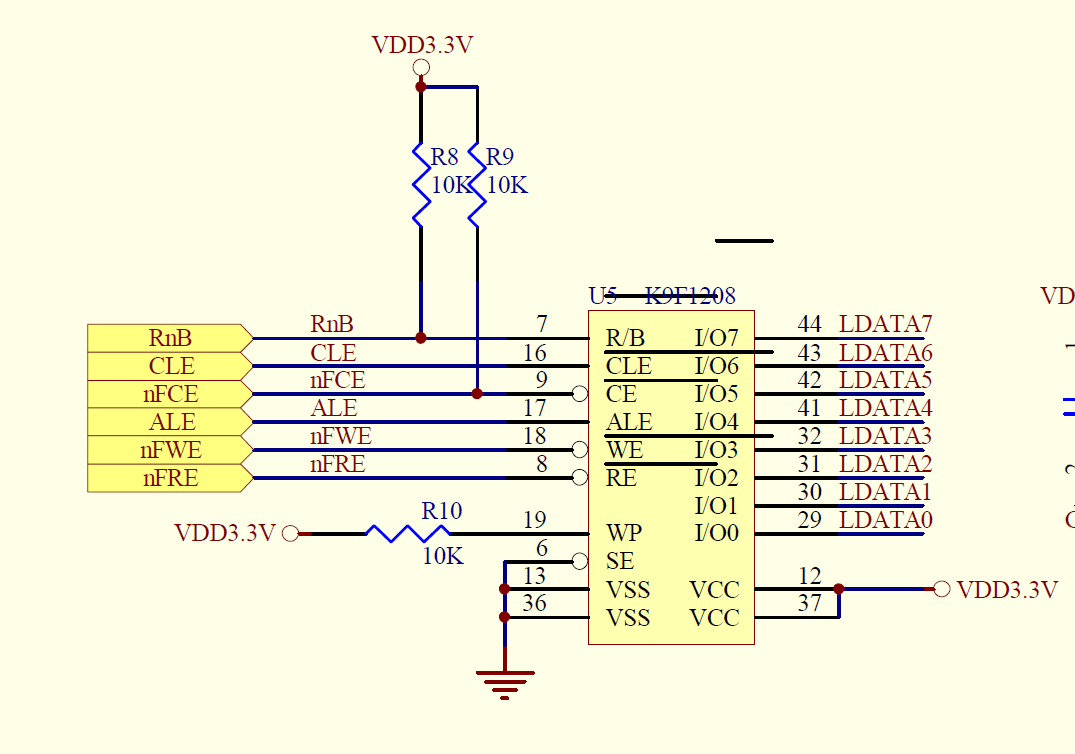
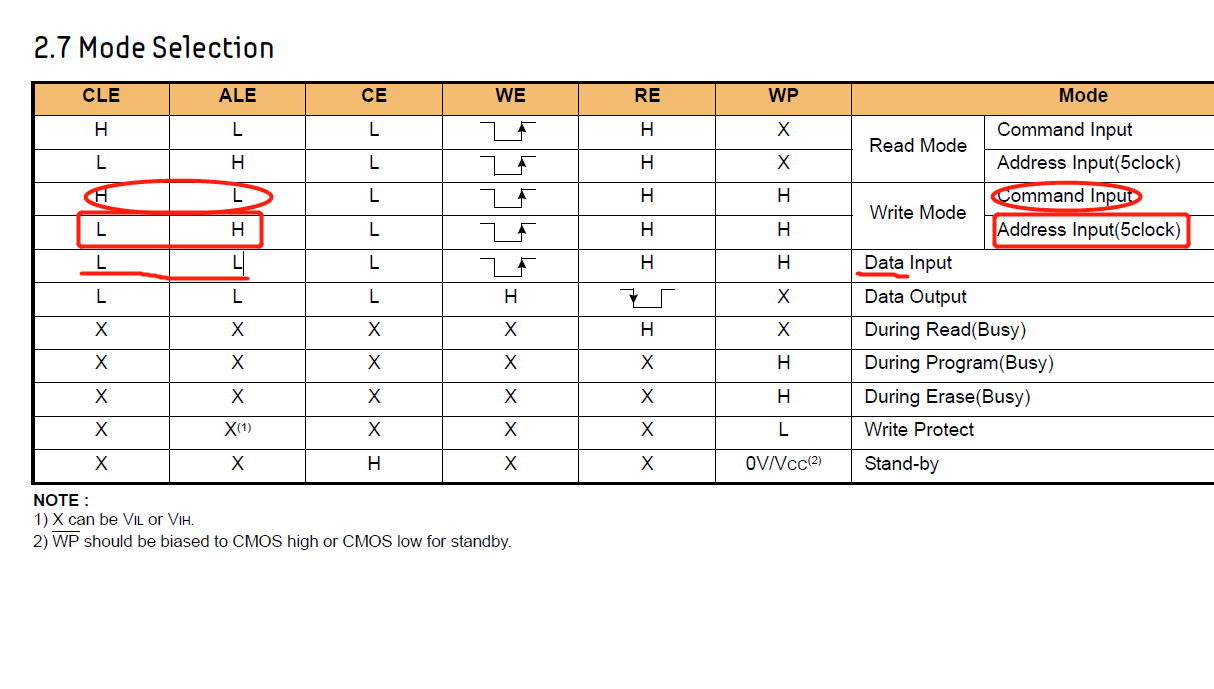
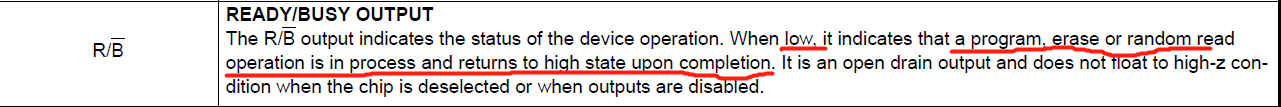
## nand\_flash简单操作原理



