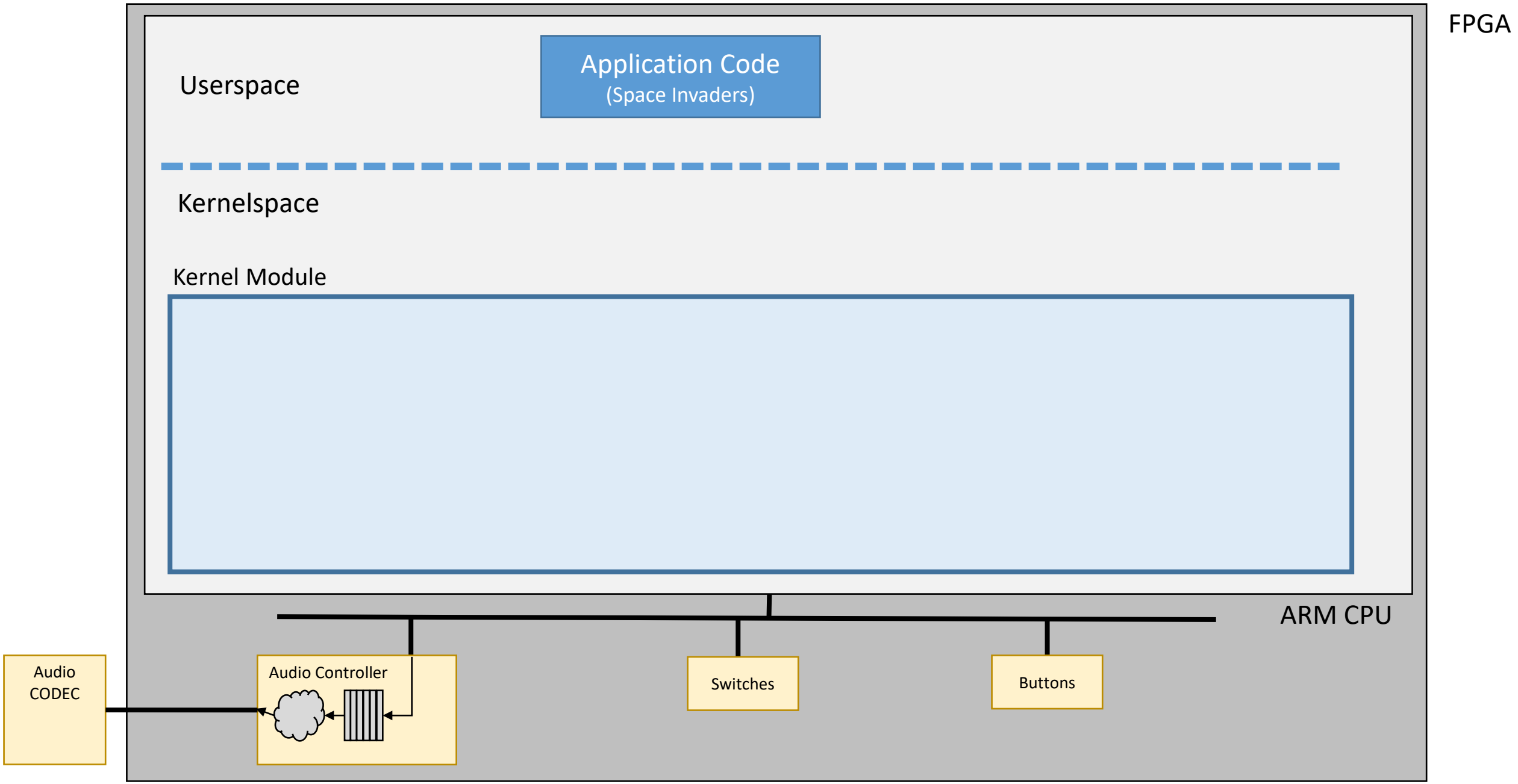
Linux Driver for Audio

Lab 5 Milestone 1 & 2

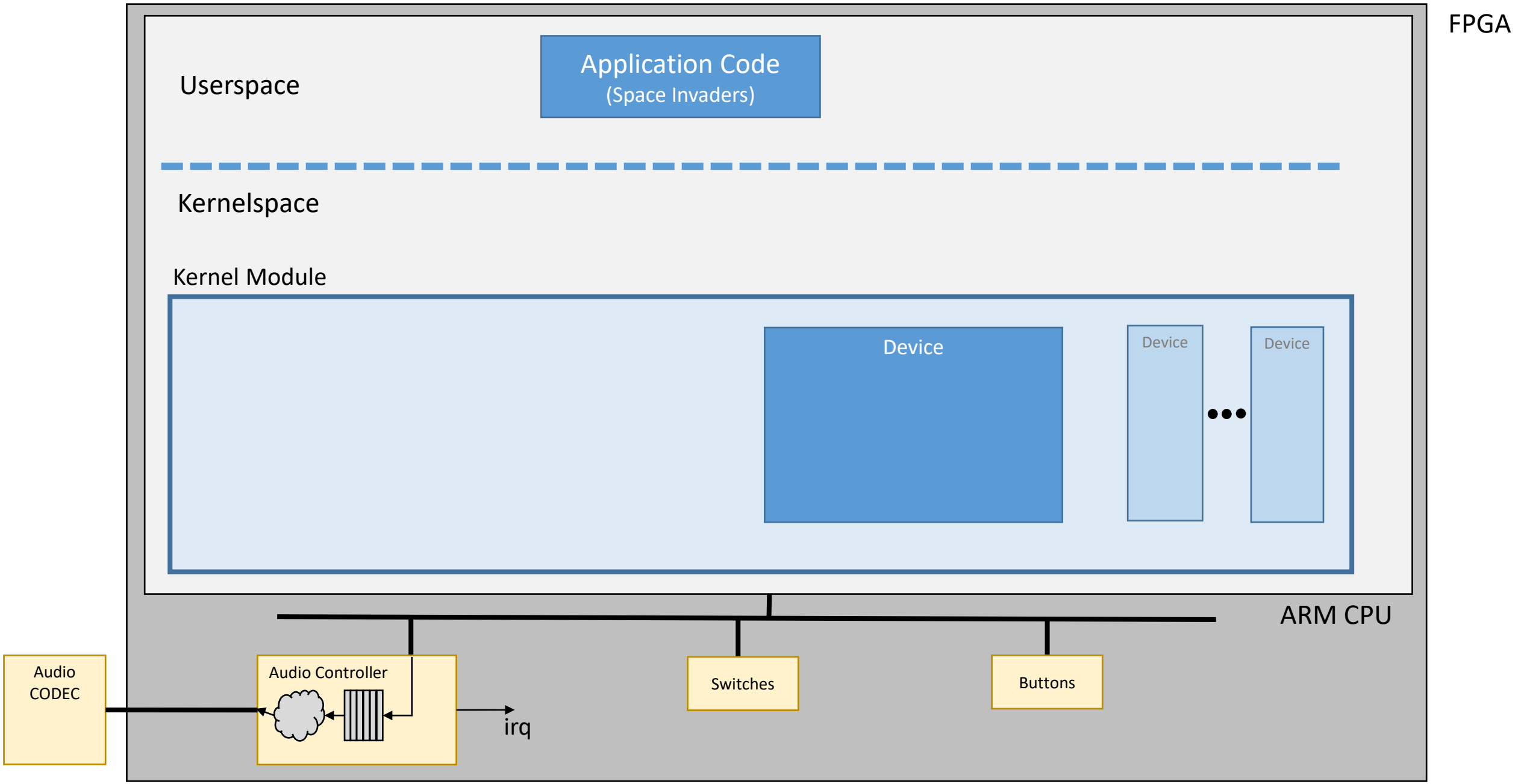
What our driver needs to do:

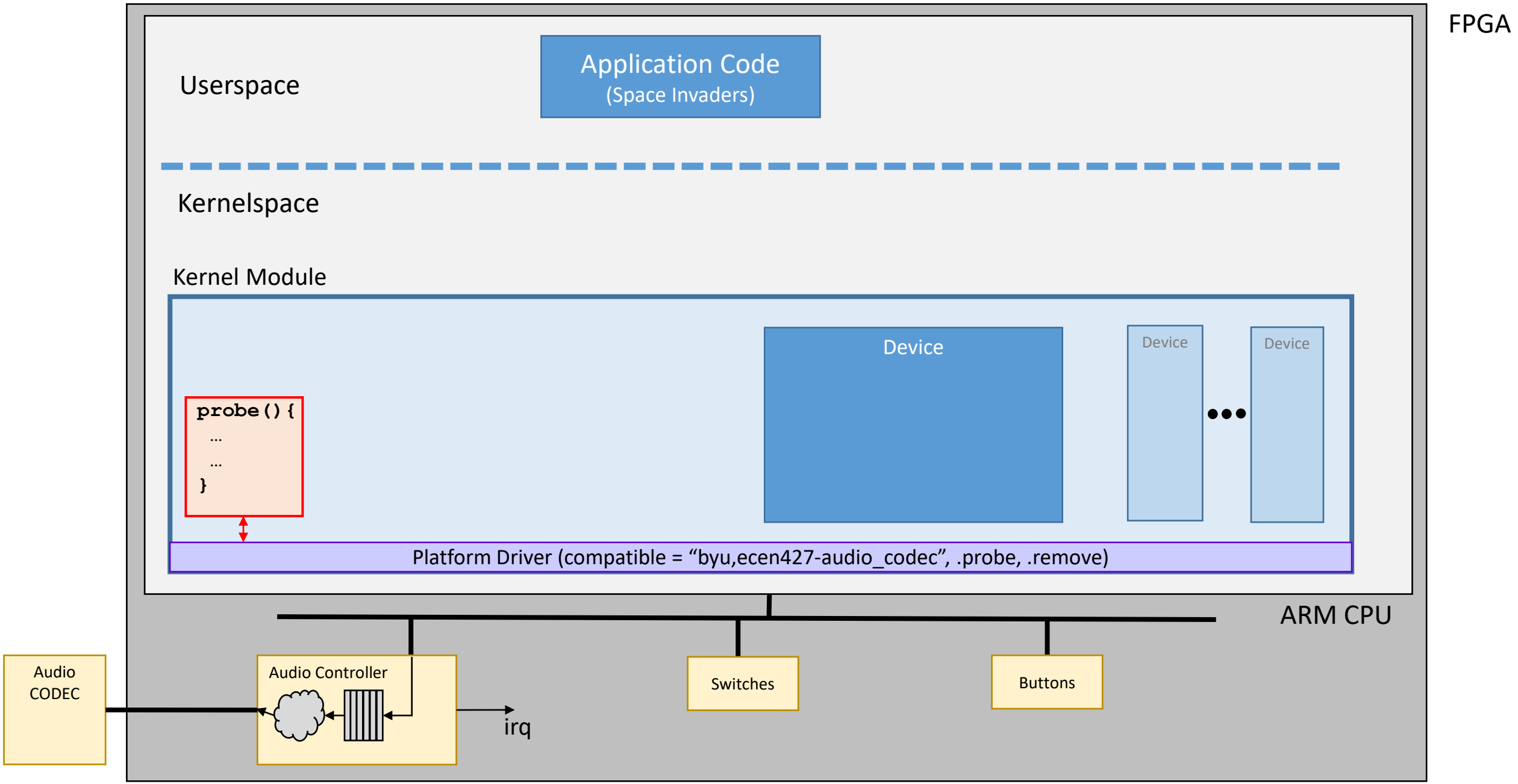
- Be notified of hardware in the system (Milestone 1)
- Allow user code to talk to it (Milestone 1)
- Talk to the hardware (Milestone 2)
- Handle interrupts from the hardware (Milestone 2)


FPGA
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 (Milestone 1)
- Allow user code to talk to it (Milestone 1)
- Talk to the hardware (Milestone 2)
- Handle interrupts from the hardware (Milestone 2)


FPGA
Userspace
Application Code
(Space Invaders)
Kernelspace
Kernel Module
Device
Device
Device
ARM CPU
Audio
CODEC
Audio Controller
irq
Switches
Buttons



`platform_driver_register(struct platform_driver *)`

What our driver needs to do:

- Be notified of hardware in the system (Milestone 1)
- Allow user code to talk to it (Milestone 1)
- Talk to the hardware (Milestone 2)
- Handle interrupts from the hardware (Milestone 2)

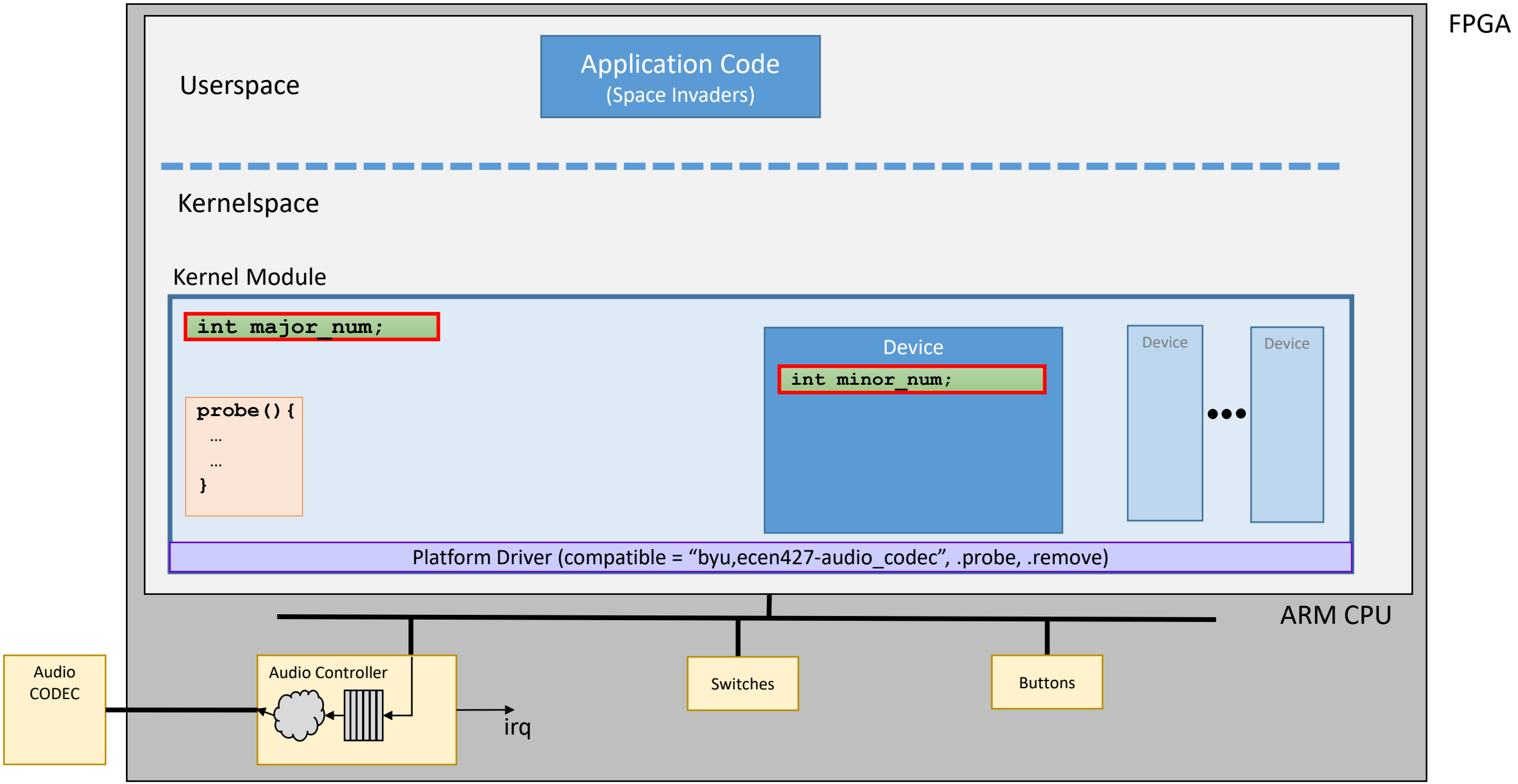
User Code Needs to Talk to Driver

End Goal: Create a device file (/dev/xxx) that we can read() and write() to. (*Recall how you used /dev/uio*)

