```
/**
 * struct uio_info - UIO device capabilities
 * @uio_dev:            the UIO device this info belongs to
 * @name:               device name
 * @version:            device driver version
 * @mem:                list of mappable memory regions, size==0 for end of list
 * @port:               list of port regions, size==0 for end of list
 * @irq:                interrupt number or UIO_IRQ_CUSTOM
 * @irq_flags:          flags for request_irq()
 * @priv:               optional private data
 * @handler:            the device's irq handler
 * @mmap:               mmap operation for this uio device
 * @open:               open operation for this uio device
 * @release:            release operation for this uio device
 * @irqcontrol:         disable/enable irqs when 0/1 is written to /dev/uioX
 */
struct uio_info {
struct uio_device       *uio_dev;
        const char              *name;
        const char              *version;
        struct uio_mem          mem[MAX_UIO_MAPS];
        struct uio_port         port[MAX_UIO_PORT_REGIONS];
        long                    irq;
        unsigned long           irq_flags;
        void                    *priv;
        irqreturn_t (*handler)(int irq, struct uio_info *dev_info);
        int (*mmap)(struct uio_info *info, struct vm_area_struct *vma);
        int (*open)(struct uio_info *info, struct inode *inode);
        int (*release)(struct uio_info *info, struct inode *inode);
        int (*irqcontrol)(struct uio_info *info, s32 irq_on);
```

```
31.  };
```

uio_info structure是create uio device时要提供的structure。

in drivers/uio/uio_pdrv_genirq.c

```
1.  static int uio_pdrv_genirq_probe(struct platform_device *pdev)
2.  {
3.          struct uio_info *uioinfo = dev_get_platdata(&pdev->dev);
4.   ......
5.  }
```

driver framwork在binding uio device 和uio_pdrv_genirq uio driver后，调用uio_pdrv_genirq_probe()来初始化

device。

这里一上来就从struct device的platform_data field获取uio_info instance,这个instance是在哪儿被创建的呢？

这行code有点奇怪，难道在uio_pdrv_genirq_probe()之外还有create uio_info的地方？ 不理解！

---

uio_info instance中的信息都是在uio_pdrv_genirq_probe()赋值的。

```
struct uio_info {
        struct uio_device       *uio_dev;
        const char              *name;
        const char              *version;
        struct uio_mem          mem[MAX_UIO_MAPS];
        struct uio_port         port[MAX_UIO_PORT_REGIONS];
        long                    irq;
        unsigned long           irq_flags;
        void                    *priv;
        irqreturn_t (*handler)(int irq, struct uio_info *dev_info);
        int (*mmap)(struct uio_info *info, struct vm_area_struct *vma);
        int (*open)(struct uio_info *info, struct inode *inode);
        int (*release)(struct uio_info *info, struct inode *inode);
        int (*irqcontrol)(struct uio_info *info, s32 irq_on);
};
```

上面标红的field的信息其实都来自于dts中的uio device中的hardware info description。

```
        pip_irq@f9100000  {
                compatible = "generic-uio";
                id = <0>;
                reg = <0 0xf9100000 0 0x10000>;
                interrupt-parent = <&gic>;
                interrupts = < 0 206 4 >;
        };
```

dts device info

--> device_node

--> create platform device and record these info in struct platform_device

--> create uio_info instance

```
struct uio_info {
        struct uio_device       *uio_dev;
        const char              *name;
        const char              *version;
        struct uio_mem          mem[MAX_UIO_MAPS];
        struct uio_port         port[MAX_UIO_PORT_REGIONS];
        long                    irq;
        unsigned long           irq_flags;
        void                    *priv;
irqreturn_t (*handler)(int irq, struct uio_info *dev_info);
int (*mmap)(struct uio_info *info, struct vm_area_struct *vma);
int (*open)(struct uio_info *info, struct inode *inode);
int (*release)(struct uio_info *info, struct inode *inode);
int (*irqcontrol)(struct uio_info *info, s32 irq_on);
};
```

这些callback function的初始化是在不同类型的uio device driver中，比如uio_pdrv_genirq driver中

```
        uioinfo->handler = uio_pdrv_genirq_handler;
        uioinfo->irqcontrol = uio_pdrv_genirq_irqcontrol;
        uioinfo->open = uio_pdrv_genirq_open;
        uioinfo->release = uio_pdrv_genirq_release;
        uioinfo->priv = priv;
```

uio_pdrv_genirq driver没有用到mmap callback。

irqreturn_t (*handler)(int irq, struct uio_info *dev_info);

这是uio device的2nd interrupt handler。

int (*open)(struct uio_info *info, struct inode *inode);

int (*release)(struct uio_info *info, struct inode *inode);

int (*irqcontrol)(struct uio_info *info, s32 irq_on);

这3个callback functions分别会被uio device的open / close / write operation调用到。