

Partitioned Control Challenge Problem

Introduction

The lack of model-based tools to analyze and implement the distribution of software functionality between multiple targets is a problem faced in the automotive industry today. Typically the problem is addressed by implementing and analyzing potential solutions with hardware and software prototypes. This trial and error process is both costly and time-consuming. The challenge posed to the MoBIES researchers is to develop model-based tools and methods that enable engineers to effectively analyze the partitioning and distribution of functionality to multiple targets in a virtual manner, without requiring hardware and software prototypes.

The partitioned control challenge problem is described in terms of an Electronic Throttle Control (ETC). The ETC application is distributed between two processors (MPC555 and HC08) on the powertrain Open Experimental Platform (OEP). The ETC application implemented on the powertrain OEP provides a realistic framework for describing a distributed control problem. Although the ETC application involves a small-scale distributed platform, the resulting tool solutions should be applicable to vehicle-wide distributed control problems. The focus of the challenge problem is on tools and methods, not a particular application.

The partitioned control challenge problem falls under the category; Allocation of System Function and Performance to Distributed Computing Environments. However, many elements of the problem correspond to other challenge problem categories described in the [lead document](#).

Electronic Throttle Control

ETC replaces a mechanical system consisting of a linkage between the gas pedal to the throttle plate. In the mechanical system the vehicle operator directly regulates the engine airflow by adjusting the position of the throttle plate via the gas pedal. At idle speed conditions, the airflow bypasses the throttle plate and is regulated with an Idle Air Control (IAC) valve.

In the ETC system the throttle plate is actuated electronically. The desired throttle plate position (setpoint) is determined based on the pedal position as well as other inputs and operating conditions. A primary benefit of ETC is it enables system designers to incorporate throttle control into other automotive functions, such as cruise control and vehicle stability control. ETC is considered a safety critical system. As a result a considerable portion of ETC functionality is in place for redundancy and safety monitoring.

For the powertrain OEP, an MPC555 32-bit floating point processor with an OSEK compliant operating system is used for engine and transmission control. In order to implement ETC with redundancy, the MPC555 is interfaced with an HC08. The HC08 is an 8-bit fixed point processor that, for this application, uses a simple non-preemptive scheduler. In addition to providing redundancy for ETC, the HC08 with simple scheduler provides a low -end target to evaluate the scalability of tools for target specific analysis (schedulability/throughput, memory resource) and automatic code generation.

For the MoBIES project the ETC functionality will be limited to simple pedal following control with redundancy and safety monitoring. Figure 1 illustrates the basic architecture of the ETC system. The MPC555 and the HC08 redundantly measure the pedal position and calculate the throttle position setpoint. Actuator driver electronics generate clockwise and counter-clockwise PWM signals to control the DC servo motor. The PWM driver signals are fed back to the HC08 and the MPC555 to ensure the driver electronics are functioning properly. The actuator driver electronics supply an overcurrent diagnostic signal, which is redundantly sensed by the MPC555 and the HC08. The output from the throttle plate position sensor is fed back to the MPC555 and the

HC08. For redundancy, both processors calculate a throttle plate output feedback control signal based on the throttle position setpoint and the measured throttle position.

A safety monitor is in place to ensure the ETC system is working properly. It monitors redundant calculations and measurements and, if discrepancies are detected, puts the ETC system into a failsafe mode. The safety monitor inputs come in pairs, one from the MPC555 and the other from the HC08. The monitor executes on either the MPC555 or the HC08 but not on both. The host target for the monitor is a design decision made as part of functional partitioning analysis. Input information is provided to the monitor from its host computer and from the other computer over the serial communication bus, as shown in figure 1.

