ECEN 5623

MMIO

MMIO

Interfacing to Off-Chip Devices

Embedded I/O (HW View)

- Analog I/O
  - DAC analog output: servos, motors, heaters, ...
  - ADC analog input: photodiodes, thermistors, ...
- Digital I/O
  - Direct TTL I/O or GPIO
  - Digital Serial (I2C, SPI, ... - Chip-to-Chip)
  - Digital Interface to UART/Line Driver, Serializer-deserializer (Serdes), or Physical I/F
    - Digital I/O to Line Driver Interface (e.g. RS232)
    - 10/100 MII, 1G RGMII, 10G XGMII interfaces to Serdes to Phy

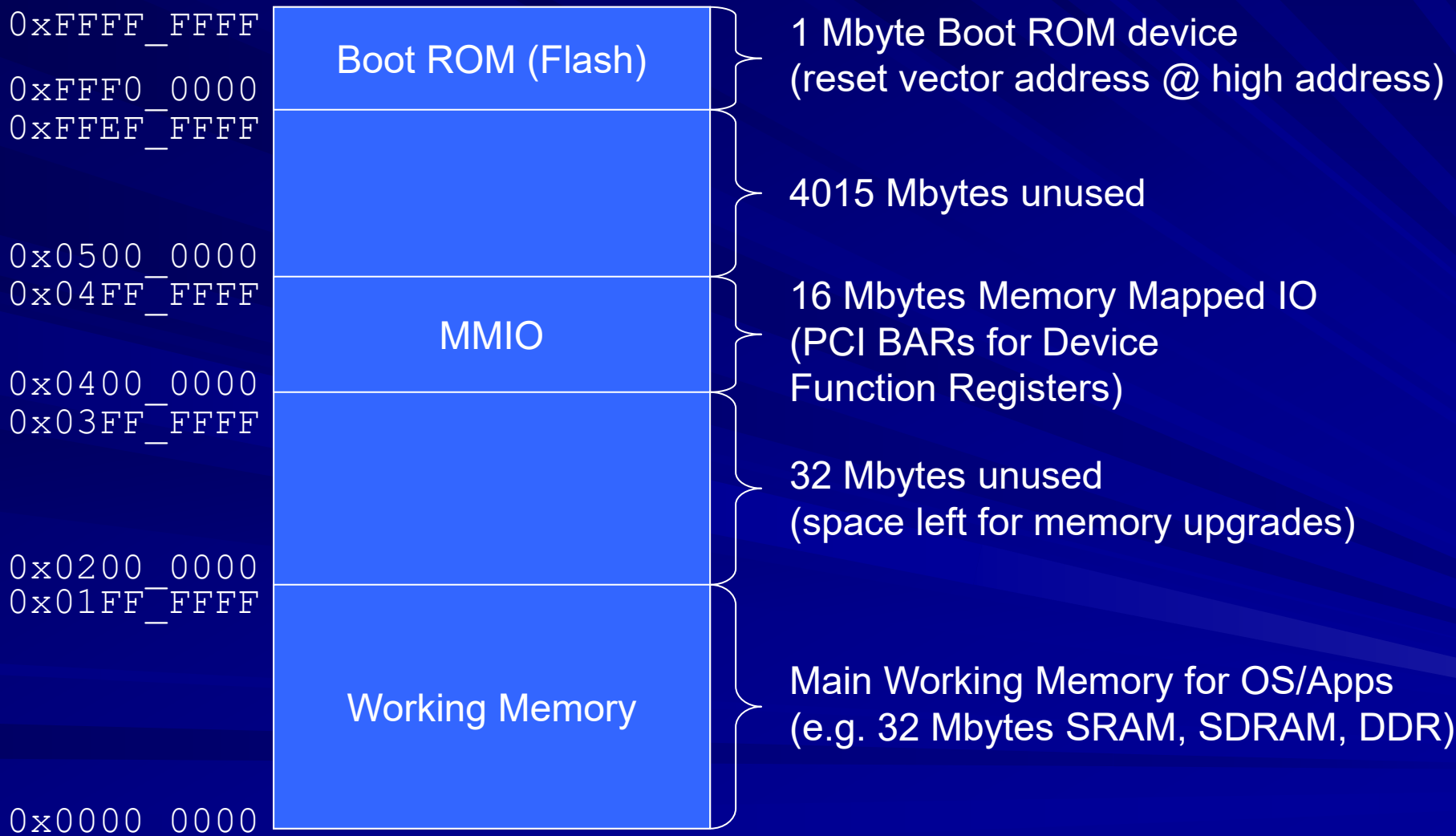
CPU Core I/O (HW View)

