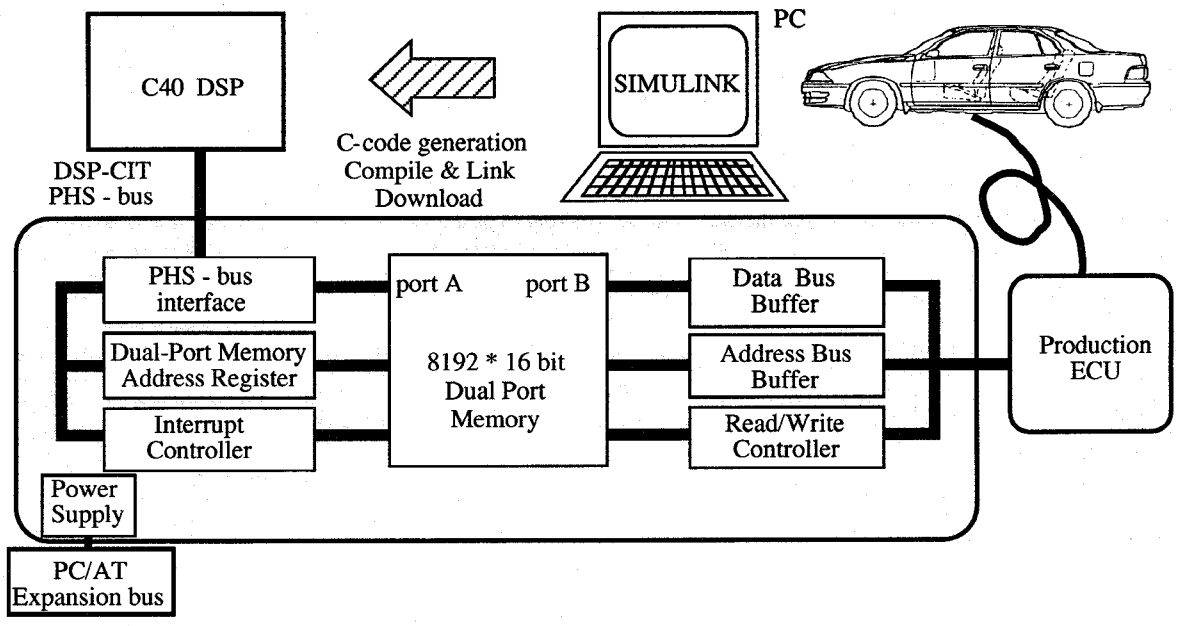

Figure 5 Schematic of Rapid Proto ECU

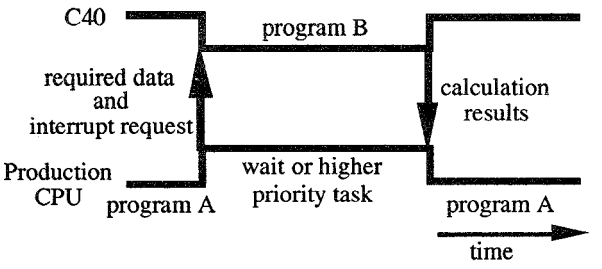

Figure 6 Calculation flow on Rapid Proto ECU

6. Engine control experiment

We applied the RPE to the fuel injection control for a spark ignition engine. Improvement of fuel injection is the most important factor for purifying the exhaust gas emissions and realizing good drivability. The fuel injection timing is decided according to the engine operating condition. The fuel injection must be finished by the middle of the intake stroke in order to obtain good combustion. Therefore, the start timing of the fuel injection depends on the engine speed and the amount of fuel which is injected. The calculation is strictly scheduled in the engine control. Therefore calculation timing is checked every 30 degrees crank angle.

The logic calculates compensation of fuel wetting described in Section 3. We made the logic with SIMULINK blocks in less than 10 days, which is almost 3 times shorter than programming on the production CPU

Figure 7 shows the comparison of CPU time between the hand coded C program on production CPU and the generated C code on RPE. The comparison shows that the RPE can calculate the logic faster than the production CPU when it is injection timing However RPE executes this logic at every 30 degrees crank angle. This frequent execution is caused by the specification of SIMULINK switch block. The switch block selects one input after all input values are decided. This indicates that SIMULINK has a poor capability of control flow description and the functionality should be enhanced so that RPE works more efficiently.

Figure 8 shows the comparison of the compensating amount of fuel injection mass calculated by the production CPU and the RPE. A slight numerical difference can be seen, although the characteristics of both are quite similar with each other. The reasons for this are probably the floating point calculation on the RPE and the difference of handling interrupts on the RPE.