通过原理图可以看到nor和cpu只连接了数据线并没有地址线，因此在LDATA线上传送数据，地址和命令

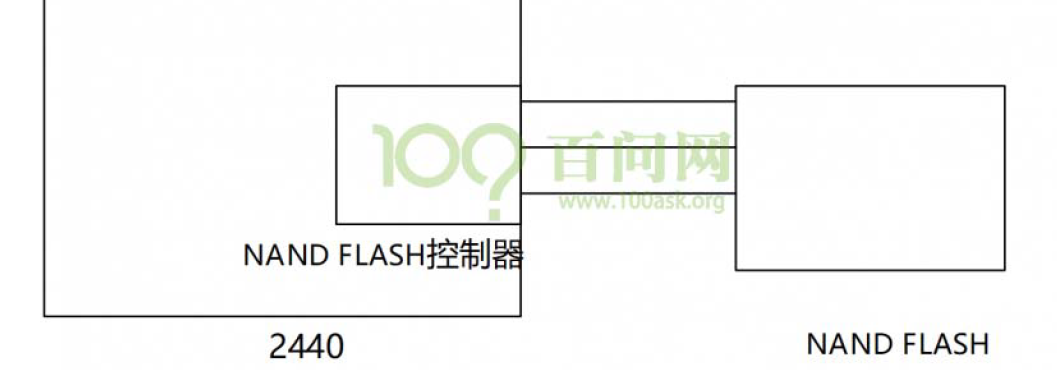


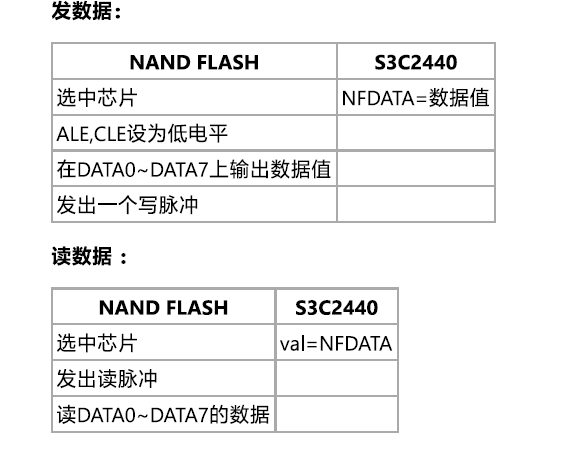
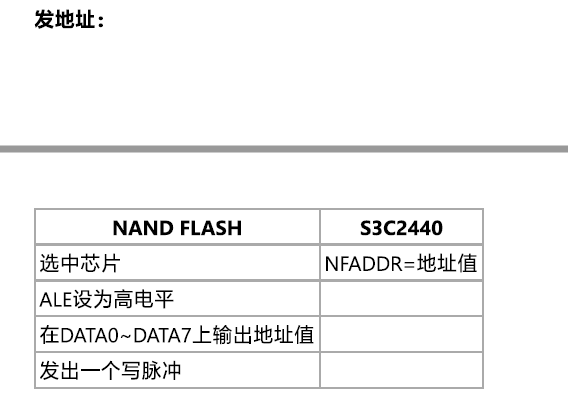
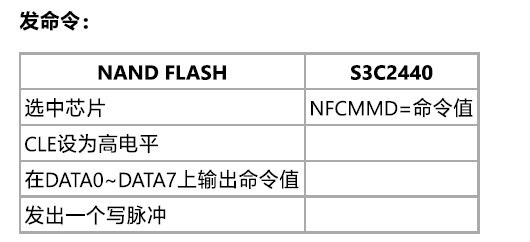
可以看到 当CLE和ALE引脚的高低电平决定了LDATA传输的内容

