Device Drivers

Chapter 8

Layer Model

- Device drivers are the liaison between the hardware and the operating system, middleware and application layers
- Layers
  - Hardware
  - Device driver
  - Middleware, or operating system
  - Application

Categories

- Types of device drivers
  - Architecture specific
    - Specified by the master processor
  - General
    - Not specified by the master processor
    - Specified by the hardware
- Code components
  - Initialize hardware
  - Manage access to hardware

Device Driver Functions

- Functionality
  - Hardware startup/shutdown
  - Hardware enable/disable
  - Hardware acquire/release
  - Hardware read/write
  - Hardware install/uninstall
  - Hardware status query
  - Concurrency control
  - Provide interfaces to upper layers

Example: Case 12

- Boot loader
  - A driver to start and initialize the whole system
- For ATmega128
  - Set up the interrupt vector table
  - Clear SREG
  - Set stack pointer
  - Enable sleep
  - Initialize DATA and BSS
  - Configure on-board devices (SPI)
  - Enable interrupt
  - Start main

Device Driver Models

- Models
  - Inclusive source codes
    - Included in source codes of applications
  - Linked function calls (static link)
    - Compiled in OBJ and linked in executables
  - Loaded system modules (dynamic link)
    - Compiled and loaded as modules and libraries
- Provide mechanisms for upper layers to
  - Control devices
  - Communicate with devices

Inclusive Source Codes

- Layers
  - Hardware
  - Device driver
    - No operating system or integrated into operating system
  - Application
- Interface to upper layers
  - No interface to operating systems
  - Wrapped interface to applications
    - Applications directly use the device driver

Inclusive Source Codes: Case 13

- In tos.chips.atm128.pins.HplAtm128GeneralIOPinP.nc
```
inline async command void IO.set() { SET_BIT  (port, bit); }
```
- In tos.chips.atm128.atm128hardware.h and other .h
```
#define SET_BIT(port, bit) ((port) |= _BV(bit))
#define _BV(bit) (1 << (bit))
#define PORTA _SFR_IO8(0x1B)
#define _SFR_IO8(io_addr) _MMIO_BYTE((io_addr) + 0x20)
#define _MMIO_BYTE(mem_addr) (*(volatile uint8_t *)(mem_addr))
```
- So, if it is **SET_BIT(PORTA, 0)**
  (*(volatile uint8_t*)(0x1B+0x20))|=(1<<0)
- In app.c, HplAtm128GeneralIOPinP0IO$set()
  - * (volatile uint8_t *)59U |= 1 << 0;

Linked Function Calls

- Layers
  - Hardware
  - Device driver
  - Middleware or operating system
  - Application
- Interface to upper layers
  - Same interface to both operating system and applications

Example: Case 14

```
; ledset(ledchoice, ledstatus)
#define ledchoice r25
#define ledstatus r24
#define ledsetreturn r16
.text
.global _U(ledset)
_U(ledset):
push XL
push XH
cpi ledchoice, 3
brlo _LED_SHIFT
ldi ledsetreturn, 0x00
rjmp _LED_RETURN
_LED_SHIFT:
mov XL, ledchoice
ldi XH, 0x01
_LED_SHIFT_START:
cpi XL, 0x00
breq _LED_SHIFT_END
lsr XH
dec XL
rjmp _LED_SHIFT_START
_LED_SHIFT_END:in XL, 0x19
cpi ledstatus, 0x00
breq _LED_CLEAR
_LED_SET:
or XL, XH
rjmp _LED_SET_END
_LED_CLEAR:
com XH
and XL, XH
_LED_SET_END:
out 0x1B, XL
ldi ledsetreturn, 0x01
_LED_RETURN:
pop XH
pop XL
ret
```

Loaded System Modules

- Layers
  - Hardware
  - Device driver
  - Middleware or operating system
  - Application
- Interface to upper layers
  - Specified Interface to operating system
  - Wrapped interface to applications

Loaded System Modules: Case 15

- Example : Linux kernel device module
  - Kernel interface

```
static int __init test_init(void);
static void __exit test_exit(void);
int test_open(struct inode* in, struct file* fp);
int test_release(struct inode* in, struct file* fp);
ssize_t test_read(struct file *filp, char __user *buf,
  size_t count, loff_t *f_pos);
ssize_t test_write(struct file *filp, const char __user
  *buf, size_t count, loff_t *f_pos);
```

  - Application interface

```
int open(const char *pathname, int flags)
int close(int fd);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
```

