Session #9: Device Drivers on the Target Board

Accessing hardware

- Where are I/O registers located?
  - Memory space or separate I/O space
- I/O operations may have "side effects"
  - Compiler optimizations can get in the way
    - Caching
    - Reordering
- I/O space must be allocated

I/O port API

```
#include <asm/io.h>
```

Single item transfer

```
unsigned inb (unsigned port);
void outb (unsigned char byte, unsigned port);
unsigned inw (unsigned port);
void outw (unsigned short word, unsigned port);
unsigned inl (unsigned port);
void outl (unsigned longword, unsigned port);
```

"String" transfer

```
void insx (unsigned port, void *addr, unsigned long count);
void outsx (unsigned port, void *addr, unsigned long count);
```

Note: x is "b", "w", or "l".

I/O memory API

```
void *ioremap (unsigned long phys_addr, unsigned long size);
void iounmap (void *address);

unsigned int ioreadn (void *address);
void iowriten (un value, void * address);

unsigned int ioreadn_rep (void *address, void *buf,
        unsigned long count);
void iowriten_rep (un value, void * address , const void *buf,
        unsigned long count);
```

Note: n = 8, 16, or 32

Allocating an I/O region

```
struct resource *request_region (unsigned long first,
        unsigned long n, const char *name);
void release_region (unsigned long start, unsigned long n);
int check_region (unsigned long first, unsigned long n);

struct resource *request_mem_region (unsigned long first,
        unsigned long n, const char *name);
void release_ mem_ region (unsigned long start,
        unsigned long n);
int check_ mem_ region (unsigned long first, unsigned long n);
```