- Word
  - Register Control/Config, Status, Data
  - Typical of Low-Rate I/O Interfaces (RS232)
- Block
  - FIFOs (2-32K), Dual-Port RAM and DMA
  - Typical of High-Rate I/O Interfaces
- System Memory Map (32 or 64 bit space)
  - Physical Memory (SDRAM, DDR, SRAM)
  - BootROM (Typically Flash)
  - Configuration and Control Registers

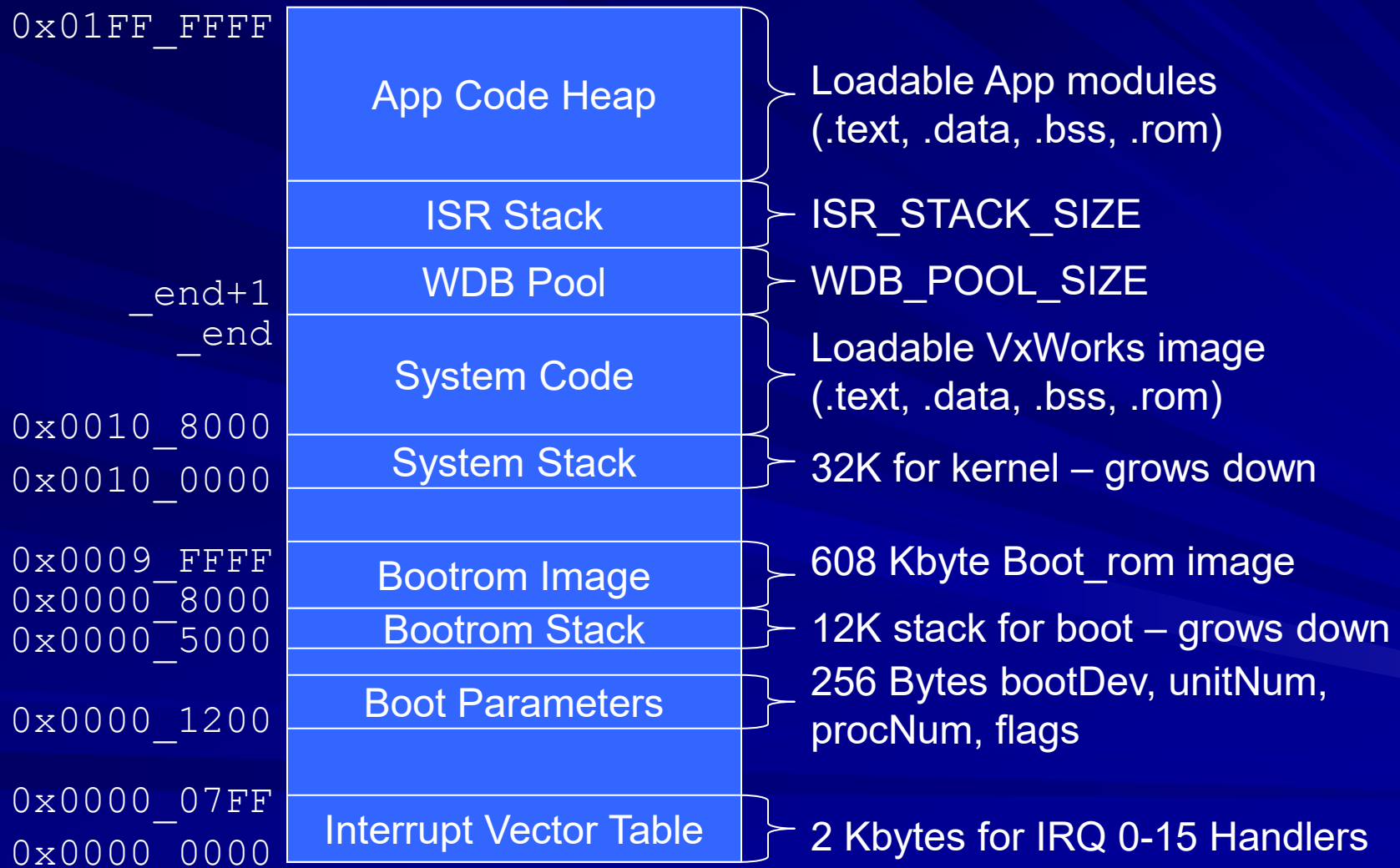
CPU Core I/O (HW View)

