EE458 - Embedded Systems
Lecture 1 - Introduction

• Outline
  - Embedded Systems
  - Embedded OS vs Superloop
  - Real-time Operating Systems
  - Embedded Systems Development
  - The RTEMS Real-Time OS
  - The RTEMS Development Environment
An embedded system is a computing system with tightly coupled hardware and software that performs a dedicated function.

Examples: Printers, Routers, Video Game Systems, Portable Music Players, Satellite Receivers, Personal Video Recorders (Tivo), Automotive Control Systems, Wristwatches,
Lecture 1 - Introduction
Embedded Systems

• Note that embedded systems are *computer systems*. An embedded system uses a microcontroller or microprocessor and is *programmable*. Pure digital logic systems are not embedded systems.

• In contrast to a general purpose computing system, embedded systems are typically programmed by the system developer, not by the user.
Lecture 1 - Introduction
Embedded OS vs Superloop

- The embedded systems you've worked with in EE354 and EE454 are superloop systems in which all of the functions are performed in a single (infinite loop).

- Superloop design is fine for small, simple systems. Larger, more complex applications will typically use an embedded operating system. Embedded operating systems offer many advantages over superloop systems.
Lecture 1 - Introduction
Embedded OS vs Superloop

• Reduced Development Time – The OS provides a library of functions for timing, interprocess communication, I/O, networking, and memory management. Applications can be developed much more quickly.

• Hardware Abstraction – Direct hardware communication is performed by the OS (and in interrupt routines). This simplifies and standardizes hardware communication.
Lecture 1 - Introduction
Embedded OS vs Superloop

- Portability – Porting an application to a new platform (board and/or processor) supported by the OS is much simpler than porting a superloop application.

- Scalability – An OS usually allows an application to scale more easily to increased loads by adding additional hardware resources.
Lecture 1 - Introduction
Embedded OS vs Superloop

• Design and Maintenance – An embedded OS application usually consists of multiple, small tasks. This approach can simplify design, development, debugging, and maintenance. It also facilitates a team design approach.

• Communication Complexity – The OS may provide routines to support networking and I/O. This can simplify application development.
Lecture 1 - Introduction
Embedded OS vs Superloop

• Timing Complexity – Different timing requirements by different hardware components can usually be handled quite easily by using different tasks. Different task priorities can be used to help prevent errors due to communication bursts or timing jitter.

• Possible Disadvantages
  - Learning Curve
  - Increased Hardware Requirements
  - Costs
Lecture 1 - Introduction
Embedded Operating Systems

• An operating system typically provides the following features:
  - A Task Scheduler (multitasking)
  - Task Control (creation, priority assignment)
  - Task Synchronization (semaphores, message queues, signals, etc)
  - Intertask Communication (mailboxes, message queues, pipes, shared memory)
  - Memory Management (allocation, deallocation)
  - Other: I/O Services, Timers, Networking
Both general purpose operating systems (GPOS) like Windows and Linux and real time operating systems (RTOS) like RTEMS are used in embedded systems.

Real time systems respond to external events in a timely fashion. “The timing correctness is at least as important as the functional correctness.” (RTC – Page 13)
Lecture 1 - Introduction
Real Time Operating Systems

• Hard real time systems must meet deadlines with a near zero degree of flexibility. A missed deadline is catastrophic.

• Soft real time systems must meet deadlines but with a degree of flexibility. A missed deadline does not cause system failure.

• Example Real Time Systems: Weapons Defense, Medical Systems, DVD Player.
Lecture 1 - Introduction
Embeddeed Systems Devel.

- Embedded systems typically use a cross-platform development model. The software is developed on the host platform (Windows/Mac/Linux) and run on the target platform.

- A cross-compiler runs on the processor on the host platform and produces executable code that runs on the target processor.
In embedded systems the application and OS are usually compiled and linked into a single executable file. (Implying that the OS source code is available.) Not all OS features are required in every application.

Many embedded systems are diskless. The application code and OS are usually stored in non-volatile memory (PROM, flash).
Lecture 1 - Introduction

RTEMS

• RTEMS is an acronym for the Real-Time Executive for Multiprocessor Systems. RTEMS was developed by On-Line Applications Research Corporation (OAR) for the U.S. Army Missile Command.

• RTEMS is released under a modified version of the Gnu Public License (GPL). It is open-source and there are no licensing fees.
Lecture 1 - Introduction
RTEMS

- The RTEMS Web Site
  - http://www.rtems.org

- After installing RTEMS, you can find the complete documentation under:
  c:/opt/rtems/docs/4.11.3
Any UNIX or Windows/MSYS (or Windows/cygwin) system may be used as the host development platform.

RTEMS is available for multiple processors: Motorola MC68xxx, Motorola ColdFire, ARM, Hitachi H8/300, Hitachi SH, Intel i386, MIPS, PowerPC, SPARC, AMD A29K, Hewlett-Packard PA-RISC, ADI Blackfin, TI C3x/C4x, OpenCores OR32.
For each processor there are typically multiple board support packages (BSPs) available. A BSP provides OS support for a particular developer board. For example, there are more than 20 BSPs for the M68K processor.

RTEMS must be “built” for a particular BSP. Cross-development tools for the processor must be in place to build the OS.
In this class, we will be using Windows/MSYS2 as our host platform and the IA-32 PC (the RTEMS pc686 BSP) as our target processor.

Instructions for installing MSYS2, RTEMS (with the cross-development tools) and VirtualBox can be found on the course website under the Installing Software link.
Since we are targeting the PC the applications we develop should run on any available PC. (Provided there is a boot manager installed capable of booting multiple operating systems.) It will usually be much more convenient to run our RTEMS applications on the VirtualBox PC emulator.
VirtualBox can use disk image files on the host as virtual disk drives. After building our RTEMS applications we will need to transfer the application to the virtual drive.

The **mtools** suite of utilities provide an easy way to copy files to/from the virtual drive from/to the host machine.