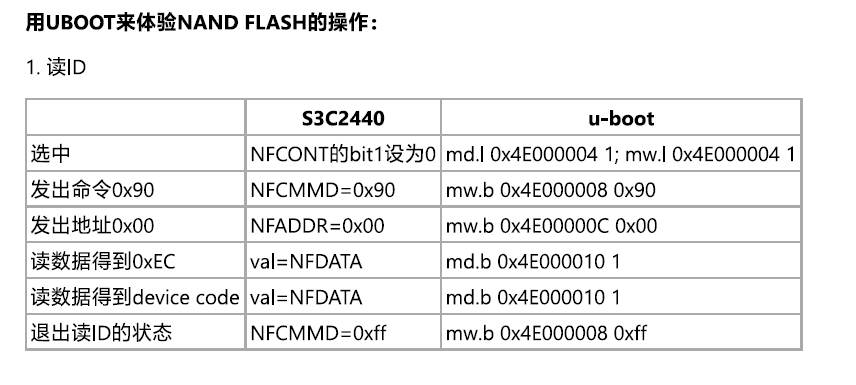
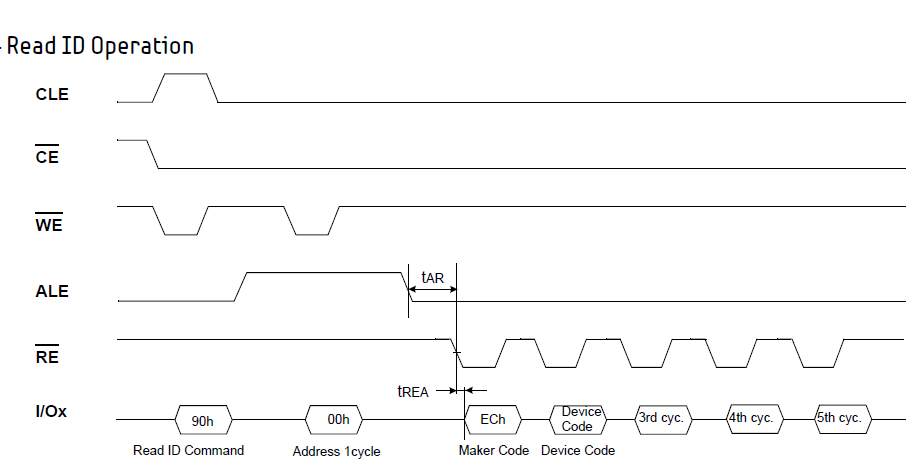


