

Lab 5 M1

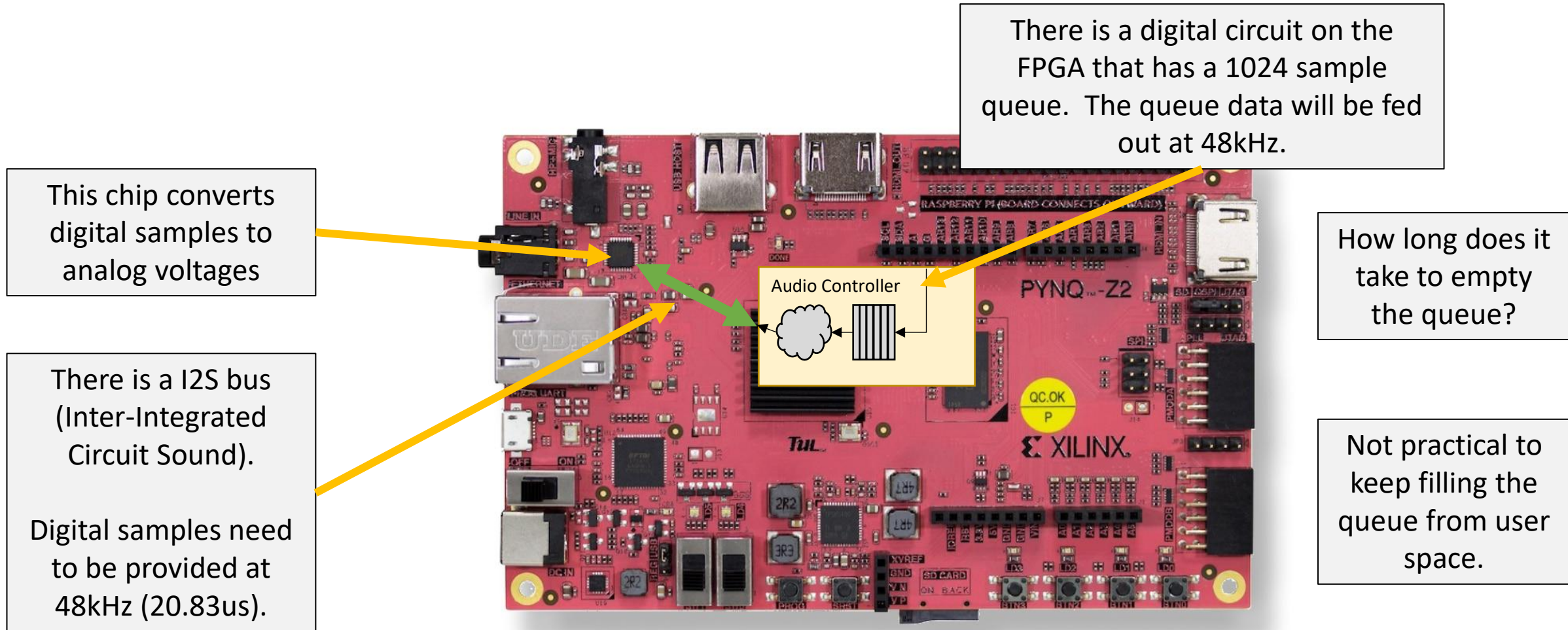
ECEN 427

BYU Electrical & Computer
Engineering

IRA A. FULTON COLLEGE OF ENGINEERING

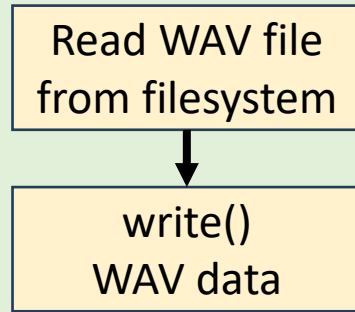
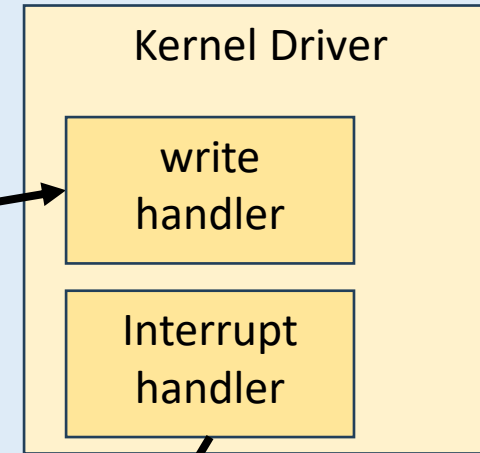
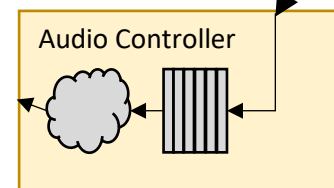
Lab 5

Goal: Play sound effects in space invaders



User Space

Kernel Space



Interrupt
(When 75% empty)

What our driver needs to do:

- Be notified of hardware in the system (Platform devices)
- Allow user code to talk to the driver (Char device, /dev file)
- Talk to the hardware (Memory reservation, mapping)
- Handle interrupts from the hardware (IRQ registration, handler)

Userspace

Application Code
(Space Invaders)