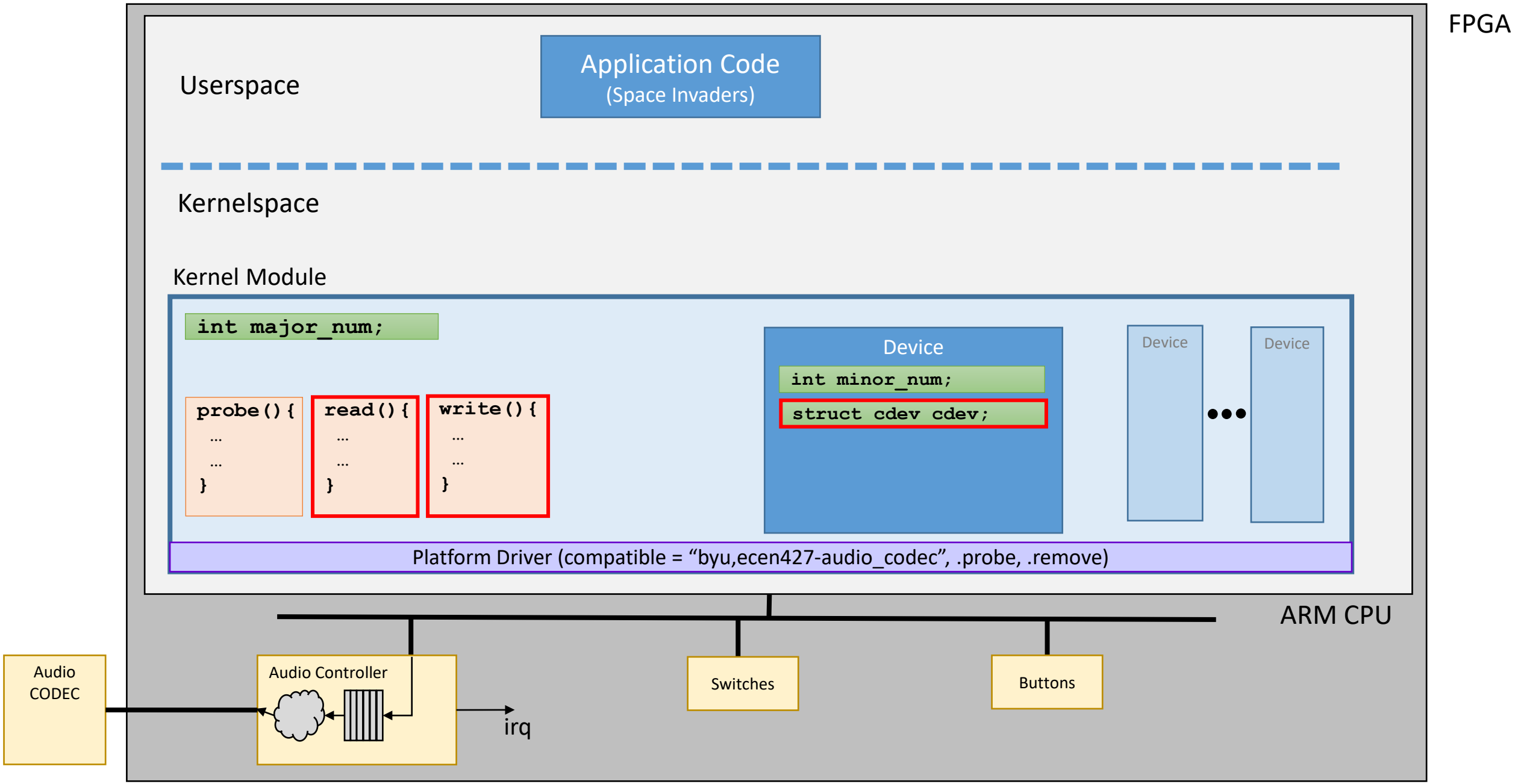
The device file (/dev/xxx) is an interface to a **character device**.

Steps:

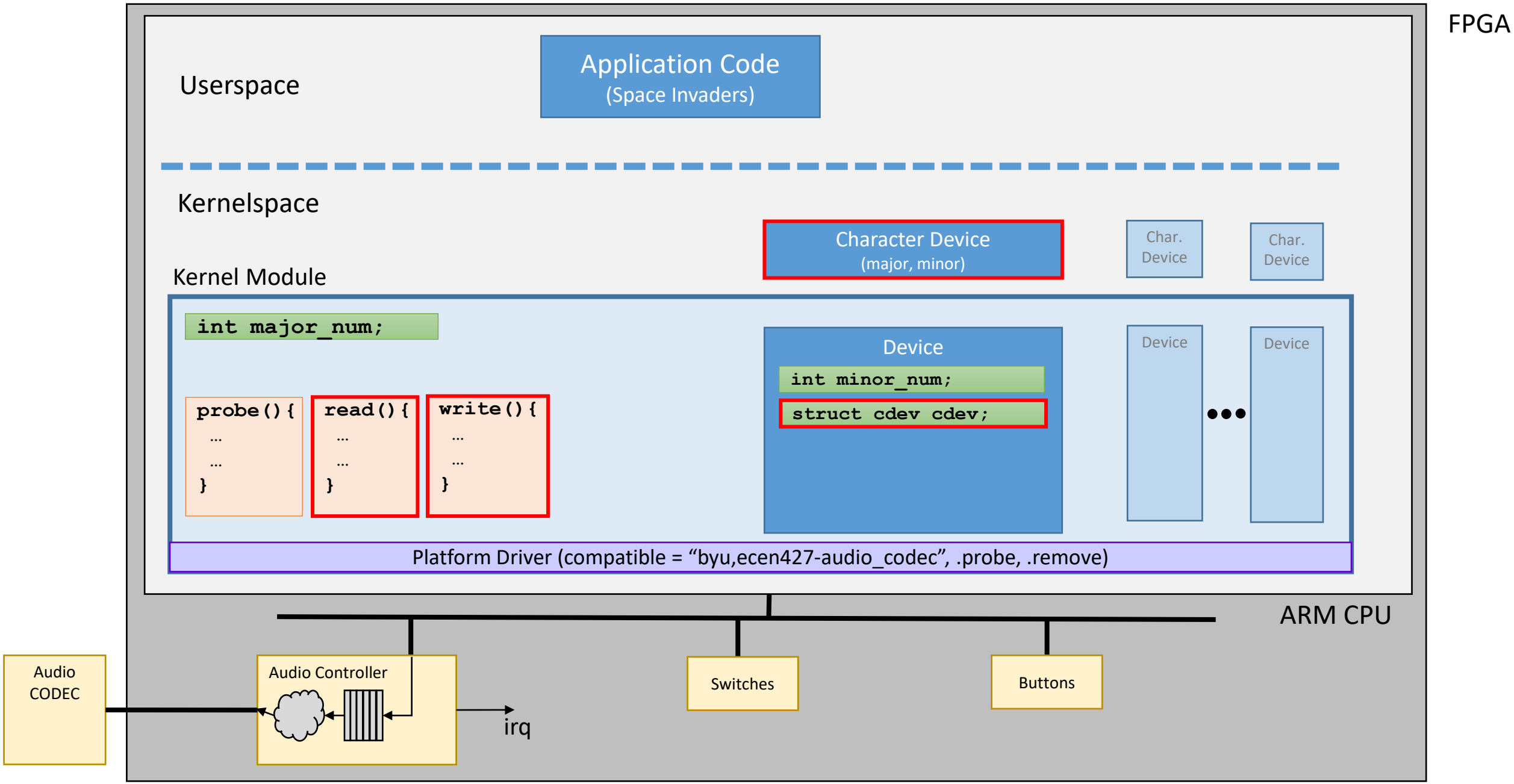
1. Create a character device
2. Create a device file