- Word
  - Register Control/Config, Status, Data
  - Typical of Low-Rate I/O Interfaces (RS232)
- Block
  - FIFOs (2-32K), Dual-Port RAM and DMA
  - Typical of High-Rate I/O Interfaces
- System Memory Map (32 or 64 bit space)
  - Physical Memory (SDRAM, DDR, SRAM)
  - BootROM (Typically Flash)
  - Configuration and Control Registers

Example 4Gb System Memory Map



OS and App Use of 32 Mb Memory



Linux Platform I/O (FW View)

- Interface Between Protected User Space Processes and Kernel Space Device Interfaces
  - Address Space Protection, Security, Usage Policies
  - Kernel Module or Built-in
- Driver Interfaces Used By Applications
  - Standard Entry Points: Application Interface through /dev/<device> using mknod and <major>, <minor> either manually or automatic
  - Device Interface: FW Interface to HW or UEFI/BIOS
- UEFI/BIOS Driver Interface
  - Firmware LSP (Linux Support Package), Uboot is most Universal for Embedded Linux
  - UEFI 2.3.1 (Well deployed in server systems)
  - Legacy BIOS in Smaller Desktop Systems
- User Space Driver Types
  - Word-at-a-Time I/O Drivers
    - Serial byte streams
    - Device Configuration, Control, and Status
  - Block I/O Drivers
    - Disk, Flash, High-Rate Network (Typically DMA)
  - Network or Storage Stack I/O Drivers
    - File system, TFFS/DOC, TCP/IP
  - OS by-pass more recently

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability, flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

Linux Platform I/O (FW View)

- Interface Between Protected User Space Processes and Kernel Space Device Interfaces
  - Address Space Protection, Security, Usage Policies
  - Kernel Module or Built-in
- Driver Interfaces Used By Applications
  - Standard Entry Points: Application Interface through /dev/<device> using mknod and <major>, <minor> either manually or automatic
  - Device Interface: FW Interface to HW or UEFI/BIOS
- UEFI/BIOS Driver Interface
  - Firmware LSP (Linux Support Package), Uboot is most Universal for Embedded Linux
  - UEFI 2.3.1 (Well deployed in server systems)
  - Legacy BIOS in Smaller Desktop Systems
- User Space Driver Types
  - Word-at-a-Time I/O Drivers
    - Serial byte streams
    - Device Configuration, Control, and Status
  - Block I/O Drivers
    - Disk, Flash, High-Rate Network (Typically DMA)
  - Network or Storage Stack I/O Drivers
    - File system, TFFS/DOC, TCP/IP
  - OS by-pass more recently

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability, flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

Linux Platform I/O (FW View)

