SJSU SAN JOSÉ STATE UNIVERSITY

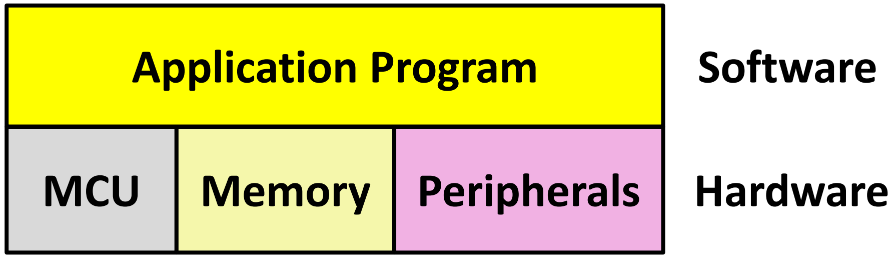
Charles W. Davidson College of Engineering

Department of Computer Engineering

Real-Time Embedded System Co-Design CMPE 146 Section 1 Fall 2024

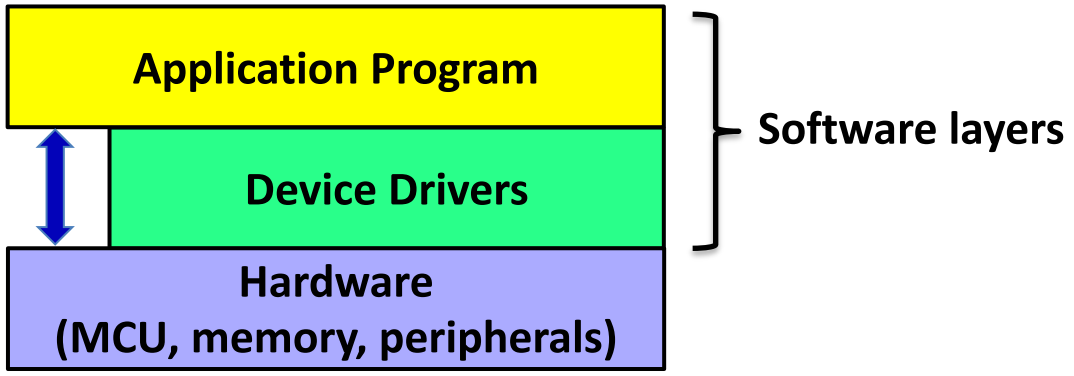
Real-Time Operating System

Software Architecture



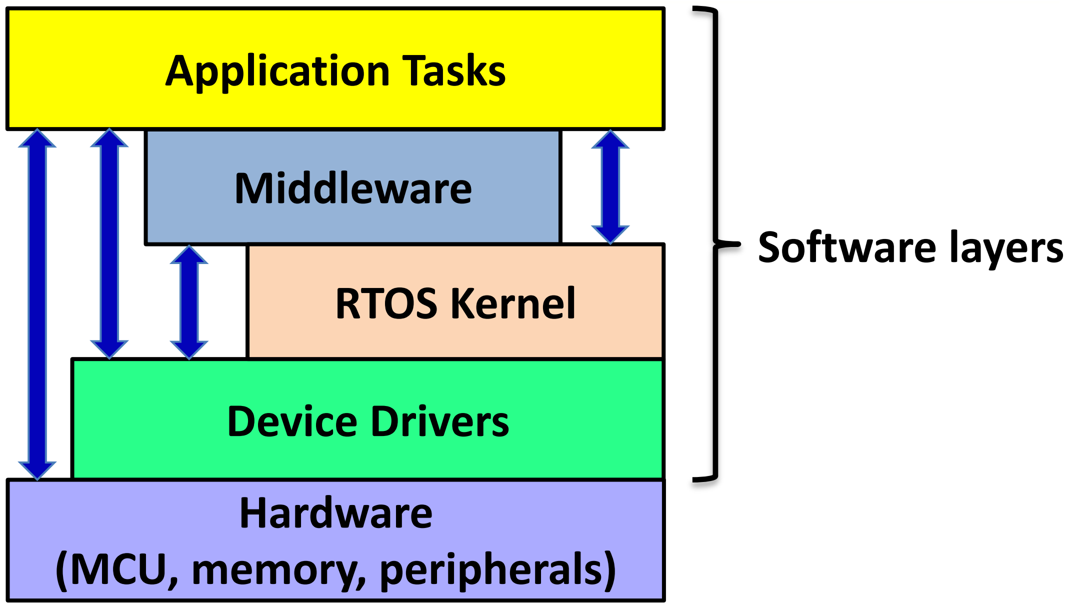
- For simplest applications
- Entire software is customized
- Application program may not even include interrupt handlers