Kernelspace

Kernel Module

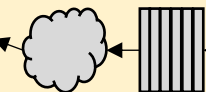
ARM CPU

Audio
CODEC

Audio Controller

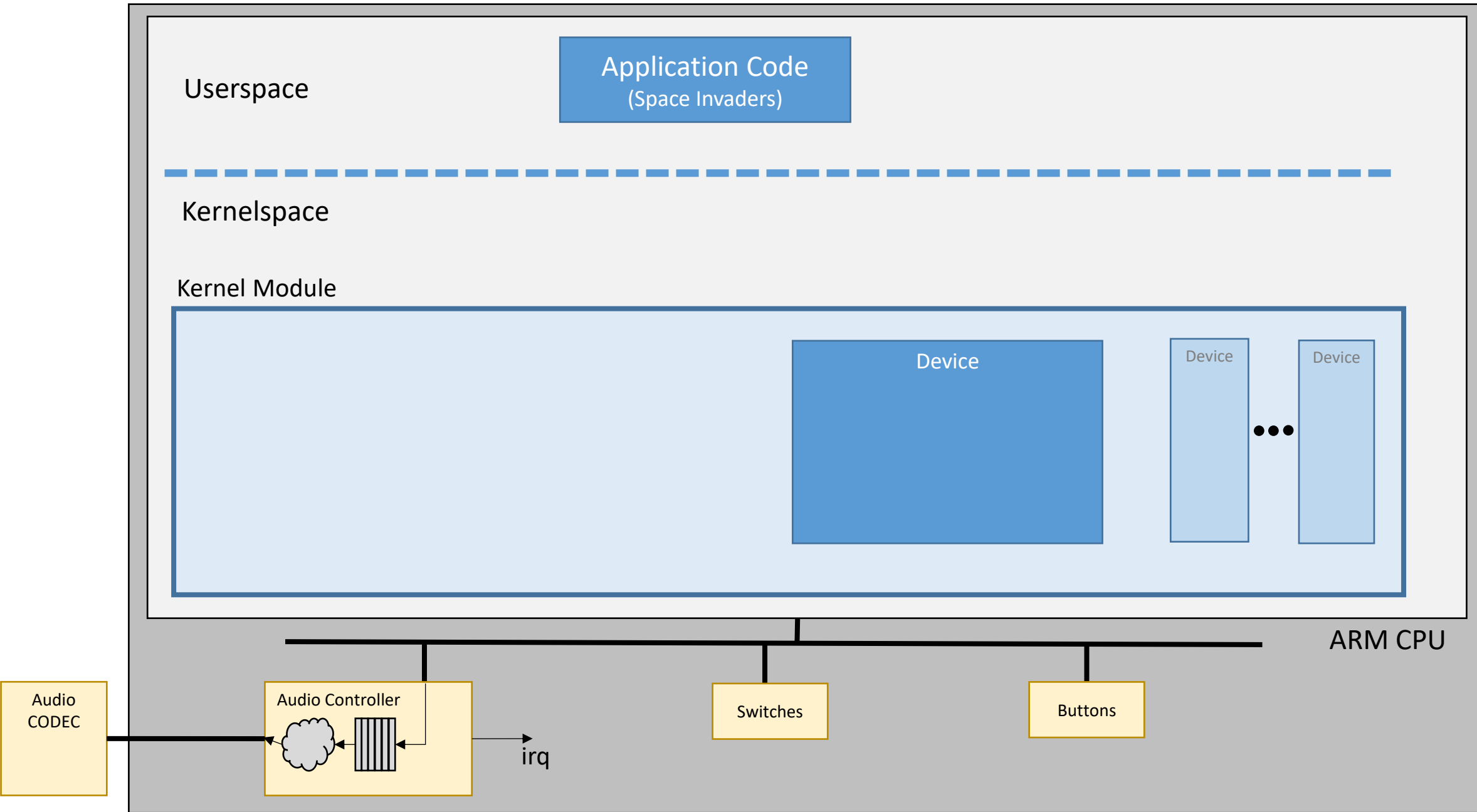
Switches

Buttons



What our driver needs to do:

- Be notified of hardware in the system (Platform devices)
- Allow user code to talk to the driver (Char device, /dev file)
- Talk to the hardware (Memory reservation, mapping)
- Handle interrupts from the hardware (IRQ registration, handler)



Userspace

Application Code
(Space Invaders)

Kernelspace

Kernel Module

```
probe () {  
  ...  
  ...  
}
```

Device

Device

Device

Platform Driver (compatible = "byu,ecen427-audio_codec", .probe, .remove)

ARM CPU

Audio
CODEC

Audio Controller

Switches

Buttons

irq

```
platform_driver_register(struct platform_driver *)
```

What our driver needs to do:

- Be notified of hardware in the system (Platform devices)
- Allow user code to talk to the driver (Char device, /dev file)
- Talk to the hardware (Memory reservation, mapping)
- Handle interrupts from the hardware (IRQ registration, handler)

User Code Needs to Talk to Driver

Goal: Create a device file (/dev/xxx) that we can read() and write() to.

The device file (/dev/xxx) is an interface to a **character device**.

Steps:

1. Create a character device
2. Create a device file

Userspace

Application Code
(Space Invaders)

Kernelspace

Kernel Module

`int major_num;`

```
probe() {  
  ...  
  ...  
}
```

struct

Device

`int minor_num;`

Device

Device

...

