2004-601

A Real-Time Course for Computer Engineers

Anthony Richardson, Dick Blandford

University of Evansville

Abstract

Computer Engineers work in that area that lies between the electronic hardware domain of the Electrical Engineers and the organized formal software systems of the Computer Scientist. Much of this work involves both hardware and software that is synchronized in time to events outside of the system being designed. Such systems are commonly referred to as "real-time systems".

In many computer engineering programs, real-time concepts are sprinkled through several courses, but a formal course that concentrates exclusively on real-time systems is usually an elective. In this paper a real-time course proposal is presented. The course concentrates on real-time concepts central to computer engineering. It is offered as a junior-level course and is required for all computer engineering majors.

The two major objectives while developing the course were: 1) keep hardware and software costs low so that the course can be offered inexpensively and so that students could do course assignments in their home/dorm and, 2) use hardware and software similar to that currently being used in industry. Both of these objectives were achieved by using a standard PC as the hardware platform and by using real-time operating systems and development environments that are available at little or no cost.

A course outline and a list of course resources are presented.

1. Introduction

Real-time programming and embedded systems is a topic which has taken on considerable importance in the computer engineering curriculum over the last 20 years. Many programs treat the subject piecemeal with portions treated in microcontroller, algorithms, and operating systems classes. The course described here attempts to bring all of the concepts related to real-time programming and embedded systems together in one required course in computer engineering.

2. Why a Required course in Real-Time Embedded Systems?

Over the last several years the IEEE and ACM have been working together to produce a computer engineering curriculum description. A paper was presented at the 2002 ASEE annual conference in which core and elective topics were described[3]. While the final details of this body of knowledge describing the computer engineering curriculum are still being discussed, much of the curriculum is in place. Embedded Systems and Operating Systems are included as "core topics" in the Computer Engineering Body of Knowledge (BOK). The BOK suggests that as much as half a course could be devoted to these topics as core material included in every computer engineering program. An expedient way in which this can be done is to require a 3-hour core course in "Embedded Systems and Real Time Programming" as part of the computer engineering curriculum.

Hughes and Nelson[1] define computer engineering as the following:

> "Computer engineering embodies the science and the technology of design, construction, implementation and maintenance of the hardware and the software components of modern computing systems and *computer-controlled equipment*. Computer engineers are solidly grounded in the theories and principles of computing, mathematics and engineering, and apply these theoretical principles to design hardware, software, networks, and *computerized equipment and instruments* to solve technical problems in diverse application domains." (Emphasis added.)

According to this definition, computer engineers are responsible for *computer-controlled equipment*. This single topic distinguishes computer engineering from both computer science and electrical engineering from whence it came. Computer-controlled equipment, with very few exceptions, includes the application of embedded systems and real-time programming. Computer engineers who do not understand the fundamentals of embedded systems and real-time programming will find themselves doing applications that may be better done by an electrical engineer or a computer scientist.

3. Objectives

3.1 Objectives in Creating the Course

In looking at the computer engineering curriculum at the University of Evansville (UE) it seemed that the spring term of the junior year was the best time to offer such a course. This allows students to obtain adequate prerequisite preparation. It also gives them the opportunity to follow-up and complete a major capstone project which includes real-time and embedded systems during their senior year. The objectives in creating the course were:

- Keep hardware and software costs low so that the course can be offered inexpensively and so that students can do course assignments in their home/dorm as well as in a lab.

- Use hardware and software similar to that currently being used in industry.

- Rely heavily on projects as the primary teaching mechanism as opposed to the standard lecture/exam format.

- Leave students with sufficient understanding to be able to design and implement a complex real-time embedded system.

3.2 Course Objectives and Outcomes

With these guidelines in mind, the following course outcomes have been established:

- Students will complete at least three projects involving aspects of a real-time operating system such as semaphores, interrupts, messaging, etc.

- Students will complete one major project which includes a complete real-time system and serves as a summarizing project for the whole course.

- Students will demonstrate the use of modern software development systems, including project management, debugging techniques, and aspects of software design and documentation.