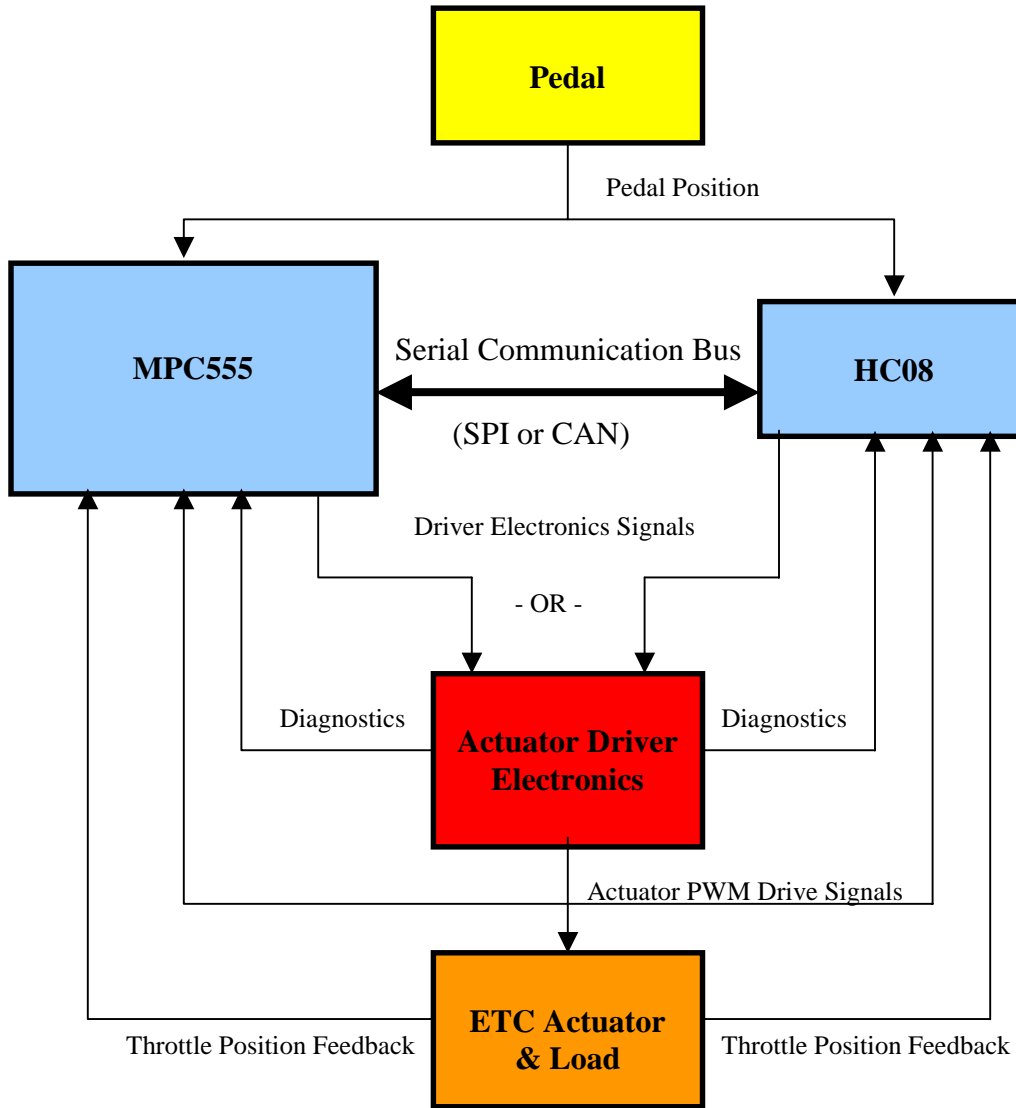


Figure 1 - Powertrain OEP ETC Architecture

Challenge Problem

The partitioned control challenge problem is formulated according to a design process with the desired tool capabilities discussed at each phase. It is assumed UC Berkeley will provide the necessary controller and plant models for the challenge problem.

System Design:

In the system design phase, the basic control functionality of the ETC subsystem is established and the high level interaction between ETC and other vehicle functions is evaluated. The system level analysis primarily involves vehicle and system level simulations with plant models.

In the system design phase the engineer is primarily interested in type 1 models. Type 1 models are continuous time models with no regard for controller sampling. Type 1 models have minimal structure and style.

The type 1 ETC controller models implement only the core ETC functionality, which includes throttle position set-point calculation and throttle plate feedback control. The safety monitoring, diagnostic and redundancy strategies are not included. The type 1 ETC models allow the engineer to concentrate on the basic behavior of the ETC subsystem and evaluate how it interacts with other systems in the vehicle.

A challenge is to develop tools that allow users to select different views (type 1, type2 or type3) of the same model. We envision a "single model" development process with tools that allow engineers to select different views of the ETC models depending upon the analysis to be done.

Another challenge relates to model composability. The ETC subsystem controller and plant models represent a small subset of the models in the system. Tools are needed to automate and manage the composability of the ETC models with other models in the system. Details for the composability problem are given in [model composition](#).

Component Design Phase:

In the component design phase, software specific requirements for the ETC subsystem are gathered and the high-level ETC software architecture is defined. The controller models are transformed from continuous to discrete time. New information is added to the controller models to provide a greater level of detail. Simulations are performed to verify the discretized models maintain the desired system behavior.

Migrating from the system design phase to the component design phase involves transforming the ETC controller models from type 1 to type 2. The steps in this transformation include:

- Partitioning the type 1 models into features that represent logical functional groupings. This is part of the high-level software design process.
- Adding model structure and style according to predefined guidelines. This includes rigorous signal naming and characterizations.
- Discretizing the type 1 models. Explicit timing and order of execution information is added to the models.

Currently the transformation from type 1 models to type 2 models is a manual process. A challenge is to provide tools that automate this transformation with the necessary design information provided by the user.

When partitioning models into features, functionality contained within a single type 1 model may be broken up and contained within several type 2 feature models. The composition and multi-view

model tools should provide traceability between views. Changes to a model in one view should be reflected in all views.

Another challenge is to provide model checking tools that analyze the "soundness" of type 2 models. The model checking requirements are detailed in [\[Butts\]](#).

The ETC controller models provided for the challenge problem are in the form of type 2 models, which are partitioned into three different features:

- Throttle position set-point calculation
- Throttle plate feedback control
- Safety monitor

The safety monitor represents functionality not present in the type 1 model that is added in the component design phase.