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




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




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

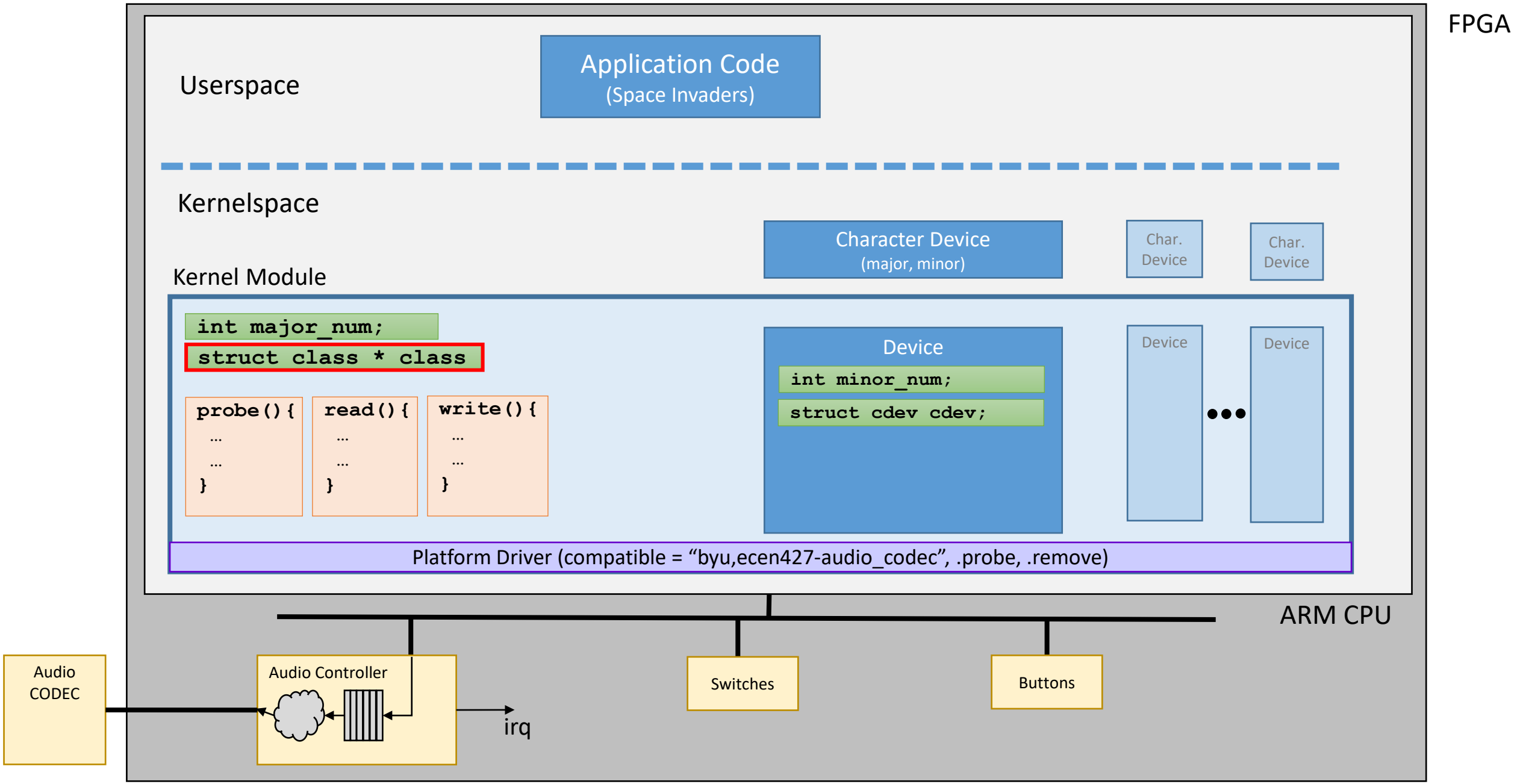
User Code Needs to Talk to Driver

End Goal: Create a device file (/dev/xxx) that we can read() and write() to. (*Recall how you used /dev/uio*)

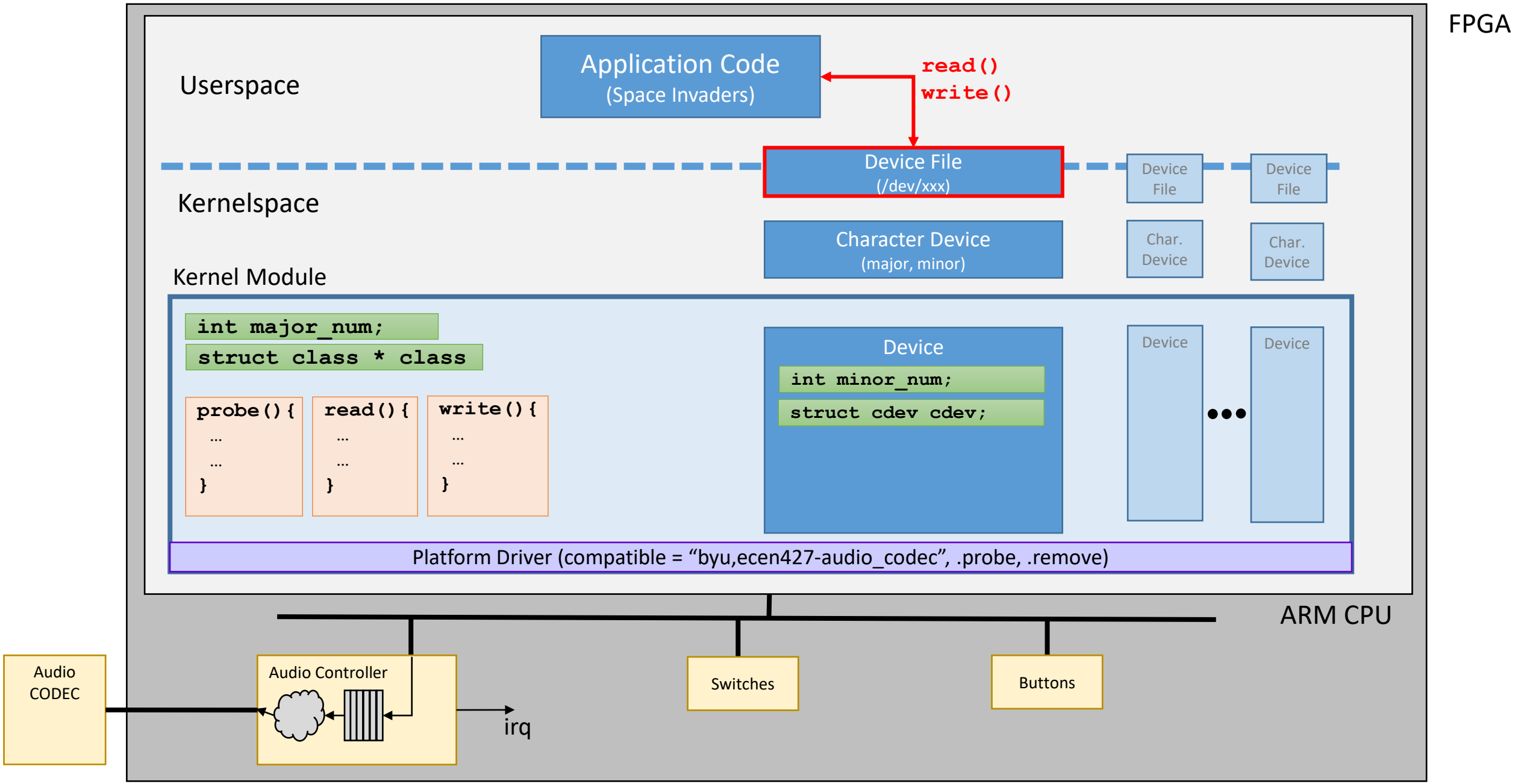
The device file (/dev/xxx) is an interface to a **character device**.