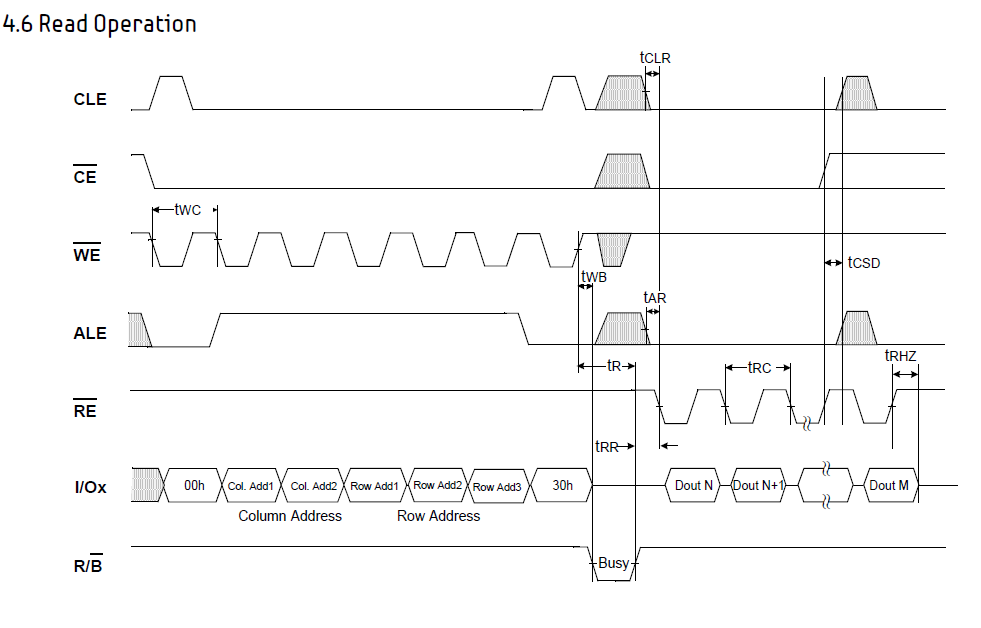
RnB引脚则用来判断操作是否完成，1完成，0未完成

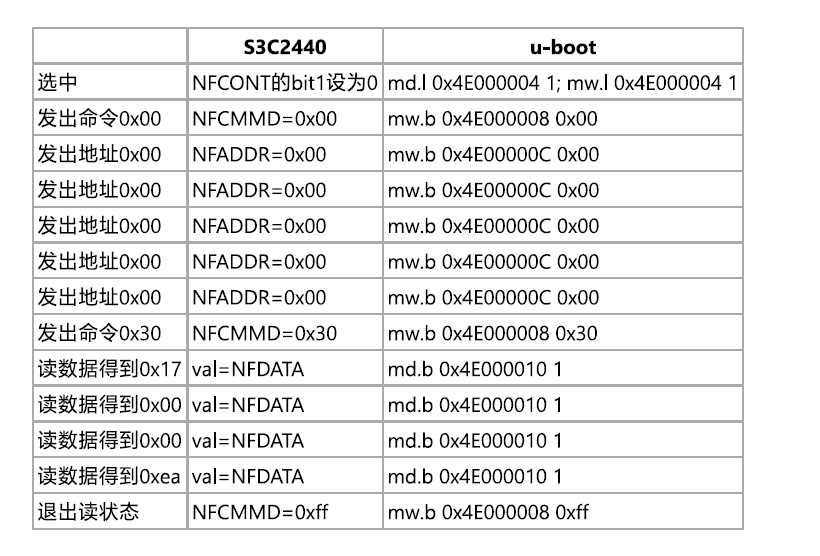
nand\_flash控制器简化了对nand\_flash的操作



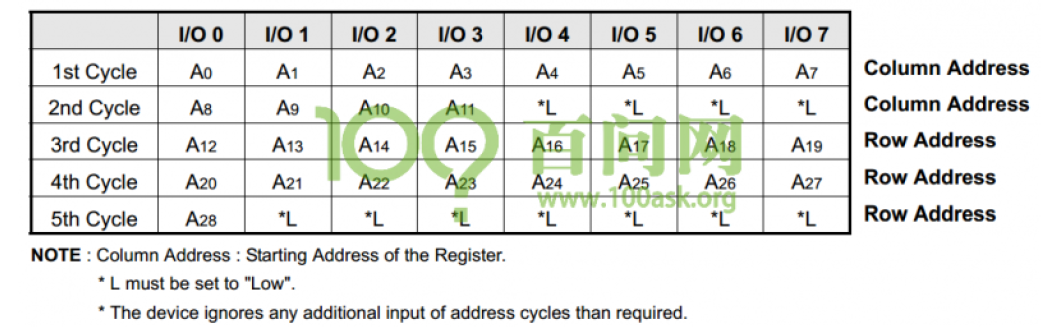






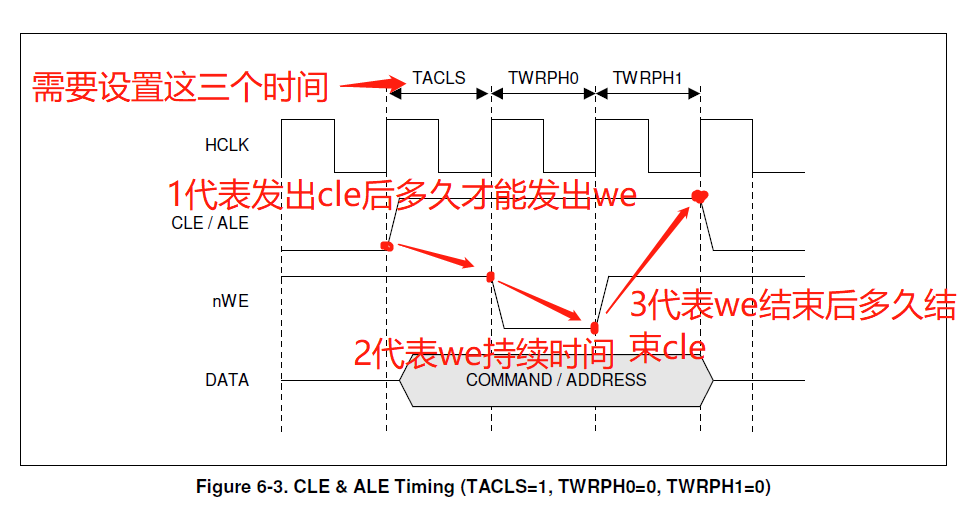