- Students will demonstrate a working knowledge of real-time concepts.

- Students will be able to explain and to demonstrate the hardware and software aspects of interrupt systems.

- Students will be able to demonstrate a clear understanding of real-time kernel architecture.

- Students will be able to demonstrate an understanding of time and task management with regard to real-time embedded systems.

- Students will understand the use of semaphores and mutual exclusion and will demonstrate their use in projects.

- Students will understand the use of messaging systems and other operating system services and will be able to demonstrate such in homework problems and projects.

4. Structure and Prerequisites

EE 458, Real-Time Operating Systems, was taught for the first time during the Spring Semester of 2003. The course was team-taught by the authors. The class met twice a week for a lecture of one hour and fifteen minutes. There were a total of twenty-nine lecture periods during the semester. A syllabus can be found on-line[6].

Prerequisites for the course included the following:

- C programming knowledge – most RTOS applications are written in C with some minimal reliance on assembly language.

- Logic design – A traditional course which includes both combinational and sequential logic design and makes use of programmable logic devices.

- Digital Systems and Microcontrollers – Introduces computer hardware, timers, interrupt systems, C and assembly language. Students typically complete at least one major design project using a microcontroller.

5. Course Topics

Course topics include: real time concepts, interrupt systems hardware and software, real-time operating system principles, kernel structure and architecture, task management, time management, semaphores and synchronization, mutual exclusion, mail boxes and messages, operating systems services, and a major design project

6. Course Outline

Figure 1 depicts an outline of course topics. The following paragraphs expand upon each topic:

- Initially the students are shown how to install and configure the real-time operating system and are introduced to the tools that are used throughout the course. The terms "embedded system" and "real-time system" are defined and examples of each type of system are provided for discussion.

- The next section of the course provides an overview of the development tools and the RTOS. Students are assigned a simple first project that requires the use of the compiler, the make program, and the video display routines provided by the RTOS. This project is intended to (1) refresh C programming skills, (2) introduce the students to make and makefiles, and (3) get the students using the documentation provided in the reference material.

- During the next week and a half the different embedded system software architectures (round-robin, super-loop or foreground/background architectures versus real-time operating systems) are discussed. The advantages and disadvantages of real-time operating systems as compared to general purpose operating systems are presented. Fundamental real-time operating system concepts (interrupt latency, shared resources, non-preemptive and preemptive kernels, task switching, reentrancy, and inter-task communication) are introduced.

- In the next segment of the course (about a week and a half) the structure of a real-time kernel is described in detail. The C code that is used to perform task scheduling and task

switching is examined. The students are assigned their second project at this time. This project requires that they write an application that uses two tasks.

- The next eight weeks of the course looks in detail at specific services provided by real-time operating systems. These topics include: task management, time management, semaphores, mutexes, events, mailboxes, message queues and memory management. Three additional projects are assigned during this period.

- The course concludes by spending a week on RTOS application design principles and in the final week a particular real-time project is examined in detail.

Additional details can be found in the course syllabus[6].

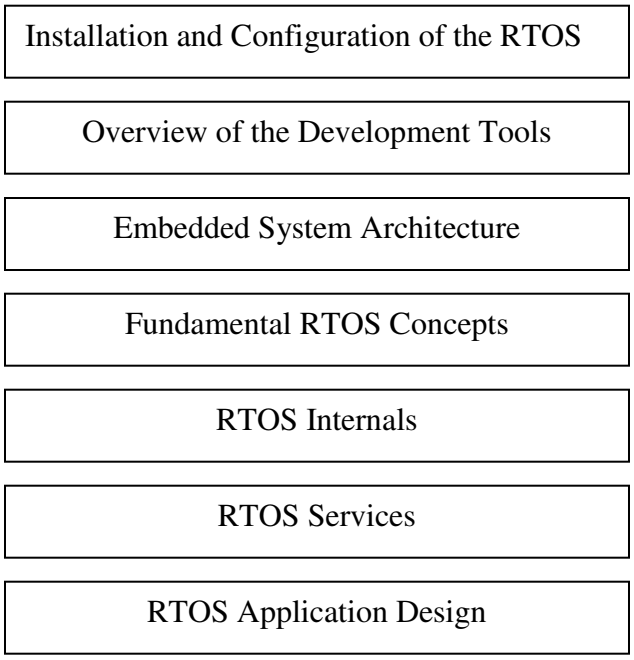

Figure 1: Course Outline

7. Project Assignments

Typical projects include:

- Etch-a-sketch – an introductory project to introduce concepts and get students familiar with the software.

