eeprom chip attach在i2c bus上，对eeprom本身的操作也是有格式的。
i2c mesage的struct如下

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

这里addr是eeprom device在i2c bus上的address，[buf, buf + len)是发送给eeprom的数据。
对eeprom device中的ram的寻址也是通过i2c_msg来实现的。

比如要修改eeprom的第8字节偏移开始的 8 个字节，则实际上必须 2 个i2c_msg来实现该write operation。

1. write 1st i2c_msg that contains offset
2. write 2nd i2c_msg that conatains data

i2c-tools-3.1.2 package中eeprom.c

```c
/* read len bytes stored in eeprom at address addr, offset offset in array buf */
/* return -1 on error, 0 on success */
int eeprom_read(int fd,
                unsigned int addr,
                unsigned int offset,        ①
                unsigned char *buf,
                unsigned char len
){
        struct i2c_rdwr_ioctl_data msg_rdwr;
        struct i2c_msg             i2cmsg;
        int i;

        if(len>MAX_BYTES){
            fprintf(stderr,"I can only write MAX_BYTES bytes at a time!\n");
            return -1;
        }

        if(eeprom_write(fd,addr,offset,NULL,0)<0)     ②
            return -1;

        msg_rdwr.msgs = &i2cmsg;
        msg_rdwr.nmsgs = 1;

        i2cmsg.addr  = addr;
        i2cmsg.flags = I2C_M_RD;
        i2cmsg.len   = len;
        i2cmsg.buf   = buf;

        if((i=ioctl(fd,I2C_RDWR,&msg_rdwr))<0){     ③
            perror("ioctl()");
            fprintf(stderr,"ioctl returned %d\n",i);
            return -1;
        }

        fprintf(stderr,"Read %d bytes from eeprom at 0x%02x, offset %08x\n",
                len,addr,offset);

        return 0;
}
```

①
这里的offset是指eeprom device内的RAM的偏移
②
先把offset通过write operation发送给eeprom device,也就是要从eeprom的哪儿开始读取len bytes
③
这才是真正读取的内容

所以一般读取eeprom某个offset处的一个byte，code大致如下

```c
static int read_reg(struct i2c_client *client, unsigned char offset, unsigned char *data)
{
    int ret;

    struct i2c_msg msgs[] = {
        {
            .addr   = client->addr,
            .flags  = 0,
            .len    = 1,
            .buf    = &offset,
        },
        {
            .addr   = client->addr,
            .flags  = I2C_M_RD,
            .len    = 1,
            .buf    = data,
        },
    };

    ret = i2c_transfer(client->adapter, msgs, 2);  // 这里 num = 2,通信成功 ret = 2
    if (ret < 0)
        tp_err("%s error: %d\n", __func__, ret);

    return ret;
}

或者

static unsigned char read_reg(struct i2c_client *client, unsigned char offset)
{
    unsigned char buf;

    i2c_master_send(client, &offset, 1);        // 发送寄存器地址
    i2c_master_recv(client, &buf, 1);           // 接收寄存器的值

    return  buf;
}
```