以256M大小的Nand flash需要28根地址线，因此最少需要4个周期，为了兼容更大的内存，用5个周期发送地址

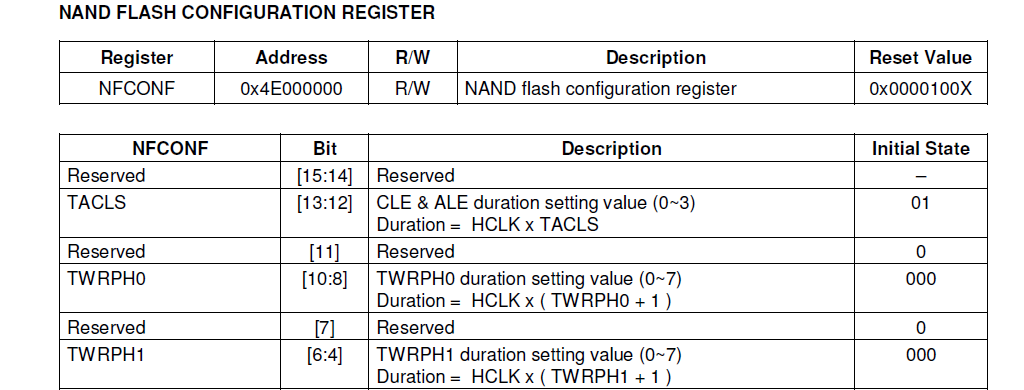


## nand\_flash初始化

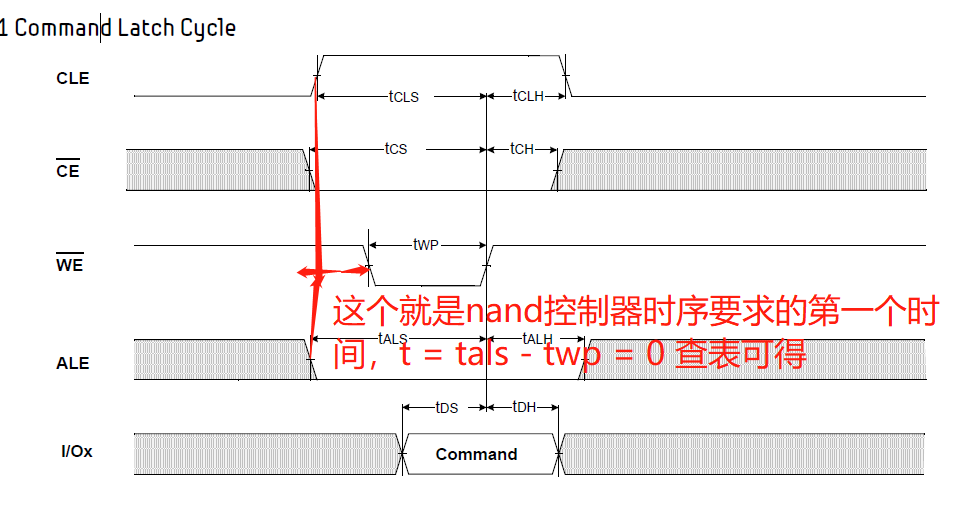
由于外接的nand\_flash不同，nand\_flash控制器对nand\_flash的控制方法也不用，因为每个nand\_flash的时序都不同