- The elevator project. There is a bank of three elevators which serve a six-story building. Each elevator becomes a task for a real-time operating system. Students can choose to minimize the energy used or they can write the project to minimize the wait time.

- Grain-bin moisture monitoring system. (This project is from real life.) When grain is dried in a bin, it may be stirred while being heated. A sensor detects the moisture of the grain and adds heat as needed to reach the desired moisture content. This example monitors 8 grain bins simultaneously with each being independent of the others. There is a user interface and a shared resource in the form of the A to D converter that reads and averages the moisture sensor.

- Simulated real-time DSP system. This project has a data input array and a data output array. There are four tasks which the system performs: data input, DSP calculation, data output, and variable shifts. These tasks must be synchronized but, with careful programming, they can be overlapped in time to appear simultaneous.

8. Course Resources and Materials

8.1 Required Hardware

Intel compatible personal computers (PCs) were used as the development platform. PCs are available in several computer labs on campus and most of the students have PCs available at home as well. By using standard PCs, the additional hardware cost is zero. (Intel compatible single board development kits are also relatively inexpensive. They are available for under $300 from a number of vendors.)

8.2 Required Software

There are a number of real-time operating systems available for Intel compatible systems. The MicroC/OS-II real-time operating system[5] was selected because it is a full-featured real-time OS and it is available for free for educational use. Except for a few small assembly language routines, MicroC/OS-II is written in C and is very portable. (The web site contains ports of the OS to a number of different development environments and processors.) The reference book for MicroC/OS-II is used as one of the required texts for the course. The book contains the source code for the OS on an accompanying CD-ROM.

The provided MicroC/OS-II code is written for the Borland compiler. The OS was ported to the HI-TECH[3] Pacific C environment because it is available for free and there are no restrictions on its use, i.e. it may be used for educational or commercial projects and is freely redistributable. Porting the code to the HI-TECH Pacific C environment was relatively easy.

The only other piece of software required was a "make" program. The dmake[2] program for DOS works extremely well and is available for free.

With this selection of hardware and software there were no additional startup costs associated with the course. In addition, by using software that may be freely redistributed or that is free for educational use, students can install the software on their home/dorm computers.

This choice in software also offers a lot of flexibility with regard to the host operating system. The software requires only an MSDOS compatible OS and so will run in a DOS window under

any version of Microsoft Windows. The software can also be run under the FreeDOS[2] operating system under the DOSEmu DOS emulator under Linux.
The complete software suite (RTOS, compiler and tools) requires under 4 MB of disk space for installation. This is small enough that University of Evansville students can install the complete suite on their personal network share. They can then work on projects from any network-connected computer on campus. No special permissions are required to run the software or install it to a network share. The suite can be easily downloaded or distributed on two floppies (compressed) or a CD-ROM for installation on home PCs.

The MicroC/OS-II port to the HI-TECH Pacific C compiler can be downloaded from the MicroC/OS-II web site[5] or from the course web site[6]. The compiler and make program can be downloaded from the Internet. Refer to the links section for the appropriate URLs.

8.3 Textbooks

The following texts were used in the course:

- *An Embedded Software Primer*, by David E. Simon. Published by Addison-Wesley[7]
- *MicroC/OS-II The Real-Time Kernel,* by Jean J. Labrosse. Published by CMP Books[4].