- Interface Between Protected User Space Processes and Kernel Space Device Interfaces
  - Address Space Protection, Security, Usage Policies
  - Kernel Module or Built-in
- Driver Interfaces Used By Applications
  - Standard Entry Points: Application Interface through /dev/<device> using mknod and <major>, <minor> either manually or automatic
  - Device Interface: FW Interface to HW or UEFI/BIOS
- UEFI/BIOS Driver Interface
  - Firmware LSP (Linux Support Package), Uboot is most Universal for Embedded Linux
  - UEFI 2.3.1 (Well deployed in server systems)
  - Legacy BIOS in Smaller Desktop Systems
- User Space Driver Types
  - Word-at-a-Time I/O Drivers
    - Serial byte streams
    - Device Configuration, Control, and Status
  - Block I/O Drivers
    - Disk, Flash, High-Rate Network (Typically DMA)
  - Network or Storage Stack I/O Drivers
    - File system, TFFS/DOC, TCP/IP
  - OS by-pass more recently

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability, flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

Linux Platform I/O (FW View)

- Device Interface Examples
  - Word-at-a-Time
    - USB, RS-232/422 UART, GPIO, RELAY, ADC
  - Block I/O
    - SCSI/SATA/SAS, ONFI (Flash), Ethernet Link Layer
  - Stack I/O
    - Last Stack Layer Interface to HW/SW Layer
      - HW/SW Abstraction Exported To Higher Layers
      - Only This Layer Must Change When HW is Changed
      - E.g. Flash Filesystem MTD in TFFS

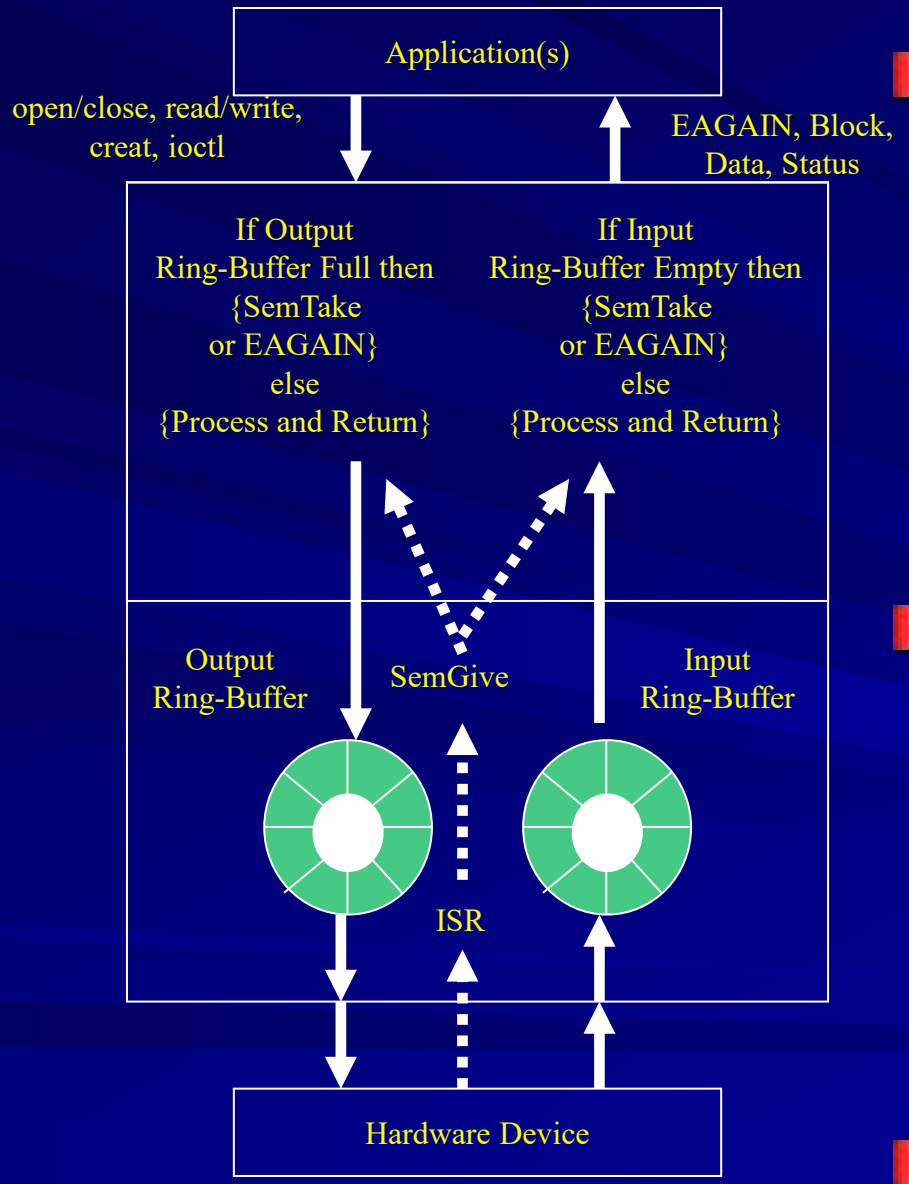
Platform/RTOS I/O (FW View)