Platform Driver (compatible = "byu,ecen427-audio_codec", .probe, .remove)

ARM CPU

Audio
CODEC

Audio Controller

Switches

Buttons

irq

```
alloc_chrdev_region(dev_t * output, minor_start, count, MODULE_NAME)
```

Userspace

Application Code
(Space Invaders)

Kernelspace

Kernel Module

`int major_num;``struct file_operations fops;``probe() {
 ...
 ...
}``read() {
 ...
 ...
}``write() {
 ...
 ...
}`

Device

`int minor_num;``struct cdev cdev;`

Device

Device

Platform Driver (compatible = "byu,ecen427-audio_codec", .probe, .remove)

ARM CPU

Audio
CODEC

Audio Controller

Switches

Buttons

irq

`cdev_init (struct cdev*, fops* {.read, .write})`

Userspace

Application Code
(Space Invaders)

Kernelspace

Kernel Module

Character Device
(major, minor)Char.
DeviceChar.
Device

```
int major_num;  
struct file_operations fops;
```

```
probe() {  
    ...  
    ...  
}  
read() {  
    ...  
    ...  
}  
write() {  
    ...  
    ...  
}
```

Device

```
int minor_num;  
struct cdev cdev;
```

Device

Device

Platform Driver (compatible = "byu,ecen427-audio_codec", .probe, .remove)

ARM CPU

Audio
CODEC

Audio Controller

Switches

Buttons

irq

```
cdev_add (struct cdev*, dev_t, count)
```

User Code Needs to Talk to Driver

Goal: Create a device file (/dev/xxx) that we can read() and write() to.

The device file (/dev/xxx) is an interface to a **character device**.

Steps:

1. Create a character device
2. Create a device file

Userspace

Application Code
(Space Invaders)

Kernel space

Kernel Module

Character Device
(major, minor)Char.
DeviceChar.
Device

```
int major_num;  
struct file_operations fops;  
struct class * class
```

```
probe() {  
    ...  
    ...  
}  
read() {  
    ...  
    ...  
}  
write() {  
    ...  
    ...  
}
```

Device

```
int minor_num;  
struct cdev cdev;
```

Device

Device

Platform Driver (compatible = "byu,ecen427-audio_codec", .probe, .remove)