Steps:

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



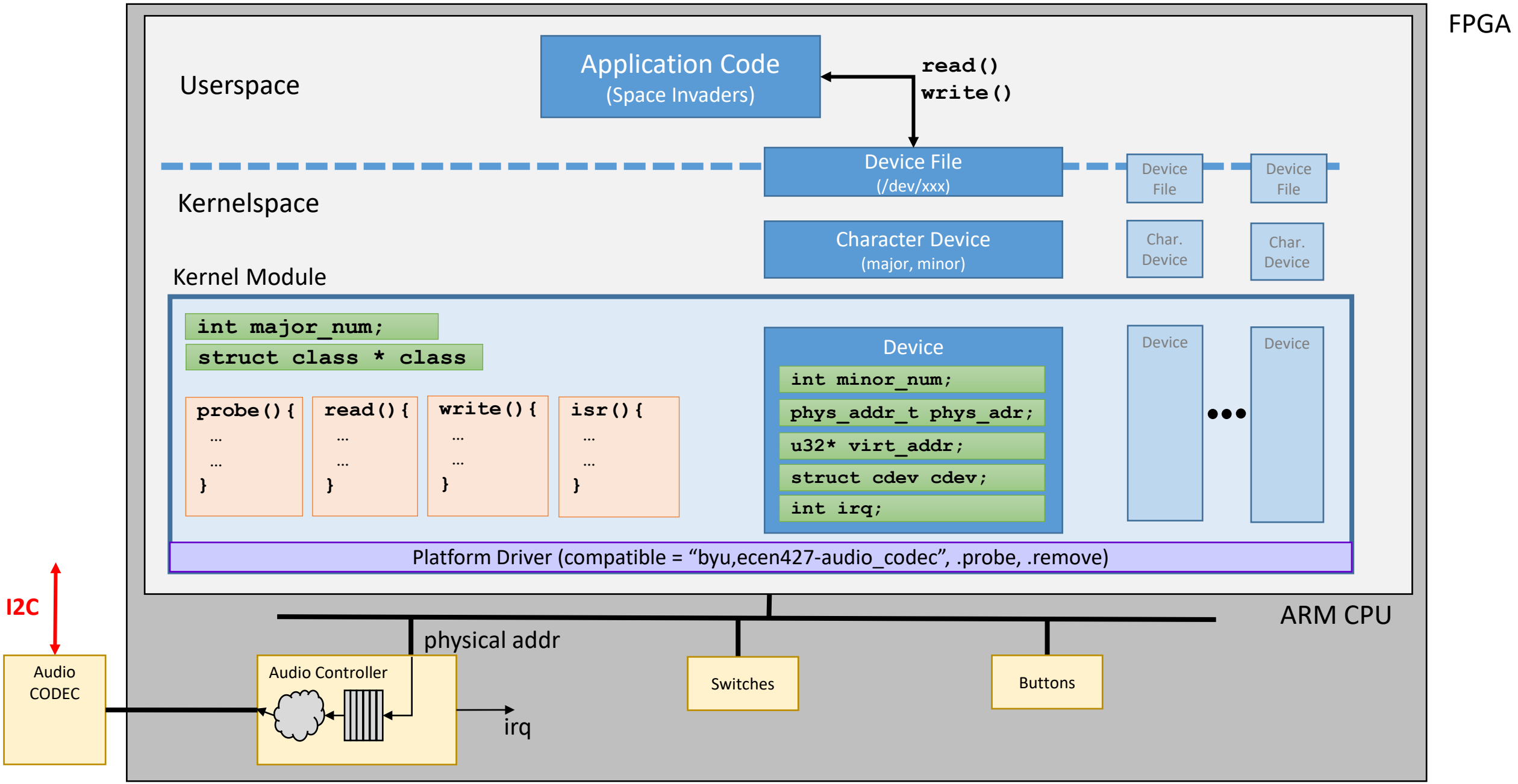

```
class_create(owner = THIS_MODULE, "my class name")
```