- Driver Interfaces Used By Applications
  - Top-Half: Application Interface
  - Bottom-Half: FW Interface to HW
- Application Interface Examples
  - Word-at-a-Time I/O Drivers
    - Serial byte streams
    - Device Configuration, Control, and Status
  - Block I/O Drivers
    - Disk, Flash, High-Rate Network (Typically DMA)
  - Stack I/O Drivers
    - Filesystem, TFFS/DOC, TCP/IP

Platform/RTOS I/O (FW View)

- Device Interface Examples
  - Word-at-a-Time
    - RS-232/422 UART, GPIO, RELAY, ADC
  - Block I/O
    - Common Flash Interface, Ethernet Link Layer
  - Stack I/O
    - Last Stack Layer Interface to HW/SW Layer
      - HW/SW Abstraction Exported To Higher Layers
      - Only This Layer Must Change When HW is Changed
      - E.g. Flash Filesystem MTD in TFFS

Driver Design



- Application Interface
  - Application Policy
  - Blocking/Non-Blocking
  - Multi-thread access
  - Abstraction
- Device Interface
  - SW/HW Interface
  - Immediate Buffering
  - Interrupt Service Routine
- HAL Interface

Cached Memory and DMA

- Cache Coherency
  - Making sure that cached data and memory are in sync
  - Can become out of sync due to DMAs and Multi-Processor Caches
  - Push Caches Allow for DMA into and out of Cache Directly
  - Cache Snooping by HW may Obviate Need for Invalidate
- Drivers Must Ensure Cache Coherency
  - Invalidate Memory Locations on DMA Read Completion
    - cacheInvalidate()
  - Flush Cache Prior to DMA Write Initiation
    - cacheFlush()
- IO Data Cache Line Alignment
  - Ensure that IO Data is Aligned on Cache Line Boundaries
  - Other Data That Shares Cache Line with IO Data Could Otherwise Be Errantly Invalidated