Software Design Phase:

The purpose of the software design phase is to establish a low level software design that imparts an efficient implementation of the required functionality. The software design phase involves target specific analysis, such as CPU schedulability, RAM and ROM usage. When distributed control platforms exist, the analysis also includes determining the optimum partitioning of functionality between targets.

In the software design phase the type 2 models are transformed into type 3 models by adding information necessary for code generation. A challenge is to develop tools that automatically convert type 2 models into type 3 models. These tools are most likely part of the automatic code generators. The tools must have a high degree of configurability to allow the user to control attributes like variable names, variable types, function and file partitioning. Details on the configurability requirements are given in [embedded software](#).

The type 3 models contain detailed software information, such as variable type and scaling. The proper selection for the variable type and scaling often depends upon the target platform (8-bit fixed point HC08 vs. 32-bit floating point MPC555). The tools should be able to determine the optimum variable type and scaling information based on defined characteristics of the target platform.

During the software design phase, the partitioning of functionality between multiple targets is analyzed. Given the current tool capabilities, this is most likely achieved through trial and error with hardware and software prototypes. A challenge is to develop tools that enable this functional partitioning analysis to be performed within the modeling environment.

In the ETC application, some of the questions to be analyzed may be:

- Can the HC08 execute all of the required redundant tasks along with the safety monitor task?
- Can additional functionality fit into the HC08, such as adaptive cruise control?
- Can redundant tasks on the HC08 and the MPC555 execute at different rates? How does this effect schedulability, system behavior?
- What type of serial communication bus is optimal, CAN, SPI? Can we use an existing vehicle wide CAN link to implement the HC08 - MPC555 communication?
- How do communication delays effect system behavior?
- Which processor provides the best interface to the actuator driver electronics?

To answer these questions the tools must be able to estimate target specific measures such as schedulability, memory usage, bus traffic and communication delays.

A challenge is to create tools that analyze the schedulability, CPU throughput and memory usage of applications that include algorithms, input/output processing and serial communication. The challenge includes defining the attributes of the target environment that enable this analysis to be done in the modeling domain. The user should be able to configure the tools for a range of targets and operating systems.

For CPU throughput analysis, the tools should be able to estimate the execution time of every task in the modeled application. The tools should provide the ability to account for un-modeled CPU resources, such as the input/output processes and serial communication. The tools should be able to determine the worst-case and nominal schedulability scenarios, as described in [embedded software](#).

For the ETC application, the tools should be able to analyze the MPC555 with a preemptive OSEK OS as well as the HC08 with a non-preemptive scheduler. When analyzing the MPC555 schedulability, the ETC tasks as well as the engine and transmission control tasks must be taken into account.

The tools should be able to estimate RAM, ROM and stack usage given a type 3 controller model and characteristics of the target. The tools should be able to determine the worst-case stack usage scenario.

Another challenge is to provide tools that automatically generate operating systems based on the timing information contained within the models (reference [embedded software](#)). For the ETC application this includes generating a preemptive OSEK OS for the MPC555 as well as a non-preemptive scheduler for the HC08. The tools should generate operating system models as well as code.

The ETC control algorithms are modeled in Simulink/Stateflow. There is a benefit for having an intermediate representation (meta-model) where different modeling and analysis tools can interface. A challenge is to define an intermediate representation that provides an open interface for different modeling, analysis, testing and code generation tools.

The intermediate representation should provide a seamless interface between domain specific modeling and analysis tools. The interface should enable the simulation of complete distributed systems with components modeled in different domains. Simulation tools like Simulink/Stateflow should be able to read the intermediate representation of model's created in other domains and represent the embodied functionality, for example, as s-functions. A tool is needed to manage the distribution of models between domains.

In the ETC application, the serial bus analysis may be best performed in a modeling domain other than Simulink/Stateflow. The tools should provide the ability to perform an accurate simulation of the complete distributed system, taking into account the inherent delays of the communication bus.

Software Implementation Phase:

In the software implementation phase generators convert type 3 controller models into "c" code. The challenge is to provide code generators that beat commercially available code generators in terms of RAM, ROM and CPU.

It may be desirable to generate code from the intermediate representation of models. This provides a generic solution in which code can be generated from any model that can be translated into the intermediate representation.

After software is generated, it is unit tested. A challenge is to provide tools that automatically generate unit test vectors from the models used to generate the code. Details on automated unit test vector generation are given in [embedded software](#).

Challenge Problem Measurement:

The ETC application will be used as a test bed to evaluate the corresponding tools generated by the MoBIES researchers. We will evaluate the tools by applying them to the ETC application as outlined in the challenge problem. We will also compare the results of tools with measurements taken from the ETC application implemented on the OEP.

References:

[Butts] Butts, Kenneth, " Analysis Needs for Automotive Powertrain Control ," Proceedings of The 7th Mechatronics Forum International Conference, September 6 – 8 2000, Atlanta, GA