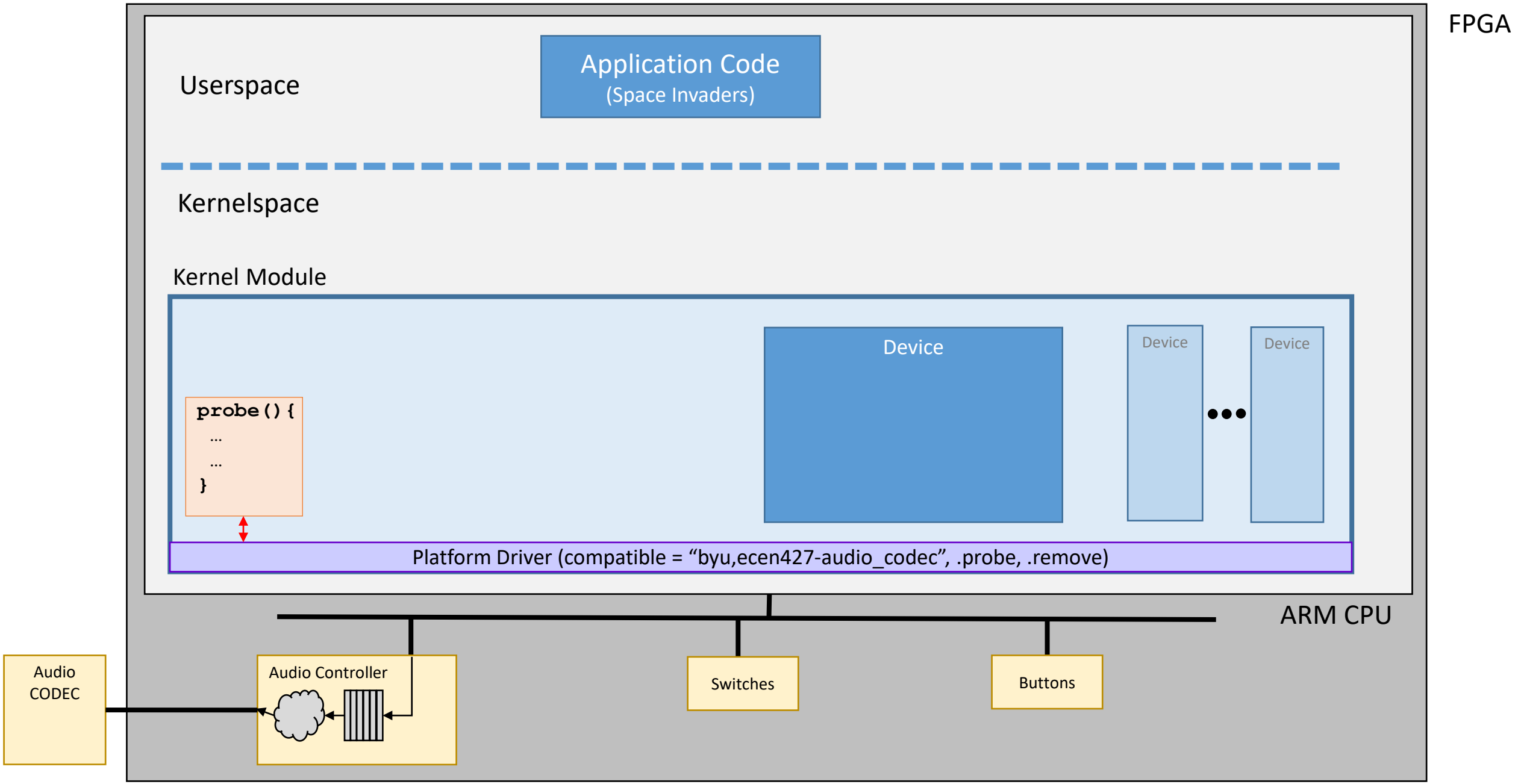


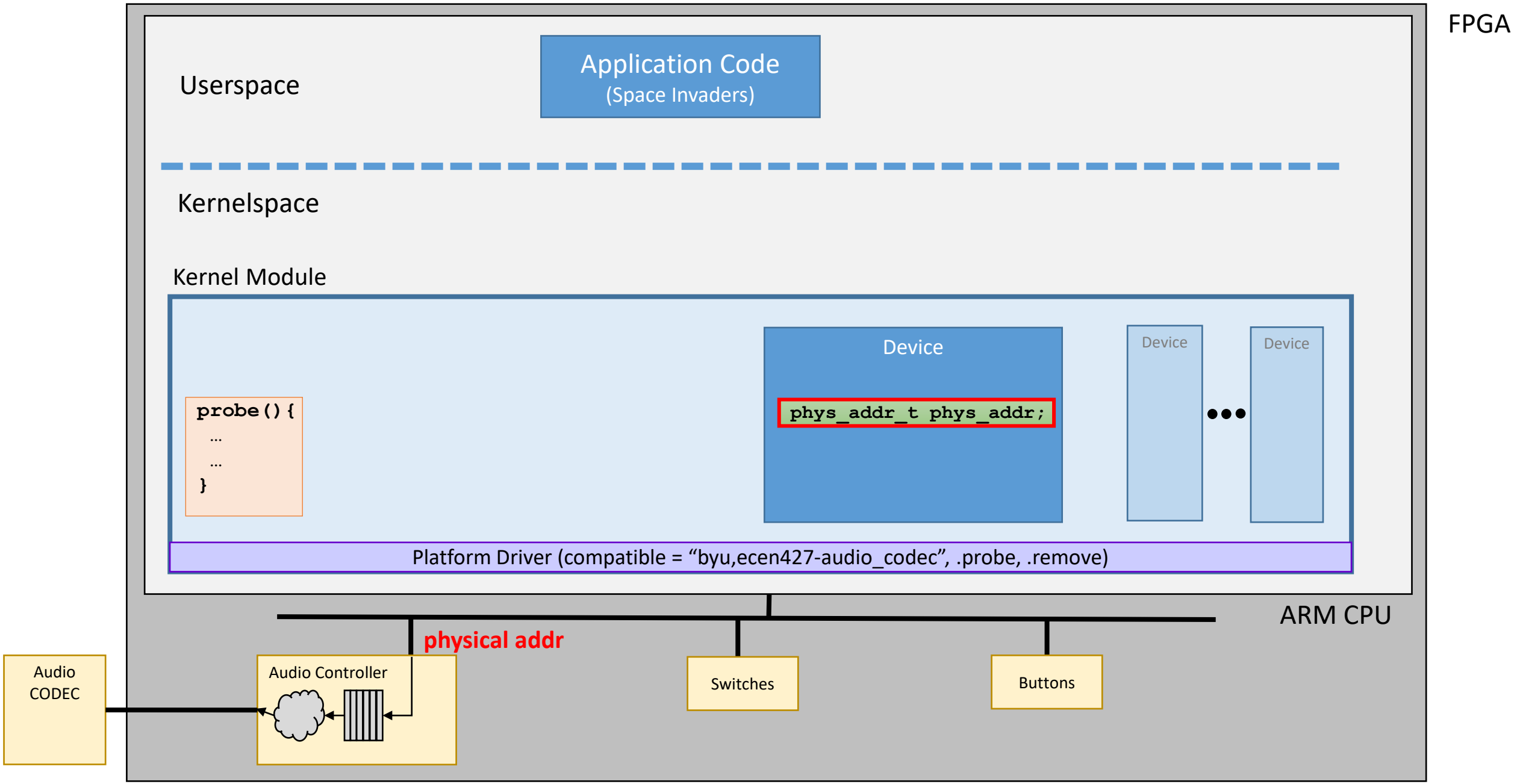

```
device_create (struct class*, parent = NULL, dev_t, “your device name”)
```

Driver needs to talk to the hardware

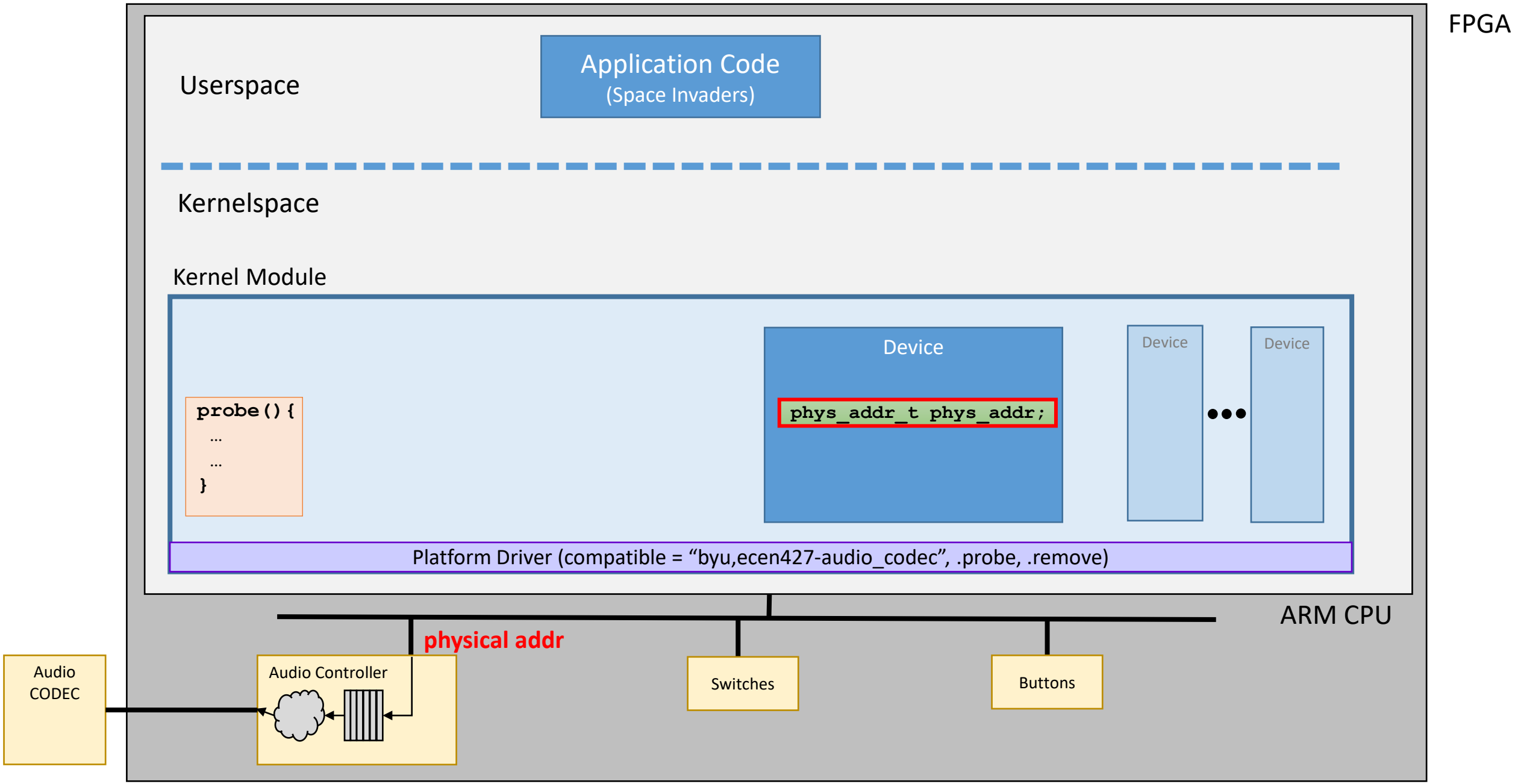
1. Need to figure out physical address
2. Need to reserve the physical address
3. Need to get a pointer (virtual address) to the physical address


FPGA
Userspace
Application Code
(Space Invaders)
read()
write()
Kernelspace
Device File
(/dev/xxx)
Device File
Device File
Character Device
(major, minor)
Char. Device
Char. Device
Kernel Module
int major_num;
struct class * class
probe(){ … … }
read(){ … … }
write(){ … … }
isr(){ … … }
Device
int minor_num;
phys_addr_t phys_adr;
u32* virt_addr;
struct cdev cdev;
int irq;
Device
Device
Platform Driver (compatible = “byu,ecen427-audio_codec”, .probe, .remove)
ARM CPU
I2C
Audio CODEC
physical addr
Audio Controller
irq
Switches
Buttons

