```
struct pxa_i2c {
        spinlock_t              lock;
        wait_queue_head_t       wait;
        struct i2c_msg          *msg;
        unsigned int            msg_num;
        unsigned int            msg_idx;
        unsigned int            msg_ptr;

    ......

};
```

```
struct i2c_msg          *msg;
```

当前正在处理的i2c_msg

```
unsigned int            msg_num;
```

还剩多少i2c_msg没处理。当msg_num为0,表示所有i2c_msg都处理完了。也就是处理了msg[msg_num] array.

```
unsigned int            msg_idx;
```

相对要处理的msg[msg_num] array,msg_idx表示正在处理的msg的index。

```
unsigned int            msg_ptr;
```

```
struct i2c_msg {
        __u16 addr;     /* slave address                        */
        __u16 flags;
        __u16 len;              /* msg length                           */
        __u8 *buf;              /* pointer to msg data                  */
};
```

每个i2c_msg指向[buf, buf + len)的buffer，而msg_ptr则指向buf中当前正在处理的位置，即当前正在发送/接收 buf + msg_ptr。

比如在i2c_pxa_do_xfer()中开始transfer以前

```
1.          i2c->msg = msg;                 ①
2.          i2c->msg_num = num;
3.          i2c->msg_idx = 0;
4.          i2c->msg_ptr = 0;
5.          i2c->irqlogidx = 0;
6.
7.          i2c_pxa_start_message(i2c);
8.
9.          spin_unlock_irq(&i2c->lock);
10.
11.         /*
12.          * The rest of the processing occurs in the interrupt handler.
13.          */
14.         timeout = wait_event_timeout(i2c->wait, i2c->msg_num == 0, HZ * 5);     ②
15.         i2c_pxa_stop_message(i2c);
```

①
初始化这4个field
②
transfer结束的判断标志是i2c->msg_num == 0，即所有i2c_msg都处理了。