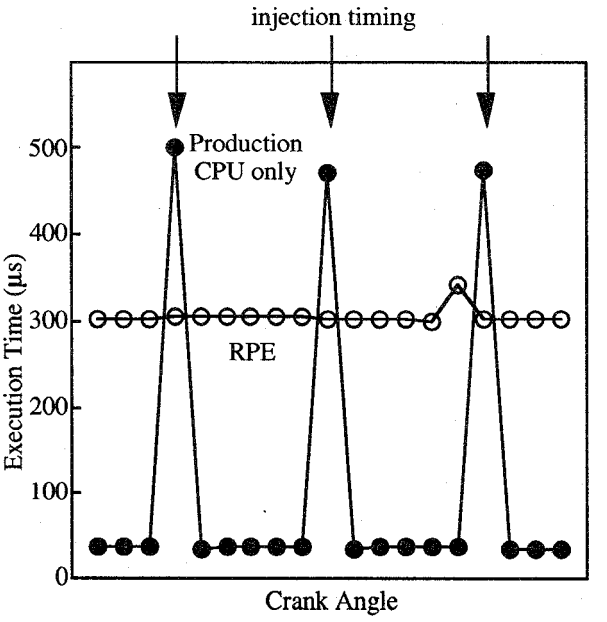

Figure 7 Execution time comparison

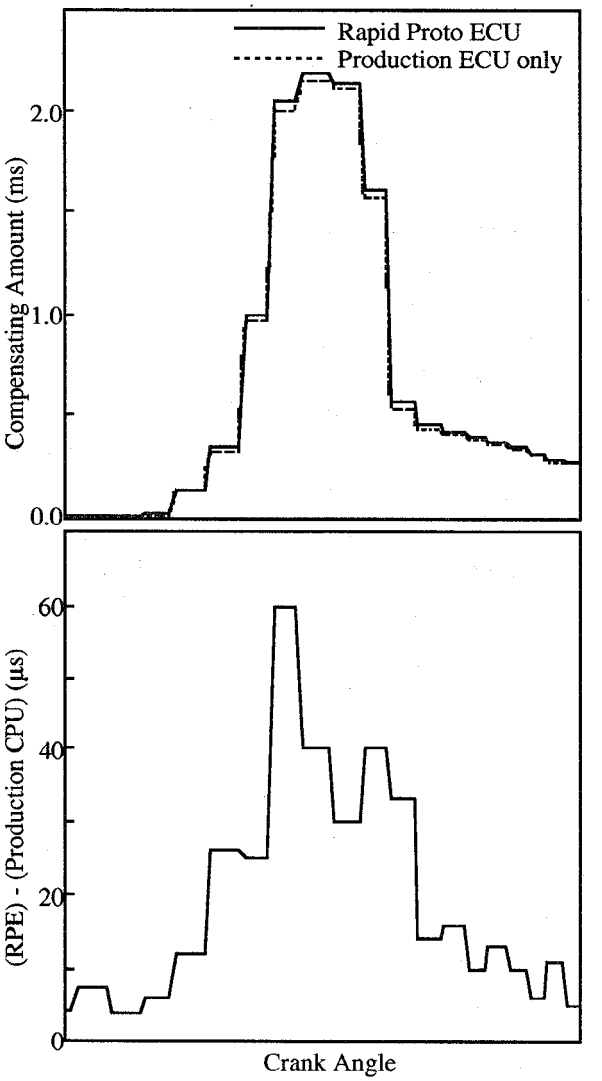

Figure 8 Calculated data comparison

7. Summary

Two types of real time simulators have been studied to develop an engine controller more efficiently. One is the engine and vehicle simulator which can evaluate closed loop behavior of the engine by connecting an actual engine controller. Through the calibration of parameters relating to fuel injection control during engine start and warm up, we have seen the possibility of the simulator to dramatically improve the process for engine control system developments. If the accuracy of the engine model is improved, it could be used to calibrate parameters precisely. The other simulator substitutes the calculation on high speed CPU for a part of engine control logic on a production CPU. It is useful to speed up test cycles to improve control logic. Both systems are supported by graphical user interface to describe engine models and control logic as block diagrams. It is helpful to understand and revise them. The simulators have been developed in short time with the diagram approach and general purpose boards, which include AD and DA converters and wave form generator and capture .

However, our system required some specialized software and hardware tools. In particular, we had to add functions for event triggered logic and treatments of multiple interrupts. And we desire a transforming system which can generate suitable logic for our production type ECU directly form the block diagram. We also need to support specific sensor and signal simulators with high resolution for the virtual engine. In the future, we hope to use more generalized tools to handle these specialized functions.

8. References

[1] The Math Works, Inc., "SIMULINK, Dynamic System Simulation Soft ware, User's Guide" The Math Works Inc. June 1995

[2] The Math Works, Inc., "Real-time Workshop, For Use with SIMULINK User's Guide" The Math Works Inc. January 1995

[3] dSPACE, "RTI40, Real-Time Interface to SIMULINK, User's Guide" dSPACE digital signal processing and control engineering GmbH

9. Trademarks