RAPID PROTOTYPING OF EMBEDDED MICROCONTROLLERS FOR MECHATRONIC SYSTEMS

Submitted to

MECHATRONICS 2000
The 1st IFAC-Conference on Mechatronics Systems
18-20 September 2000
Darmstadt, Germany

by

Ka C. Cheok, N.A. Kheir and G.E. Smid
102 Science & Engineering Building
Department of Electrical & Systems Engineering
Oakland University
Rochester, MI 48309, USA

Phone: 248-370-2217
Fax: 248-370-4633
Cheok@oakland.edu
Kheir@oakland.edu
Smid@oakland.edu

Sunday, November 07, 1999
ABSTRACT

Engineering development of a sophisticated mechatronics system often requires team effort of engineers with control, computer, electronics and mechanical engineering as well as systems engineering management backgrounds. Because of the multidisciplinary nature of mechatronics, it became clear that there is a true need for an engineering tool to systemize the process of the development. In this paper we present one such new tool called ASCET-SD Schematic Development Environment, that allows mechatronics team members to readily communicate with one another through computer simulation, controller emulation, on-line testing and evaluation of embedded controllers. The bottom line is that the integrated environment allows the whole team engineers to readily take hold of the project from concept to realization of the mechatronic system.

1. INTRODUCTION

Today’s fully loaded modern automobile easily carries over 30 items of automotive mechatronic systems [1][2][3]. Their functions are to ensure that the automobile provides high level of ride comfort and road handling performance, along with safety, fuel economy and luxury options. These features are demanded due to stricter environmental and safety standards as well as increasing customer expectations for performance, comfort and convenience.

A list of automotive mechatronic systems is provided here to emphasize the point: Electronic ignition, electronic fuel injection, electronic controlled throttle, emission control, computer-controlled transmission and transaxles, cruise control, anti-lock brakes, traction control, computer-controlled suspension, steering control, body control functions such as power lock, windows, automatic wipers, sunroof and climate control, safety functions such as airbags, security systems, keyless entry system, instrument panel display, stereo system, etc. Some examples of latest automotive innovations about to hit the market are anti-squeeze power window and sunroof, vehicle yaw stability control system, collision warning/avoidance systems, noise and vibration cancellation, anti-roll suspension, hybrid electric vehicles, navigation aids, a built-in automotive personal computer and others. (More)

It is a fact that the automobile industry invests heavily in research to develop these products. It is not surprising to find that several auto companies and suppliers are investigating similar mechatronic products at the same time. Thousands of engineers are employed to work in the area of automotive mechatronics.

2. ENGINEERING ASPECTS IN DEVELOPMENT OF EMBEDDED CONTROL SYSTEMS

A mechatronics system engineering team faces the challenge of managing multidisciplinary engineers with control, electronics, mechanical and computer background. This multi-technology multi-engineering function and team concept is illustrated in Figure 1, as pointed out in Cheok [4]. As complexity of automotive mechatronics grew, there is an undeniable need for computer-
aided engineering tools to assist the engineers in the research, development and engineering process.

To illustrate this point more vividly, we present a chart for the process is shown in Figure 2. As explained by Powers et al [3], a engineering team effort for an automotive mechatronic projects may involve tens or hundreds of engineers, technical and supporting staff with various expertise and backgrounds. The success depends on the systems engineering management of the project.

Embedded microcontrollers are ultimately an essential component of today’s sophisticated mechatronic systems. A modern automobile may utilize over a dozen of distributed microcontroller modules, functioning together with sensors and actuators, to produce performance that is optimum under various driving condition and usage.

The programming phase in the innovation cycle of embedded control systems ultimately involves control engineering and software engineering. More often than not, a control engineer knows a whole lot about the needed control strategy but little about embedded software programming. On the other hand, a software engineer can program very well but do not fully understand the science of systems behavior. The progress from often follows the paths and loops of the chart shown in Figure 2.

An Engineering Challenge in Mechatronic Project. An engineering team of control, hardware and software and test engineers in a mechatronics project faces a wide variety of tasks with separate responsibilities:

- The control engineers research the concept of a mechatronic system, conduct computer studies to evaluate the feasibility and optimize the design for the concept.
- The hardware engineers investigate for appropriate components and builds the physical parts that comprise the mechatronic system
- The software engineers evaluate appropriate computer platforms and program the codes to perform functions for the mechatronic system.
- The test engineers research design experiments for calibrating and evaluating the performance of the mechatronic system.

More often than not, the separate processes result in engineering problems such as miscommunication and misunderstanding of engineering specifications and implementation. A key source to this problem lies in the lack of a common engineering tool that allows every engineers to double check their design as well as the teammate’s. For example, a software code could have a bug (say, unchecked overflow of an integer data) that could not be easily checked by an engineer with control background. An error in control implementation (say, structure of a scheme) may not be coded correctly.

A potential solution to this problem is a common engineering software tool that ties the tasks of each engineer in the team together. The idea is to have a continuous engineering medium that links the separate processes together.

This paper will present an evaluation of one such software called ASCET-SD Schematic Development Environment [5], through implementation of an actual mechatronic project. We will provide an impartial report on its pros and cons. To appreciate and contrast the improvements, we will also present the current conventional method of programming embedded control systems, such as the SH2 C-Program Workbench Development Environment [5].
3. C-PROGRAM DEVELOPMENT ENVIRONMENT

Early embedded software development approach requires assembly language programming with attention to every detail of scheduling, memory layout and operations. For sophisticated application, this could yield large complicated codes.