Software Architecture (cont'd-1)



- For small to medium applications
- Device drivers mostly provided by the MCU and board vendors
  - May also include custom-made modules for specialized hardware or features not supported by the vendors
  - Example: TI's DriverLib

Software Architecture (cont'd-2)



- For medium to large applications
- Classic layer architecture
  - The higher a layer goes, the farther from the hardware; hardware becomes more abstract
  - Clear functionality boundaries
- Entire application is constructed as a set of tasks
- Middleware is similar to libraries
  - Contains well-defined functions or services, e.g., network protocols, encryption

SJSU SAN JOSÉ STATE UNIVERSITY

Real-Time Operating System

- For many embedded applications, it is usually worthwhile to adopt a current and well-established real-time operating system (RTOS)
- Programmers can focus on the application software development
  - No need to worry too much about the hardware details
  - No need to design a software infrastructure from scratch
    - Just plug in high-level software components to a well-established and proven software infrastructure
  - Especially in a multi-processor system
    - Handling the hardware complexity can be quite overwhelming
- Advantages
  - Speed up system development
  - Achieve higher system reliability
    - Fewer new components to test and debug
  - Application is more portable to other hardware platforms
  - May achieve better system performance than home-made modules
    - For example, ISRs are already highly optimized by vendor to provide excellent response time

Real-Time Operating System (cont'd)

- Disadvantages
  - Takes up additional system resources
    - Memory space, CPU time
  - May impact performance
  - There is a learning curve
  - May need to pay a licensing fee
    - Add cost to the product
  - Lose the ability to fix bugs in the software components
    - Due to no access to the source code
  - Version upgrade may break the system

Operating System Components

- The central component of an operating system is the kernel
  - Provides services to application tasks to use system resources efficiently
  - Provides protection to tasks
- Main kernel services
  - Task scheduling and management
  - Task synchronization and communication
  - Memory protection (with MMU) and management
  - Interrupt and event handling
  - Device and I/O management
- Operation system can also include additional components to provide other services
  - File system
  - Network protocol handling, e.g., TCP/IP
  - I/O communication protocol handling, e.g., USB
  - Middleware
  - Device drivers
  - Debugging facilities

SJSU SAN JOSÉ STATE UNIVERSITY

Operating Modes

- Under an OS, processor <u>typically</u> executes instructions in either kernel or user mode
  - User mode is a non-privileged mode
    - Certain instructions are not allowed and some registers cannot be accessed
    - All application tasks run in user mode
  - OS runs in kernel mode
- A context switch occurs when processor switches between kernel mode and user mode
  - It takes a finite amount of time (at least tens of clock cycles) to complete
    - The process can be quite involved
    - An overhead that can significantly affect system performance

Kernel Types

There are two broad categories of kernels

- Monolithic kernel
  - All OS components run in kernel mode
  - All OS data structures are in kernel space (protected memory space)
  - No context switch within the OS
- Microkernel
  - Kernel consists of only basic OS building blocks
  - Essential components run in kernel mode
    - For examples, task management and communication, interrupt handlers
  - Other OS components run in user mode
    - For examples, network protocol handlers, middleware
  - Advantages over monolithic kernel
    - Much smaller memory footprint
    - Better protection as user-mode components will not crash the OS core
  - Disadvantage over monolithic kernel
    - Slower if context switch within the OS is required

SJSU SAN JOSÉ STATE UNIVERSITY

RTOS Features

- RTOS provides deterministic response time to external events whereas general-purpose OS does not
  - Constraint of deadline exists in RTOS
  - RTOS uses preemptive priority-based scheduler to ensure meeting soft or hard deadlines
- Short latency in task switching
  - Low overhead in context switch
- Highly efficient ISRs
- Small memory footprint
  - Highly configurable
    - Contains only components that are needed
    - For very limited purposes (unlike a general-purpose OS)
- Inexpensive
  - Low cost for mass-produced products