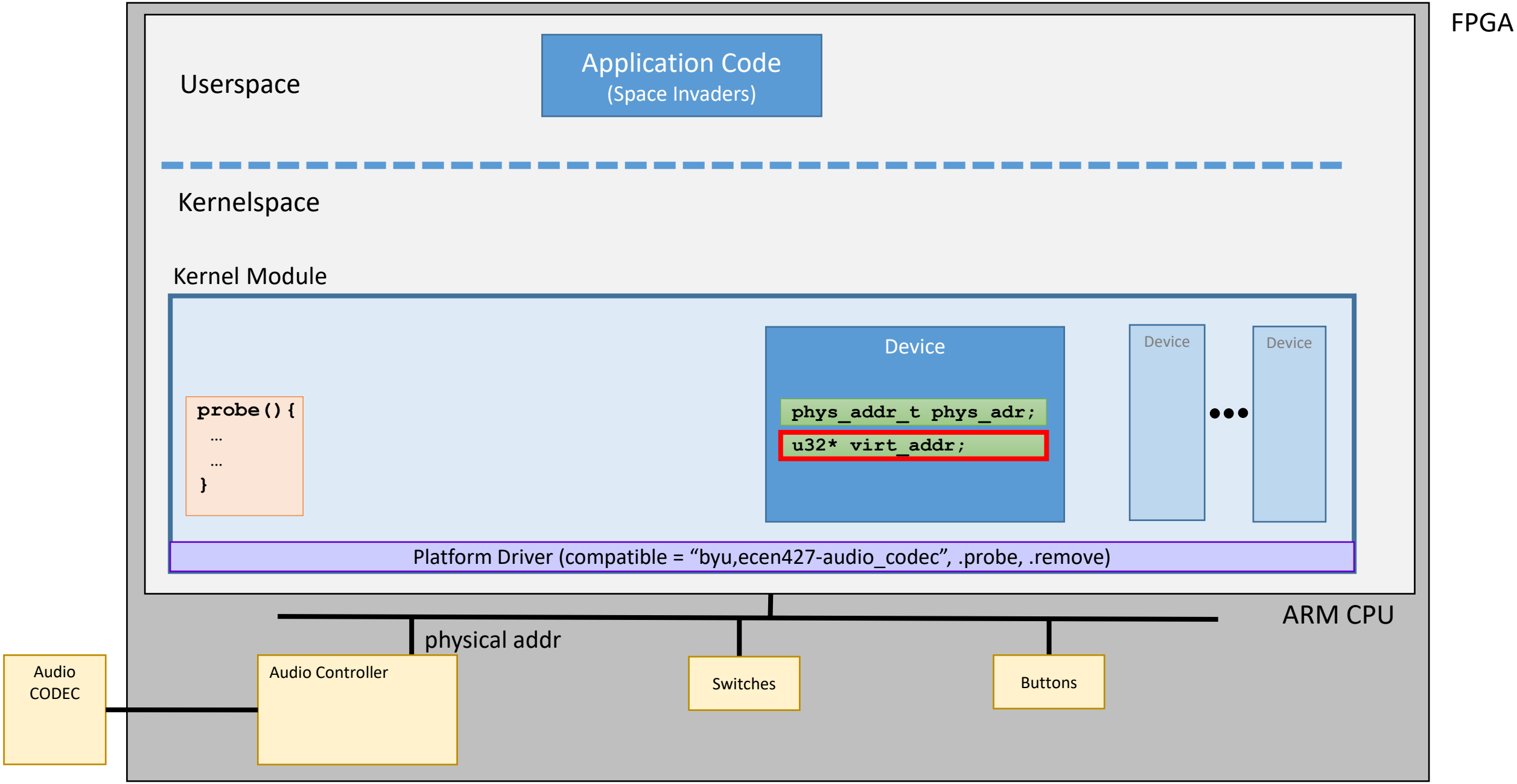

FPGA
Userspace
Application Code
(Space Invaders)
Kernelspace
Kernel Module
probe() {
...
...
}
Device
Device
Device
Platform Driver (compatible = “byu,ecen427-audio_codec”, .probe, .remove)
ARM CPU
Audio
CODEC
Audio Controller
Switches
Buttons




```
platform_get_resource(struct plaform_device * dev, IORESOURCE_MEM, 0);
```




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

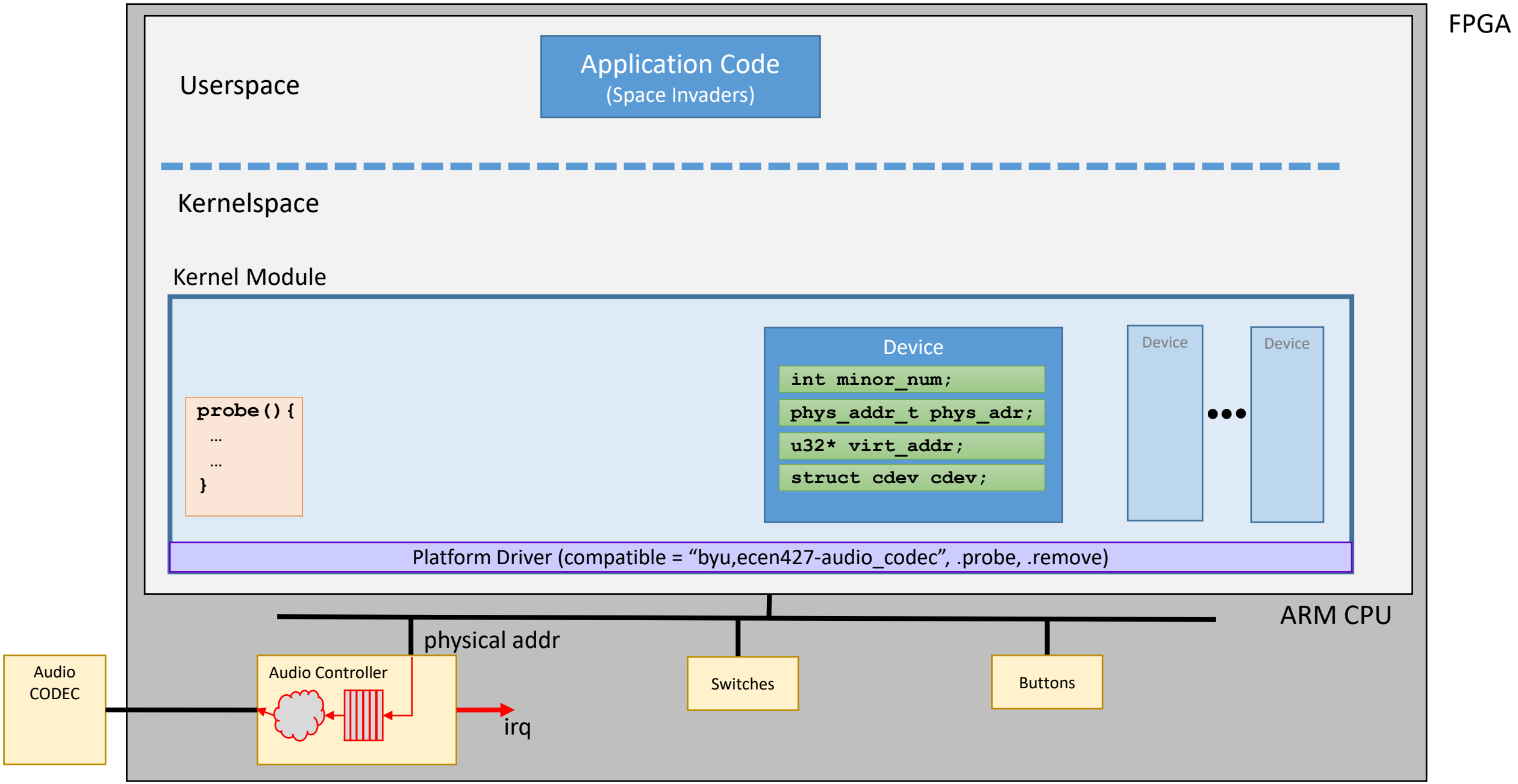



```
virt_addr = ioremap(phys_addr, size);
```

Driver needs to talk to the hardware

1. Need to figure out physical address
2. Need to reserve the physical address
3. Need to get a pointer (virtual address) to the physical address
4. Talk to the hardware with:
   - iowrite32 (value, virt_addr + offset)
   - ioread32(virt_addr + offset)