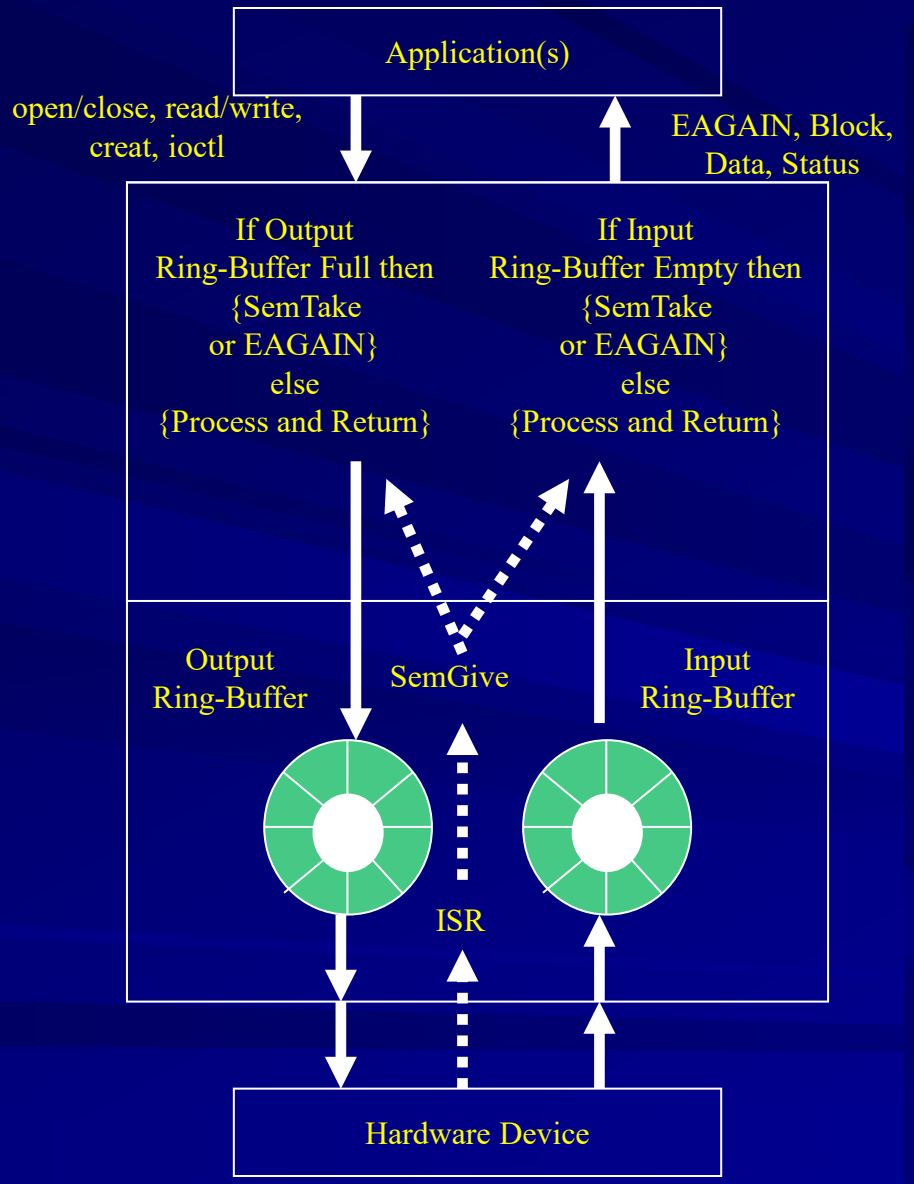
Disadvantages of Abstracted Driver

- Full Drive Abstraction Adds Overhead and Complexity
  - Function calls and table look-up
  - More Code than Single Layer API
- Adds Buffering and Copy
  - Can Use Zero-Copy Buffering (Pointers) to Minimize Impact
  - Zero-Copy Buffers can be Complex
- PDL – Peripheral Driver Library, Focus on Device Interfaces [Devices Interfaces and HAL]
- Generally the Software Engineering Value Outweighs any Inefficiency

Advantages of Abstracted Driver

- **Portability**
  - If SW/HW Interface changes, change BH
  - If Application Interface changes, change TH
- **Testability (Test BH and TH Separately)**
- **Single Point of Access and Maintenance**
- **Enforce Multi-Thread Usage Policies**
- **Separate Buffering and ISR from Usage**
- **Common Application Entry Points**
- **Scheduled I/O (Most Work in Task Context rather than ISR Context)**

Driver Design



- Top-Half
  - Application Policy
  - Blocking/Non-Blocking
  - Multi-thread access
  - Abstraction (Standard Entry Points)
- Bottom-Half
  - SW/HW Interface
  - Immediate Buffering
  - Interrupt Service Routine
- BH-to-TH Interface

Driver Writer Resources

- Linux Device Drivers
  - Linux Device Drivers, by Rubini and Corbet
  - Jerry Cooperstein's Linux Device Drivers - http://www.coopj.com/, http://www.coopj.com/LDD/ Also Linux Driver Solutions from Cooperstine.
  - Linux Foundation
- VxWorks RTOS Device Interfaces
  - Section 3.9 of VxWorks Programmer's Guide, pages 166-174
  - VxWorks Programmer's Guide, Appendix D for x86 Architecture – pages 431-478
- FreeRTOS Device Drivers
  - FreeRTOS plus IO , a Peripheral Driver Library
  - Writing Drivers for FreeRTOS
  - MPLAB Harmony