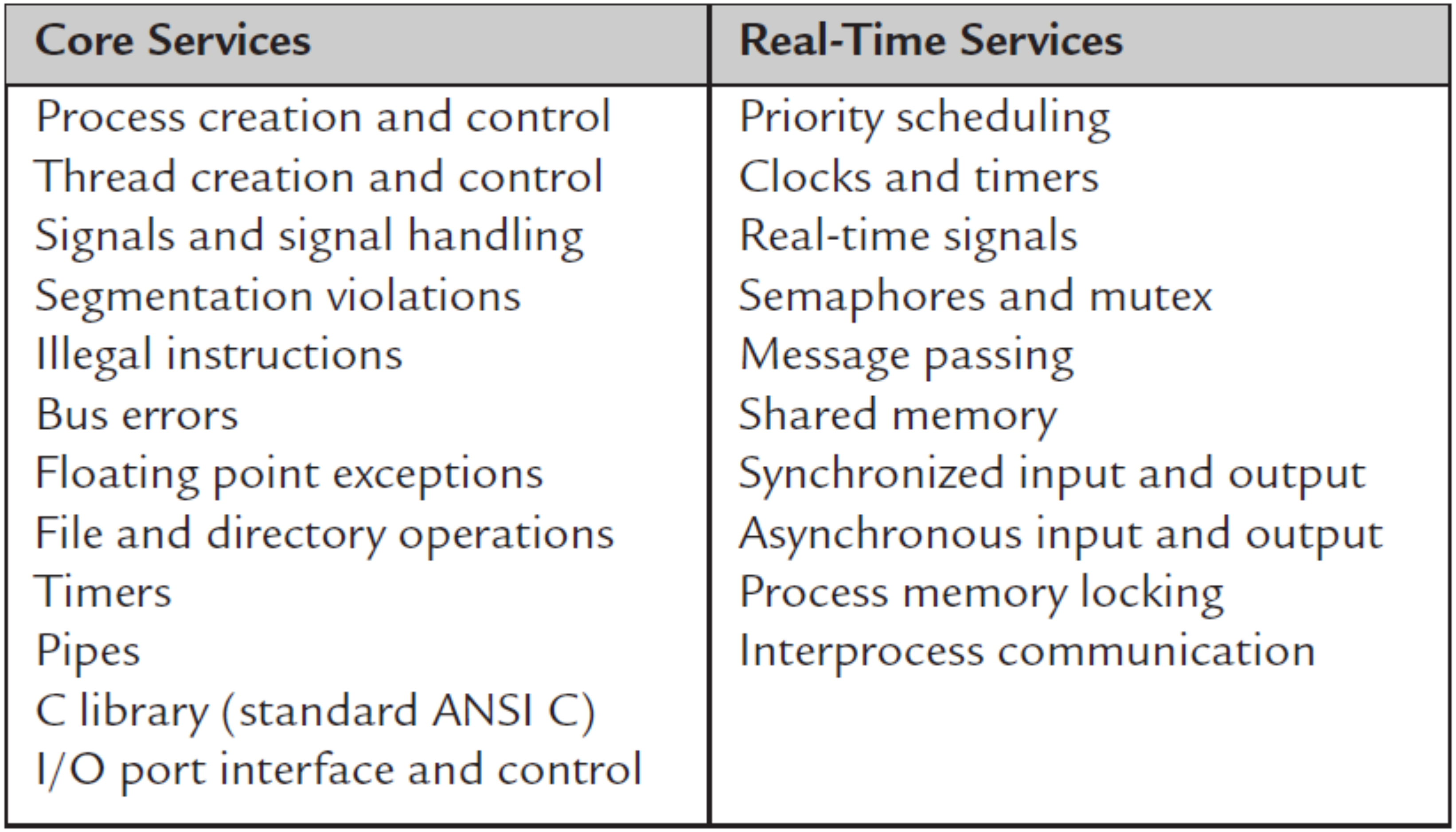
POSIX

- Many RTOSes support a standard operating interface — POSIX (Portable Operation System Interface)
  - Ideally, a program written and compiled for execution on one OS can also be compiled, without changing the source code, for execution on another OS, as long as POSIX is supported by both
- POSIX is an IEEE standard
  - Defines a standard way for an application to interface with the operating system
- In POSIX, each executing instance of a program is called a process
  - Each process has its own protected memory space
- A POSIX thread is a single flow of execution that runs within a process
  - Each thread within a process has its own stack and maintains various processor registers for its own execution stream

POSIX Real-Time Extension

Extension was added to address the needs of improving real-time performance

POSIX core and real-time services



Core Services	Real-Time Services
Process creation and control	Priority scheduling
Thread creation and control	Clocks and timers
Signals and signal handling	Real-time signals
Segmentation violations	Semaphores and mutex
Illegal instructions	Message passing
Bus errors	Shared memory
Floating point exceptions	Synchronized input and output
File and directory operations	Asynchronous input and output
Timers	Process memory locking
Pipes	Interprocess communication
C library (standard ANSI C)	
I/O port interface and control	