ARM CPU

Audio
CODEC

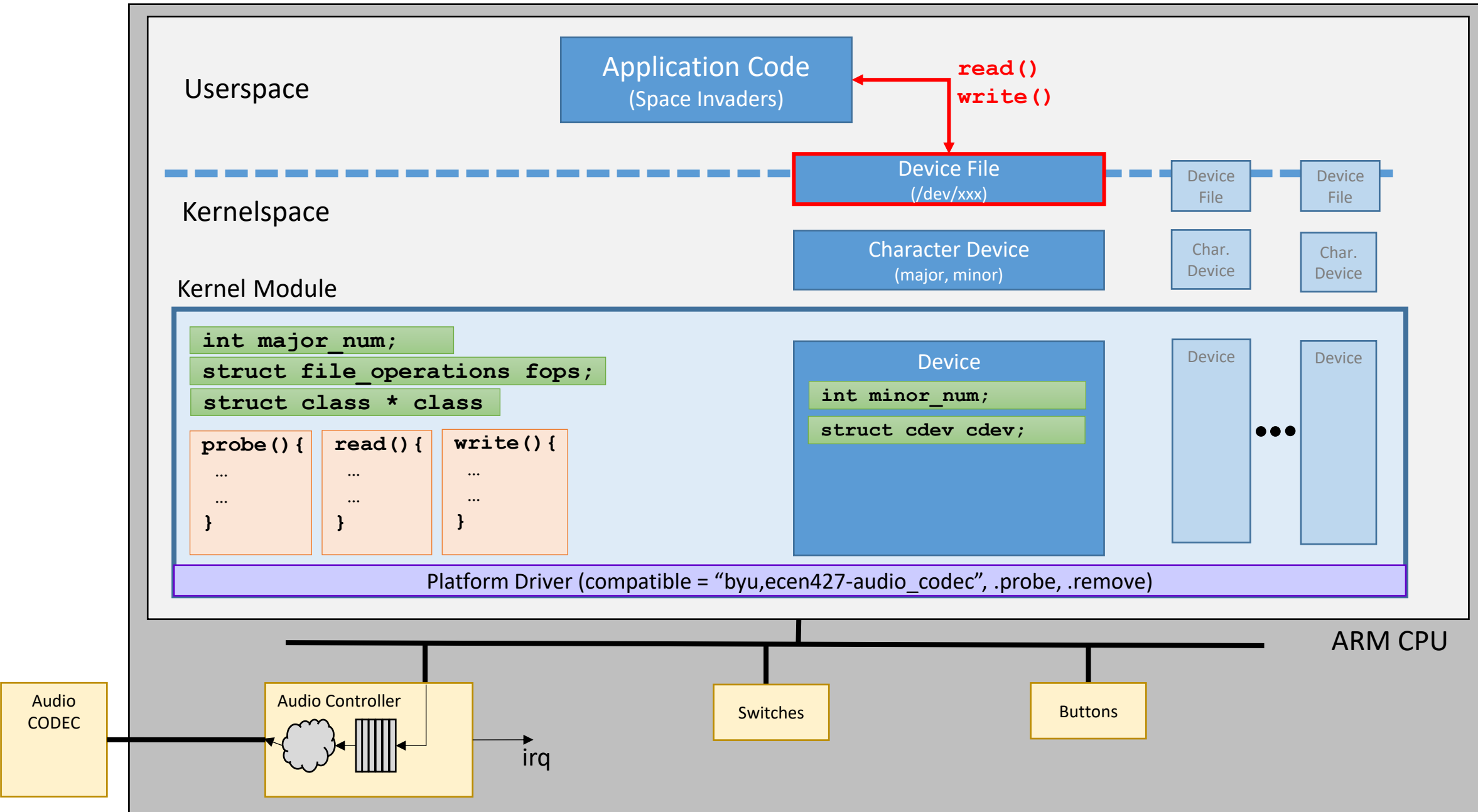
Audio Controller

Switches

Buttons

irq

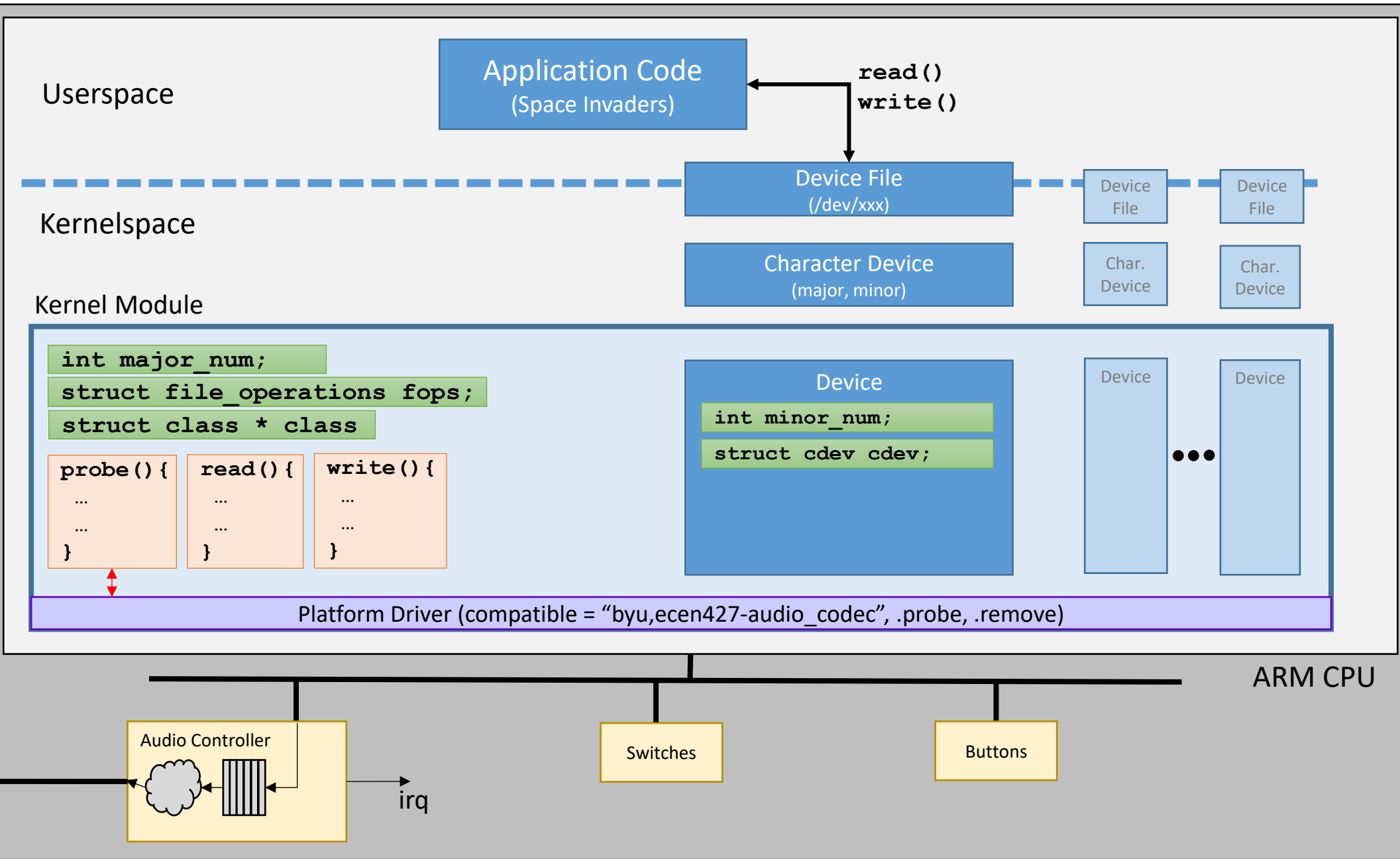
```
class_create("my class name")