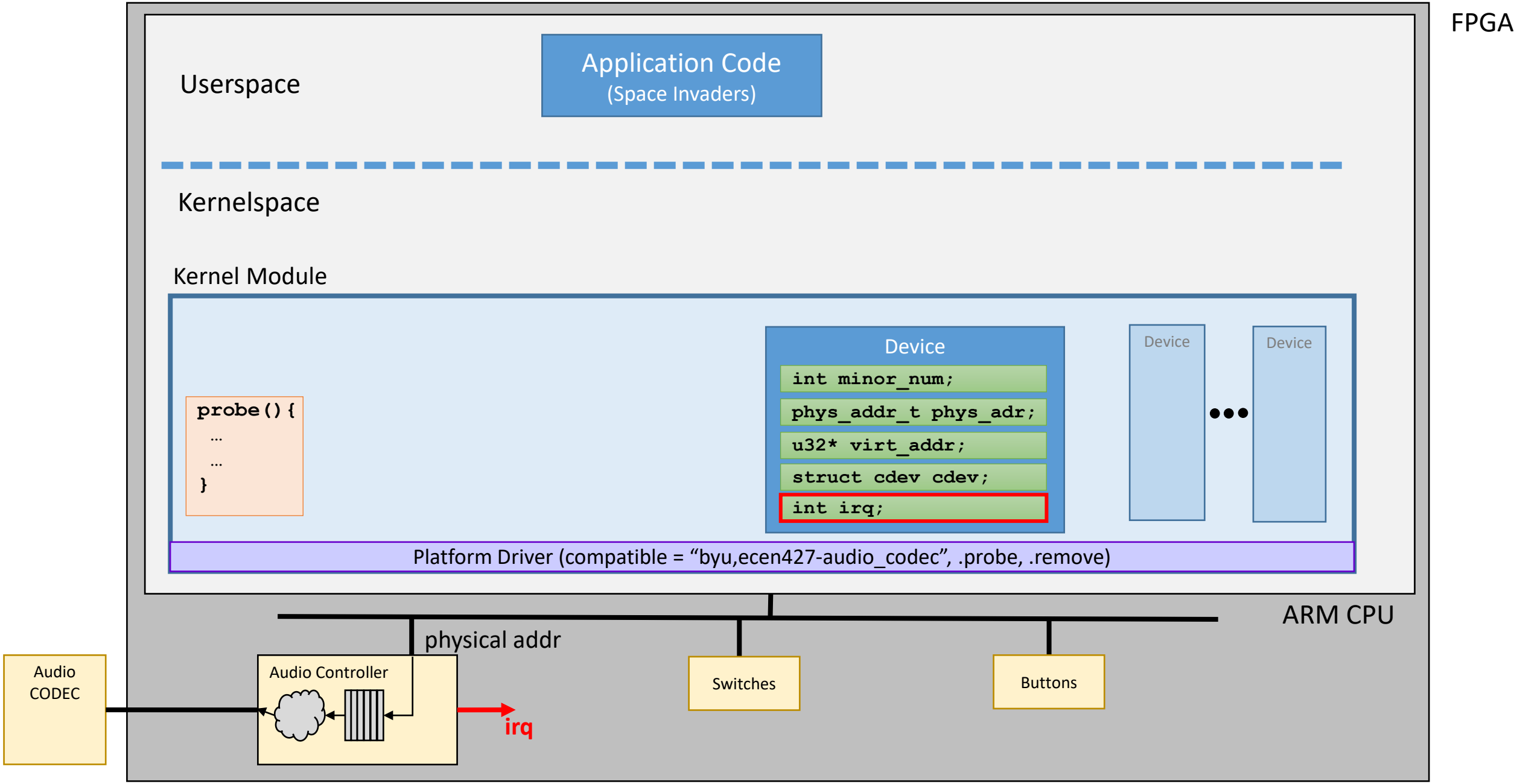
Driver Needs to Handle Interrupts


FPGA
Userspace
Application Code
(Space Invaders)
Kernelspace
Kernel Module
probe(){
...
...
}
Device
int minor_num;
phys_addr_t phys_adr;
u32* virt_addr;
struct cdev cdev;
Device
Device
Platform Driver (compatible = “byu,ecen427-audio_codec”, .probe, .remove)
ARM CPU
physical addr
Audio
CODEC
Audio Controller
irq
Switches
Buttons

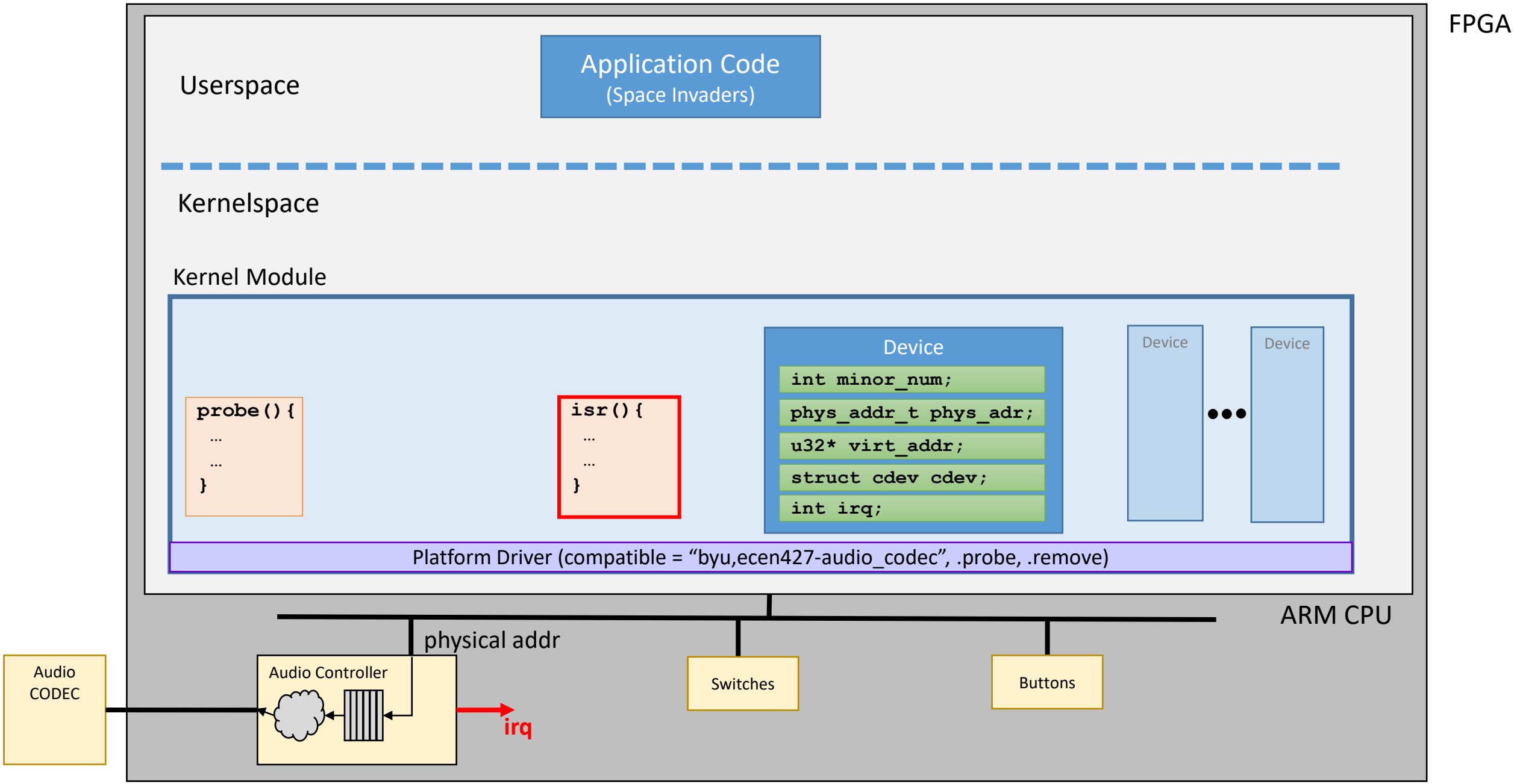
Driver Needs to Handle Interrupts

1. Get IRQ Number

2. Register Interrupt Handler with Linux




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




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

Driver Needs to Handle Interrupts

1. Get IRQ Number

2. Register Interrupt Handler with Linux