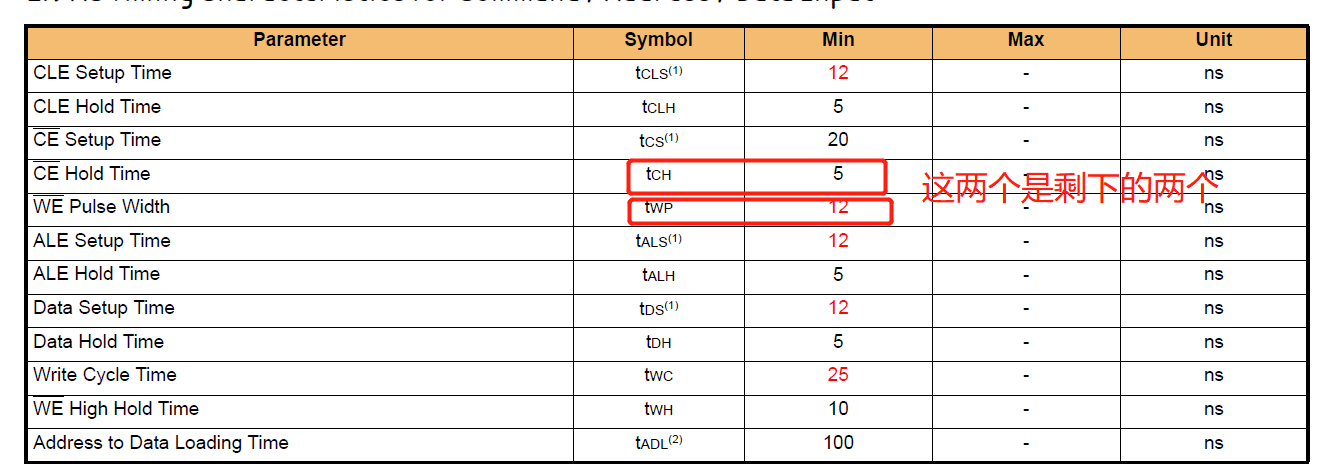


找到设置这三个时间的寄存器

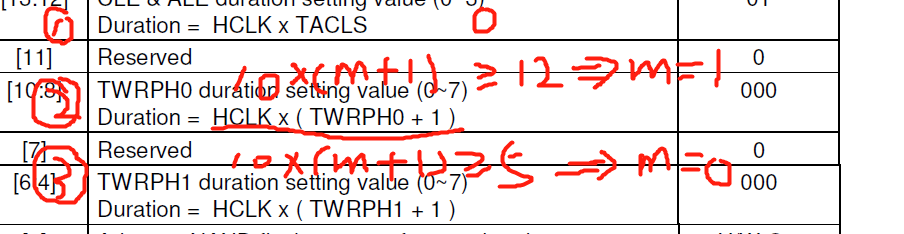


前面说过nand\_flash控制器是根据不用的nand\_flash来设定的，所以还要知道nand\_flash的时序