SJSU SAN JOSÉ STATE UNIVERSITY

TI-RTOS

Highlights:

- Small footprint
- Multitasking
- Configurable
- Middleware
- Wide range of connectivity support
- Power manager

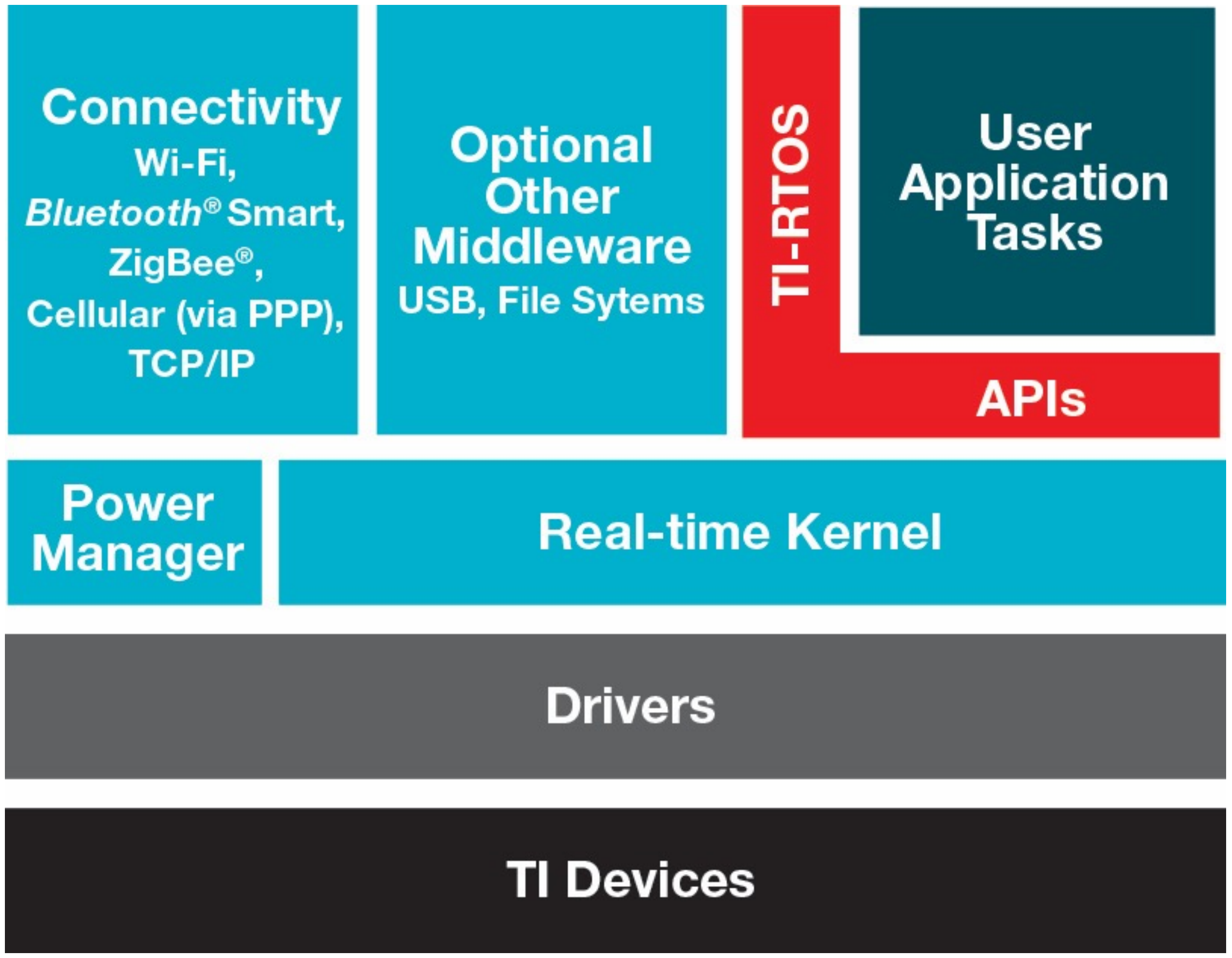


Diagram source: TI-RTOS: A real-time operating system for TI devices (https://www.ti.com/lit/pdf/sprt646)