Discussion

- What functions are implemented in the three examples?
  - Inclusive driver
    - Hardware startup/shutdown
    - Hardware read/write
    - Hardware status query
  - Static linked driver
  - Dynamic linked driver
    - Hardware install/uninstall
    - Hardware read/write
    - Hardware acquire/release

Examples of Device Drivers

- Interrupt-handling device driver
- Memory device driver
- On-board bus device driver
- Board I/O device driver
- Our own device driver
  - SPI
  - LED
  - USART

Interrupt-handling Device Driver

- Components
  - Interrupt controller
    - Interrupt-handling startup
    - Interrupt-handling shutdown
  - Interrupt service
    - Interrupt-handling disable
    - Interrupt-handling enable
    - Interrupt-handler servicing

Interrupt Trigger

- Level trigger
  - CPU samples the level at the end of the execution of an instruction.
  - After a level triggers an interrupt, if the level is not disabled before the next CPU sampling, the same interrupt will be serviced again.
- Edge trigger (falling or rising)
  - An edge is latched until the end of the execution of an instruction.
  - If two interrupts sharing the same line make an edge at the same time, only one will be served.

Interrupt Acknowledgment

- Done by hardware
  - Take an amount of time
- Steps
  - A device requests an interrupt.
  - The CPU acknowledges the device via IACK.
  - The requesting device clears the request.
    - Before the device receives the IACK, the device will not requests another interrupt.
    - The device has an internal mechanism to do so.
  - The CPU services the requested interrupt.

Interrupt Priority

- Why priority
  - When two interrupts happen at the same time.
  - When an interrupt deals with an emergency.
- Priority levels
  - Highest priority : non-maskable interrupt
    - RESET, illegal or invalid operations
  - Lower priority
    - Normally, ordered by the interrupt numbers
    - Normally, cannot interrupt higher priority interrupts

Context Switch

- Stop the current execution
- Save context
  - Save the PC (done by the hardware)
    - a) Save into a register and then (the handler) push into the stack
    - b) Push into the stack
  - Save registers (done by interrupt handler)
- Restore context
  - Restore the PC

Interrupt-Handling Performance

- Latency : from when an interrupt is triggered until the interrupt service routine is started
  - Recognize the interrupt request
  - Acknowledge the request
  - Obtain the interrupt vector
  - Switch the context
- Priority
  - Low level interrupts are suspended by high level interrupts

Case 12: SPI

- SPI device in master mode
  - Initialization of SPI
    - Determine mode
    - Configure SPI control register
  - API to transmit data via SPI
    - Determine the API label and the arguments
    - Code the API body
  - Interrupt when transmission completes
    - Determine the interrupt routine and the context
    - Code the interrupt

Case 16: USART0

- Drive components for talking with java listener
  - Initialization of USART
  - PHY layer driver
    - Transmitter and receiver
  - Link layer driver
    - Error check and report
    - ACK protocol
      - Successful receive -> ACK
      - Failed receive -> no ACK -> time out of ACK -> retransmission
    - HDLP protocol
      - Send a packet with multiple PHY frames

USART

- Interface to upper layers
  - Controlling functions
    - Baud rate control
    - Stop bit control
    - Parity control
    - Transmission complete control
  - Application functions
    - Data transmission functions
    - Transmission complete functions

Memory Device Driver

- Components
  - Memory controller
    - Memory subsystem startup
    - Memory subsystem shutdown
    - Memory subsystem disable
    - Memory subsystem enable
  - Memory access
    - Memory subsystem write
    - Memory subsystem read

Example : MPC860

- Initialize memory controller
  - Memory layout : Figure 8-18
  - Memory is divided into 512K banks
  - Memory wiring : Figure 8-20b
  - Each bank has
    - One base register
      - 17-bit starting address (the highest 17 bits of 32-bit real address)
      - 2-bit address(data) type
      - 1 bit write protection
      - ...
    - One option register

Example : MPC860

- Initialize memory management unit
  - Virtual memory and physical memory
  - 4KB page size
  - 1M page entries : 2-level table
  - TLB (translation lookaside buffer) : Figure 8-25
  - Initialization
    - Table entries
    - Control registers

Example : MPC860

- Memory read
- Memory write
  - Flash memory write
  - Flash is divided into blocks
  - Each block is the minimum to be written
  - Steps
    - Notify the flash chip a pending write
    - Send the command to write
    - Polling the flash chip until the writing is completed
    - Turn the flash chip to read mode