```

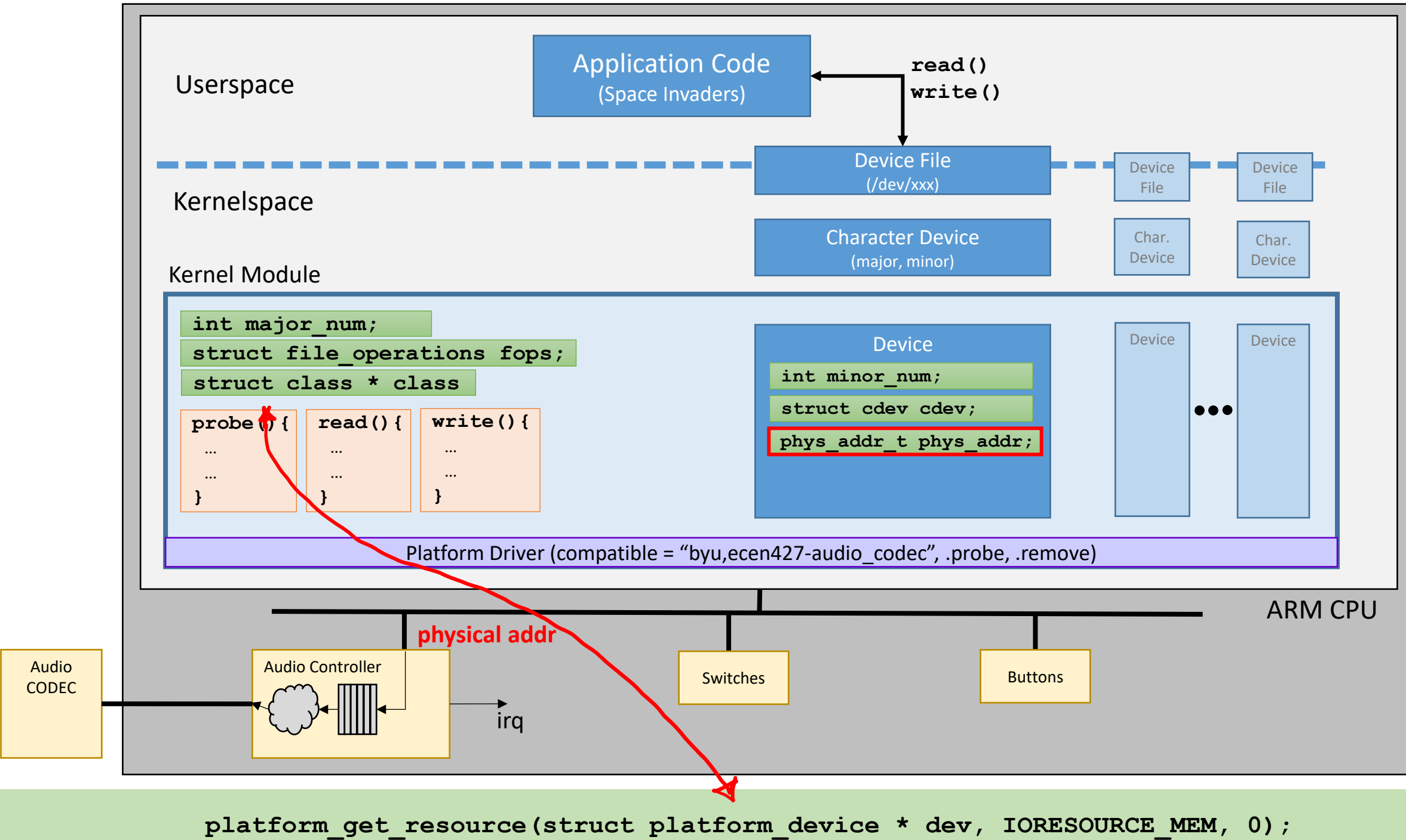


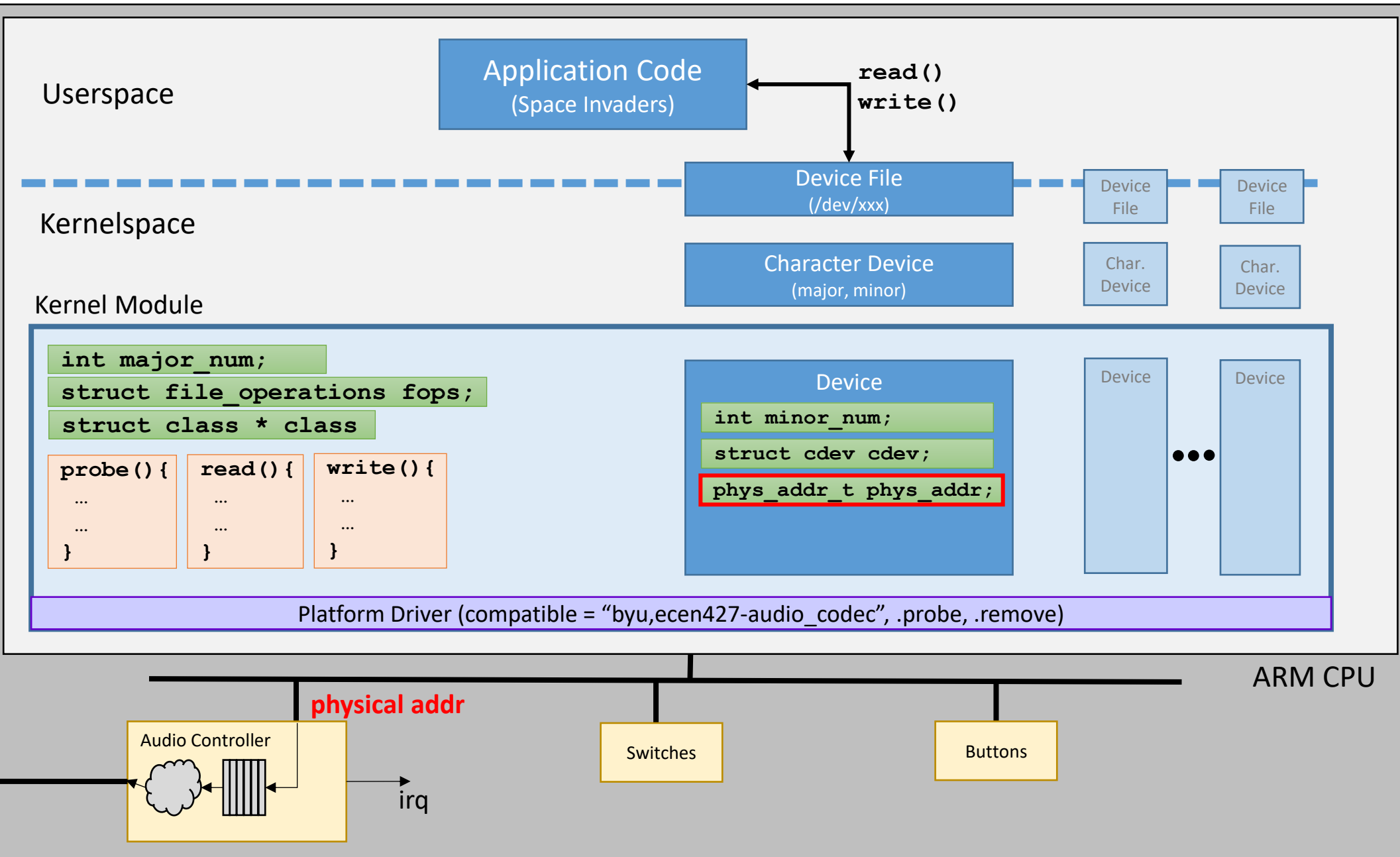
`device_create (struct class*, parent = NULL, dev_t, "your device name")`