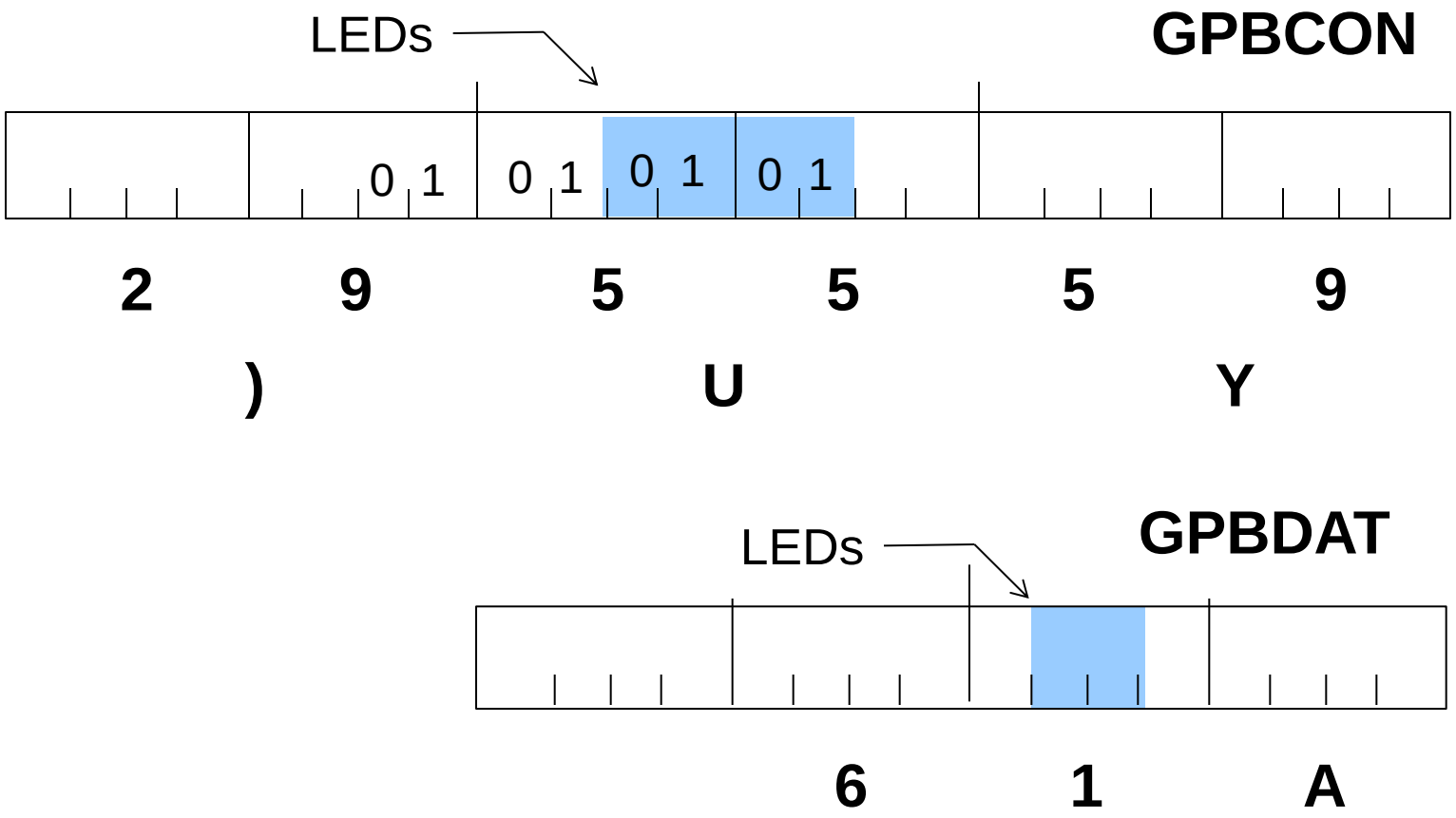
simple_hw Driver

- Open simple_hw.c in target_fs/home/src
- Module parameters
  - pbase – physical base address, PIOB
  - size – transfer size, 1, 2 or 4 bytes, 4
  - offset – within 4-byte register, 0
- 8 devices representing 8 PIO registers starting at GPIO port B
  - /dev/pio_GPBCON, etc
- Open Makefile

Build and run simple_hw

- make
- ./hw_load
- dd bs=4 count=1 if=/dev/pio_GPBCON
  - Should return “YU)” -> 0x295559
- dd bs=4 count=1 if=/dev/pio_GPBDAT
  - Should return <not printable> -> 0x61a

Register layout



Control the LEDs

- echo –n “1“ > /dev/pio_GPBDAT
  - Turns on first LED
- echo –n “2“ > /dev/pio_GPBDAT
  - Turns on second LED
- Etc.

Barriers

Compiler memory barrier

```
#include <linux/kernel.h>
void barrier (void);

for (i = 0; i < 10; i++)
{
        a = b;
        barrier();
}
```

Hardware barriers

```
#include <asm/system.h>
void rmb (void);
void wmb (void);
void mb (void);
```

Interpret hex strings

- echo ”7fa” > /dev/pio_GPBDAT
- Module parameter
  - hex = 1
- ioctl() command
  - IOCSHEX set hex value
  - IOCGHEX get hex value

therm_driver

- Convert trgdrive.c to device driver
- Use therm_driver.c as starting point
- Three devices (minor device numbers)
  - therm0 – read ADC
  - therm1 – set LED output bits (LEDs on)
  - therm2 -- clear LED output bits (LEDs off)
- Peripheral address mapping
  - s3c2410_regs.h

Data modes

- therm_driver can return data in two formats:
  - Binary mode = 0
  - Text mode = 1 – default
- Build project and execute ./therm_load
  - Add therm_driver.o to obj-m line of Makefile
- Try cat /dev/therm0
  - Data comes out very fast
- Introduce delay
  - wait_event_interruptible_timeout()
  - time measured in "jiffies"

Measuring time in the kernel

- HZ - number of kernel timer interrupts per second
  - 400 to 1000 on PCs running 2.6
- jiffies - 32-bit counter incremented at each timer tick. Declared volatile. Low order part of …
- jiffies_64 - 64-bit jiffies counter
  - Not atomic on 32-bit machines. Use get_jiffies_64()
- "Wrap safe" comparison macros
  - time_before (jiffies, x) jiffies < x
  - time_after (jiffies, x) jiffies > x
  - time_before_eq (jiffies, x) jiffies <= x
  - time_after_eq (jiffies, x) jiffies >= x

Thermostat using the driver

- Modify the network thermostat to use the driver
  - Open files to all three therm devices
- Implement ioctl in therm_driver
  - Commands defined in therm_ioctl.h
- Driver runs with mode = 0, delay = 0
  - Load it that way or use ioctl() from app

Congratulations!

You're a Linux hacker!

Review

- Install and get familiar with Linux
  - What's this Open Source stuff all about?
- Set up target board
  - Install class software – cross tool chain
  - Configure host networking and minicom
  - Part of target's root file system is on host
- Eclipse
- Building and debugging applications
  - Simple simulation environment
  - Debugging on the target

Review II

- Posix threads
- Network programming
  - Server and client on same computer or across the network
- Configuring and building the Linux kernel
- BusyBox

Review III

- Display driver
- Linux initialization
  - Starting the application at boot time
  - Putting it all in flash
- Kernel modules and device drivers
- Driver for target hardware
  - Modified thermostat runs with driver

This is it!

- Take the final
- Fill out course evaluation
- Keep on learning!
  - Troll the Internet
  - Find some good books
  - Take more classes