The first book discusses real-time operating systems in general. The examples use routines from a number of different real-time operating systems including: MicroC/OS, POSIX, Nucleus, AMX, MultiTask!, and VxWorks. The book also includes a number of end-of-chapter problems suitable for homework assignments. The book contains eleven chapters. All chapters are covered except Chapters 3 and 4 on hardware (the students are expected to know this material from their logic design course) and Chapters 9 (tools) and 10 (debugging techniques).

The second book is the reference manual for the MicroC/OS-II RTOS. The introductory chapters also provide some general information on RTOSes. This book contains 18 chapters. Chapters 1 through 12 and Chapter 18 are covered during this course. Chapters 16 and 17 are reference chapters that students refer to while doing projects throughout the semester. Chapters 13, 14, and 15 which cover porting MicroC/OS-II to different processors, are not covered in the course.

9. Future Directions

The OpenWatcom[8] environment may be used for development instead of the HITECH environment. This is also a free development environment for Windows. It has a much richer IDE than that provided by HITECH.

The eCos[9] real-time operating system may be used in the future instead of MicroC/OS-II. eCos appears to offer more services than MicroC/OS-II and is royalty free for both educational and commercial purposes. There is a reference book[10] available although the provided online documentation appears to be quite comprehensive.

Coverage of either embedded Linux[11] or one of the real-time Linux[11] variant OSes may also be added to the course. This will allow comparison between services offered by two different OSes. There is a reference book[12] available for embedded Linux development also.

A recently published reference book[13] on real-time concepts appears to be a more appropriate course text than the Simon[7] book we are currently using.

10. Summary

EE458, a course in real-time operating systems, was developed in response to the increasing importance of real-time operating systems to the practicing Computer Engineer.

A list of objectives and a list of topics for a comprehensive course in real-time programming are provided. It is possible to offer such a course with little cost using free software and standard Intel compatible personal computers. A complete courseware package containing syllabus and assignments is available for download[6].

11. References

1. Hughes, Joseph L.A., and Nelson, Victor, "Computing Curricula 2001: Computer Engineering", IEEE Computer Society/ACM Computing Curricula – Computer Engineering Task Force, ASEE Annual Conference, June 17, 2002.

2. FreeDOS.org, The dmake program can be downloaded from the FreeDOS web site: http://www.freedos.org. The DJGPP make program may also be used and is available from: http://www.delorie.com/djgpp

3. HI-TECH Software, The HI-TECH site contains the Pacific C compiler and tools. Available at http://www.htsoft.com/products/pacific.html

4. Labrosse, Jean J., *MicroC/OS-II: The Real-Time Kernel*, Second Edition, CMP Books, © 2002

5. Micrium.com, This web site contains general information about the MicroC/OS-II real-time OS. The port to the Pacific C environment can be found here also: http://www.ucos-ii.com

6. Richardson, Anthony, and Blandford, Dick, The course web site contains the syllabus, homework assignments, and project assignments: http://csserver.evansville.edu/~richardson/courseware/EE458/

7. Simon, David E., *An Embedded Software Primer*, Addison-Wesley, © 1999

8. The OpenWatcom C/C++ development environment is available from http://www.openwatcom.com/.

9. The eCos real-time OS and documentation may be downloaded from http://sources.redhat.com/ecos/.

10. Massa, Anthony J, *Embedded Software Development with eCos*, Prentice-Hall, © 2003.

11. The Linux Devices web site (http://www.linuxdevices.com/) provides an overview of and links to several embedded and real-time Linux projects.

12. Yaghmour, Karim, *Building Embedded Linux Systems*, O'Reilly, © 2003

13. Li, Qing and Yao, Caroline, *Real-Time Concepts for Embedded Systems*, CMP Books, © 2003.

12. Author Biographies

ANTHONY M. RICHARDSON obtained B.S., M.S. and Ph.D degrees in Electrical Engineering from the University of Kentucky, Syracuse University and Duke University respectively. He is an Assistant Professor in the Electrical Engineering department at the University of Evansville.

DICK K. BLANDFORD is the Chair of the Electrical Engineering and Computer Science Department at the University of Evansville.