What our driver needs to do:

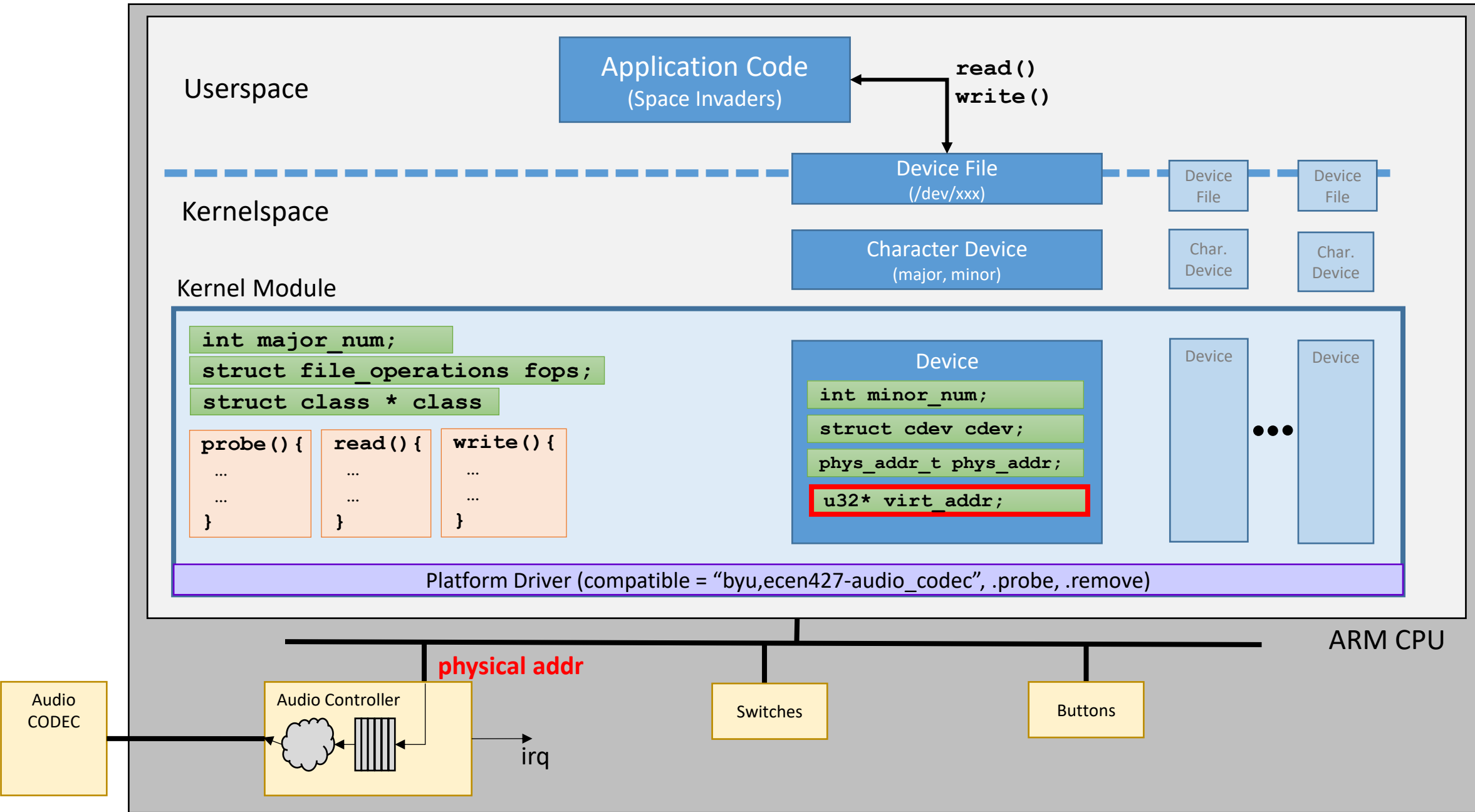
- Be notified of hardware in the system (Platform devices)
- Allow user code to talk to the driver (Char device, /dev file)
- **Talk to the hardware (Memory reservation, mapping)**
- Handle interrupts from the hardware (IRQ registration, handler)