Progress was made in later approach by using higher level language such as C, to cross-assemble and build an application program. Such approach is still the main method used in the industry and taught at educational institutions. Figure 3 illustrate such an approach using the Hitachi Programming Workbench and the SH2 EVB as an example. Here, a software engineer is required to hand code the main C program (.c and .h files) using compiler such as the GNU or MCS. Part of the codes in the main program may be automatically generated via the “MakeApp” to generate codes for I/O procedures, and/or a schematic capture auto-code generator for time behavior functions. Next, using the HDI, the engineer may then download the object files (.out and .abs files) to the SH2 EVB, to test and debug the program. Using the FDT, the debugged program (.mot file) can be download into the flash memory of the SH2, after which the EVB becomes a stand-alone application.

Although the procedure is straightforward, it is quite tedious. The programming and coding aspect does not reflect the desired functional requirements in an intuitive manner. Testing and debugging thus require tremendous effort and time. It is obvious that improvement in the development process and better supporting tools are a must.

4. SCHEMATIC DEVELOPMENT ENVIRONMENT

Figure 4 shows one such rapid-prototyping tool that allows control, software and test engineers to bring a controller concept to production-level embedded microcontroller codes within a single visual programming development environment. The tool depicted is the ASCET-SD Development Environment, developed by ETAS. It is currently the state-of-the-art embedded programming software for a few selected microcontrollers including the Motorola PowerPC MPC555. The ASCET visual environment integrate the following desirable features for embedded controller development by the team partners:

**Visual Communication and Verification of Requirement and Design**

The schematics and state machine block diagrams graphically depict the desired functions of the controllers. The schematic blocks represents the desired dynamical transient behaviors required of the controller function, and the state machine blocks represents discrete events that trigger the behaviors. These function blocks are interpreted by the software which automatically generate executable software codes for the functions. *Bottom line: The diagrams allow the whole team to readily communicate and verify their requirement and design.*

**Specifications of Target Specific Embedded Microcontroller**

The complex real-time tasks and aspects of combining data acquisition, signal processing, control, scaling, interrupts, on-board diagnosis and networking protocols are addressed automatically by the development software. The software provides specification database on the features of automotive specific target microcontrollers, and function libraries for efficient
implementation of these functions. **Bottom line:** Software implementation aspects can be readily checked by the team and verified with the online target specific database.

**Integrated Experimentation Environment**

The development software includes hardware and software facilities for calibration and measurements that can be used for laboratory testing as well as in-vehicle operation system testing. The hardware interfaces to the target processors and calibration devices, while the software allows the team to execute embedded codes and observe the experiment results from the block diagram environment. **Bottom line:** The team can directly conduct experiments from the block diagram environment, and calibrate and refine their design in a short turn around time frame.

**Document Generator**

Finally, the development software also has a facility to compile a documentation of the block diagrams, auto-generated codes, model simulation, experiment results and physical configurations for the project. The automatic documentation feature ensures that information is accurate and provides ease of communication between development partners as well as customers. **Bottom line:** The team can readily generate a latest documentation of their design, development and test results at the end of an engineering session.

**5. EVALUATION OF EMBEDDED CONTROL SYSTEM DEVELOPMENT ENVIRONMENT**

The objective of this paper is to describe the experience of using the ASCET-SD DE, and evaluate its contribution in streamlining communication among the team partners, ensuring quality embedded codes, reducing the time of testing and thus cost of development. Impressions will be solicited from a control engineer’s view as well as a software engineer’s. We will also investigate what background is necessary to bring the team up to speed in using the ASCET-SD. Comparison will be made between the schematic programming and C-programming approaches.

**References**

Figure 1. Multi-Technology, Multi-Engineering and Systems Engineering Nature of Mechatronics
Figure 2. Control Design, Development and Calibration Process in Mechatronics Engineering
Figure 3. An Example of a C-Programming Development Environment for Hitachi SH2 Microcontrollers. Here a software engineer uses:

1) An auto-code generator such as the “MakeApp” software to generate C-codes for I/O procedures, or a schematic capture auto-code generator for time behavior functions.
2) A compiler (either the GNU or MCS) to hand code the main C program (.c and .h files), include the auto-generated C-codes and build the object files (.out and .abs files).
3) The HDI software to download the object file to the SH2 EVB, and test and debug the program.
4) The FDT software to download the debugged program (.mot file) into the flash memory of the SH2, after which the EVB becomes a stand-alone application.
Figure 4. A Rapid-Prototyping Embedded Microcontroller Development Environment for Motorola PowerPC MPC555. The ASCET-SD is a visual environment that graphically supports all levels of software engineering starting from schematic diagram descriptions of desired functions to implementation of embedded software in the target product.
ETAS

Definition:

**Embedded Software.** Software that takes the specification (I/Os, memory requirement, data type, time behavior, etc.) of the target ECU into account.

**Target Identical Prototyping.** The entire arithmetic and time behavior are specified and simulated on the development system and correspond virtually exactly to target system behavior.
1.0 INTRODUCTION

In July of 1999, Hitachi-America, Inc. provided equipment and funding to Oakland University to investigate, demonstrate and encourage the use of Hitachi SH7055 Microcontroller in robotics and automotive applications. The equipment includes an

- An SH7055 Evaluation Board (EVB)
- SH7055 Programming Software Tools CD (medley of demo software)
- SH2 Programming Workbench

An integrated programming development environment was established on a PC, after numerous email communication and technical assistance from personnel of Hitachi-America and Hitachi-Europe. Work begun in the months of July, August and September to use the SH EVB as a sonar ranger for an on-going collision warning and avoidance system project.

This report describes

1. The software tools that are needed to program the Hitachi SH7055 Microcontroller EVD.
2. The details for installing software and upgrading from demo version to full version.
3. The step-by-step process for using the software to program the microcontroller for a sonar ranging application.