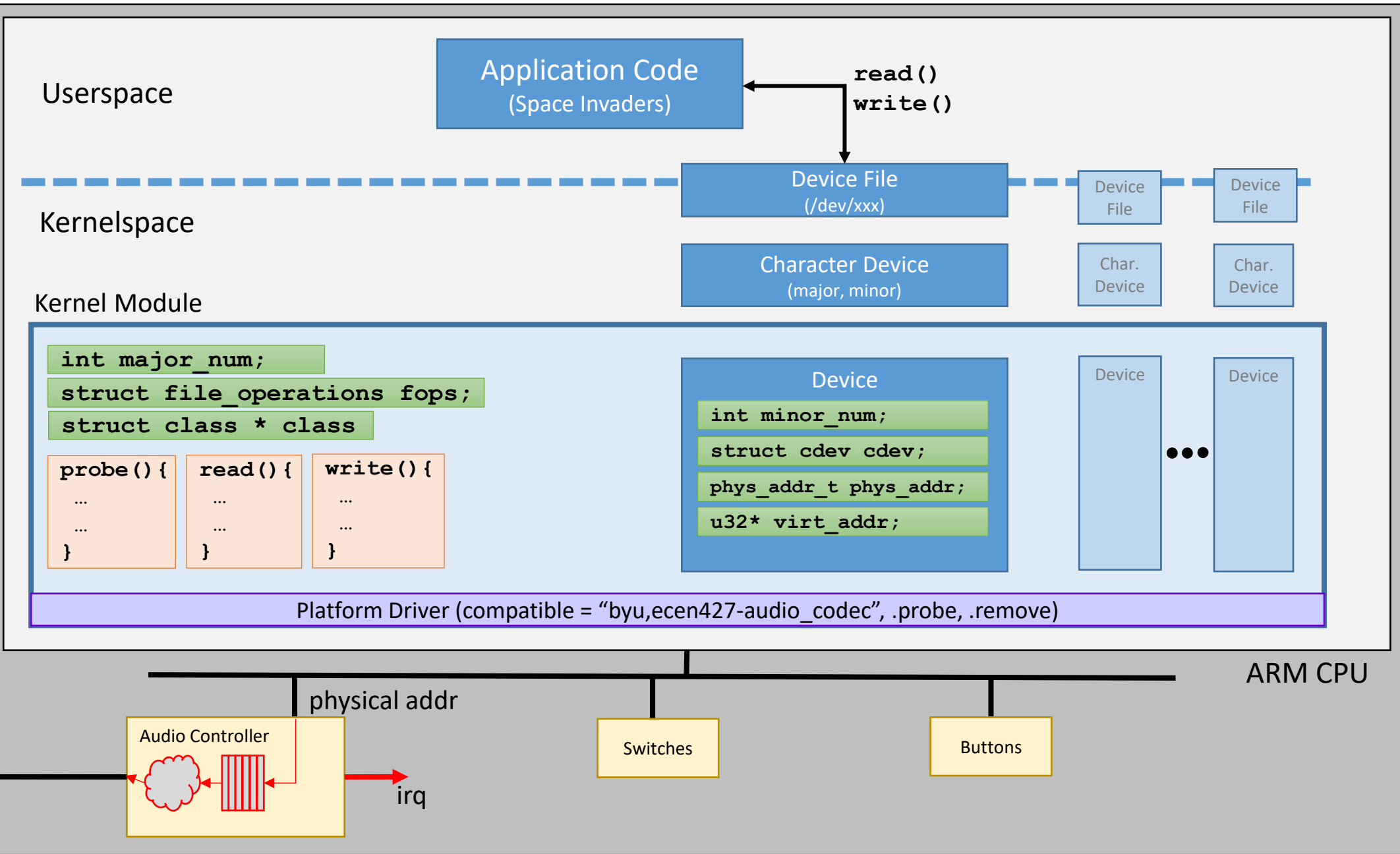
```
request_mem_region(phys_addr, size, MODULE_NAME);
```



```
virt_addr = ioremap(phys_addr, size);
```

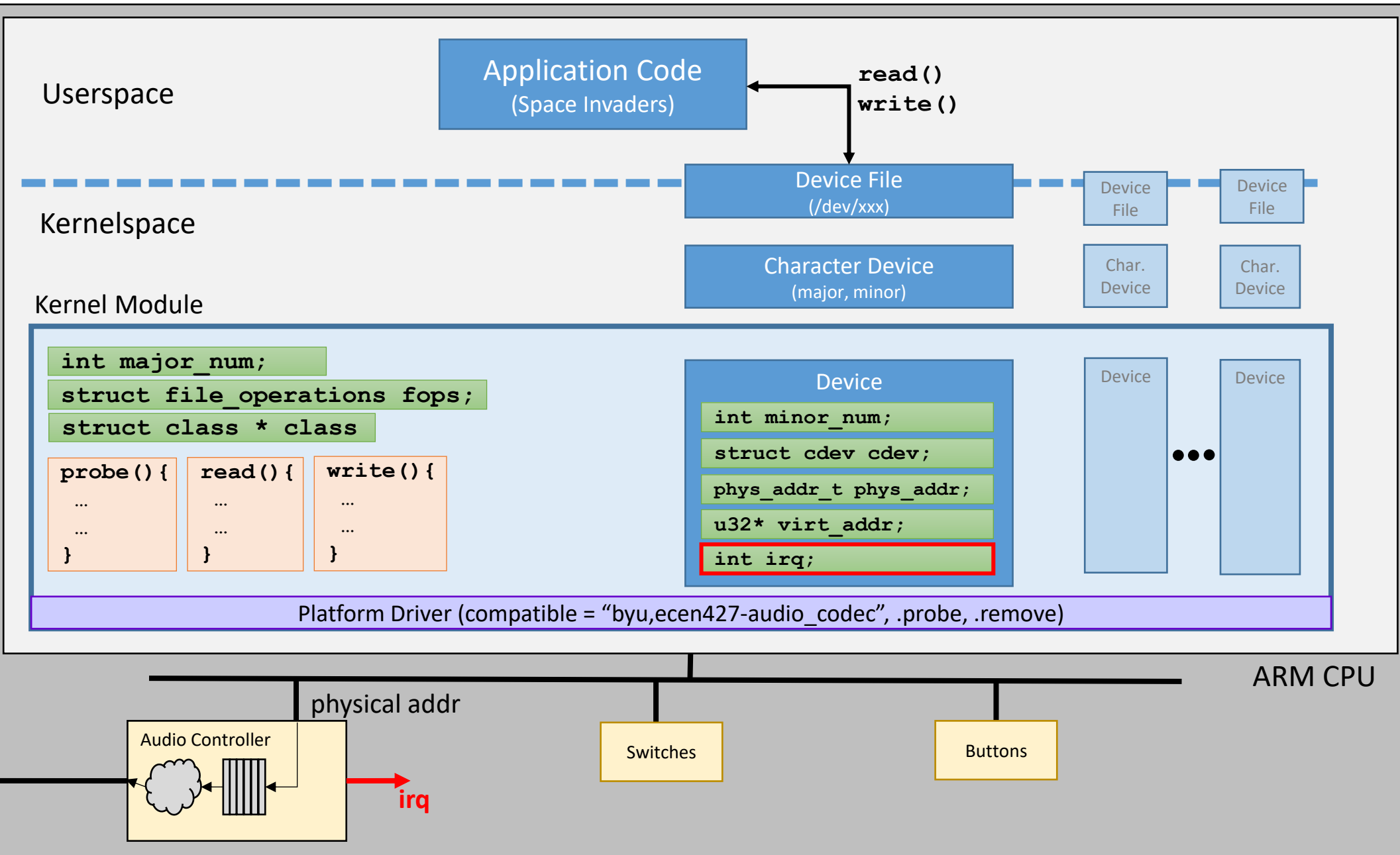
What our driver needs to do:

- Be notified of hardware in the system (Platform devices)
- Allow user code to talk to the driver (Char device, /dev file)
- Talk to the hardware (Memory reservation, mapping)
- Handle interrupts from the hardware (IRQ registration, handler)

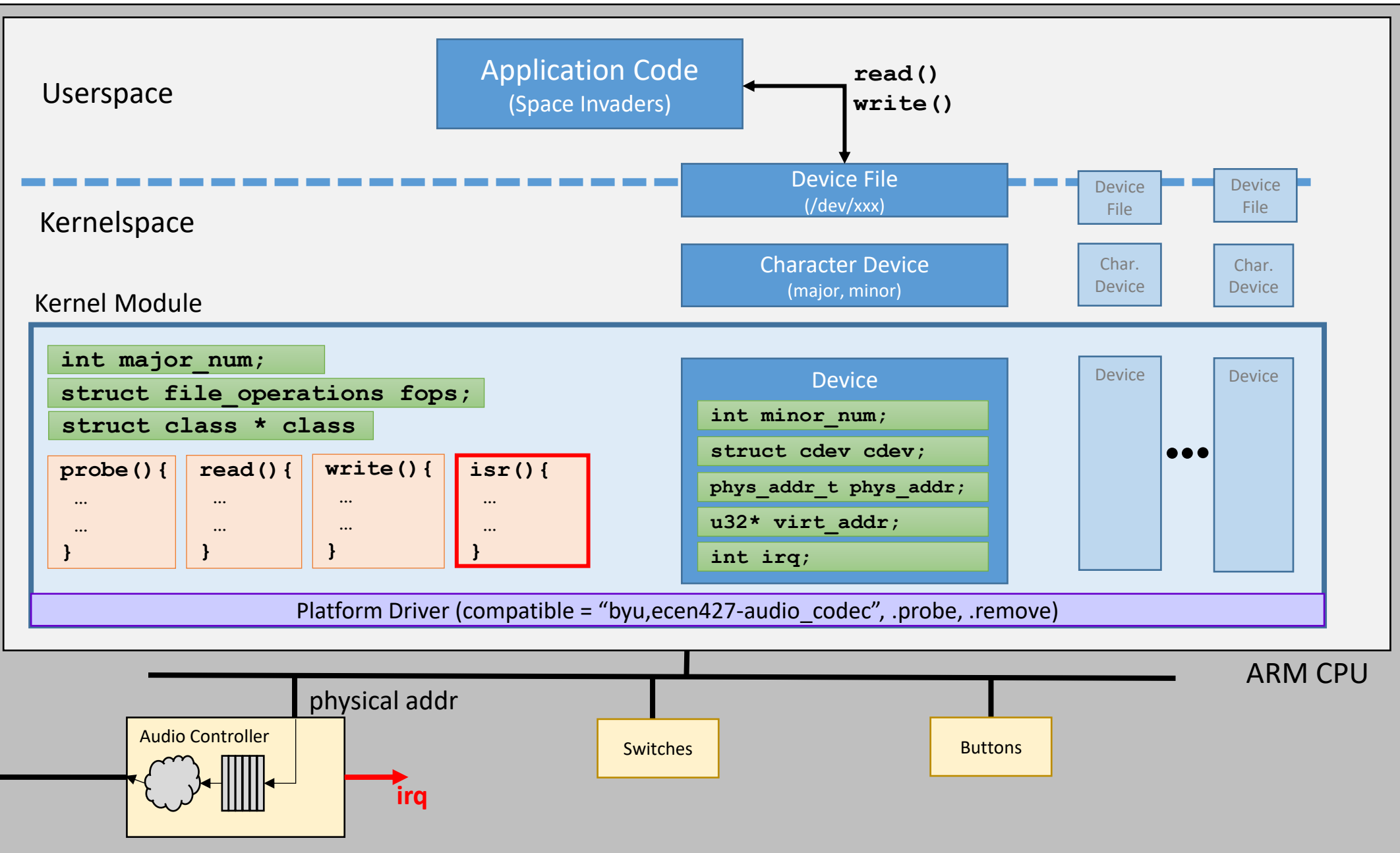


Driver Needs to Handle Interrupts

1. Get IRQ Number
2. Register Interrupt Handler with Linux



```
irq = platform_get_irq(struct platform_device * dev, 0);
```



```
request_irq(irq, isr, IRQ_NO_FLAGS, MODULE_NAME, void* dev_id)
```