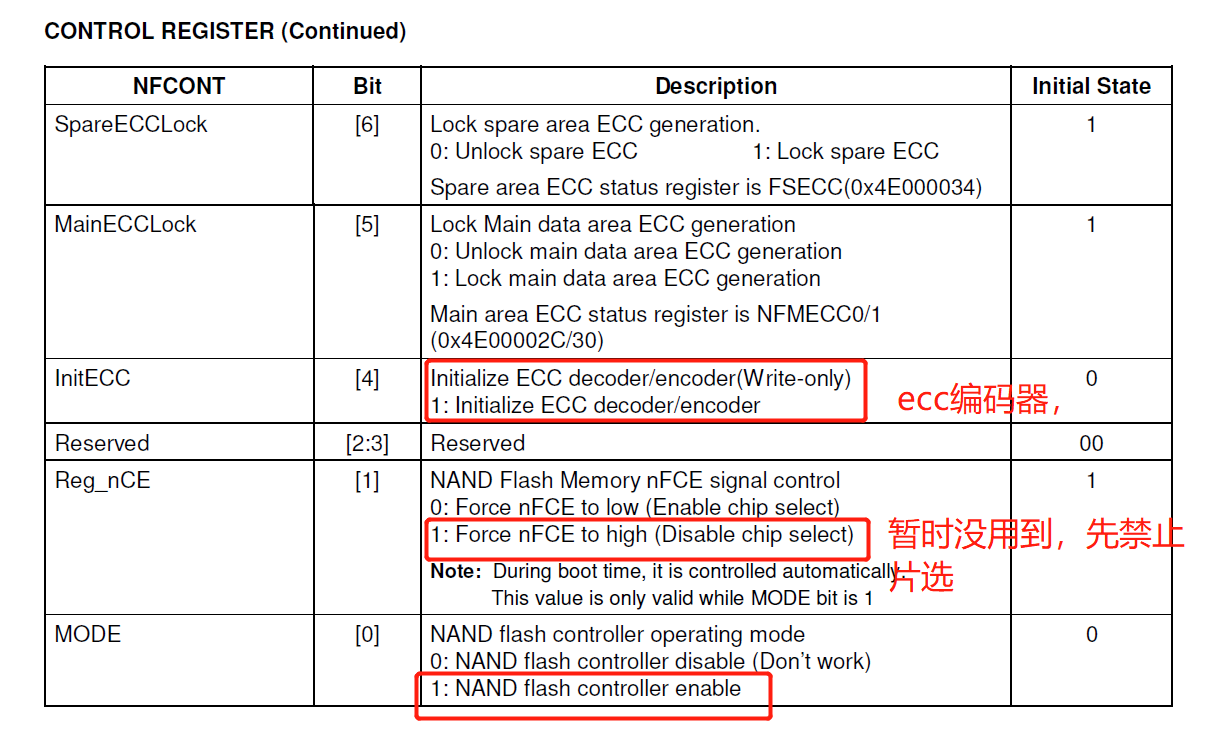




所以最后可以得出



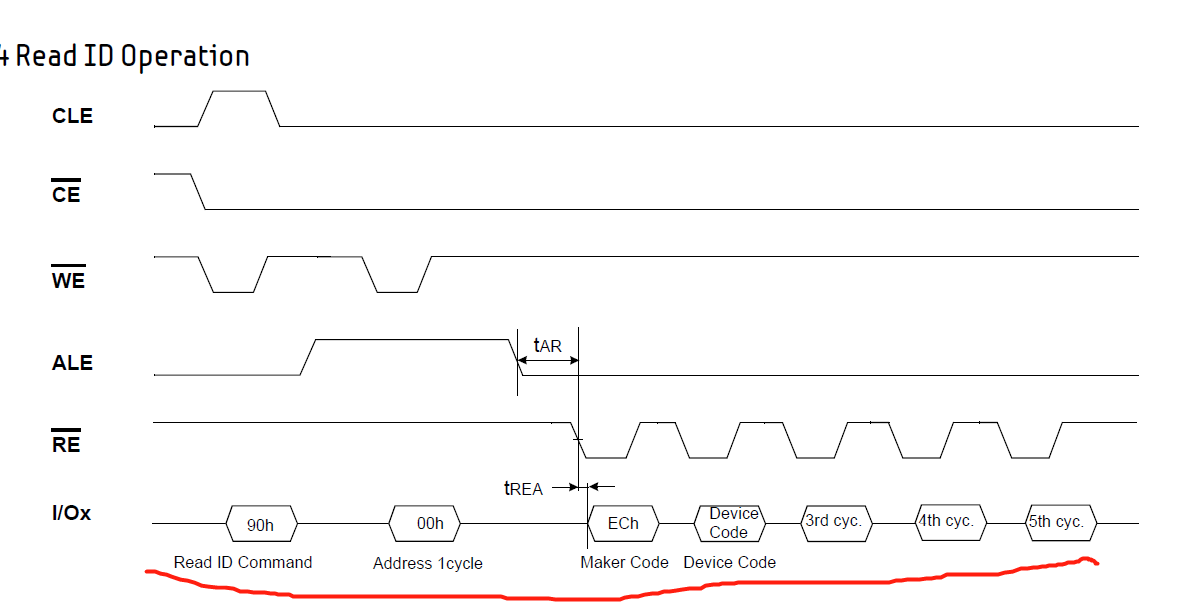
时序设置完了，需要将nand\_flash使能



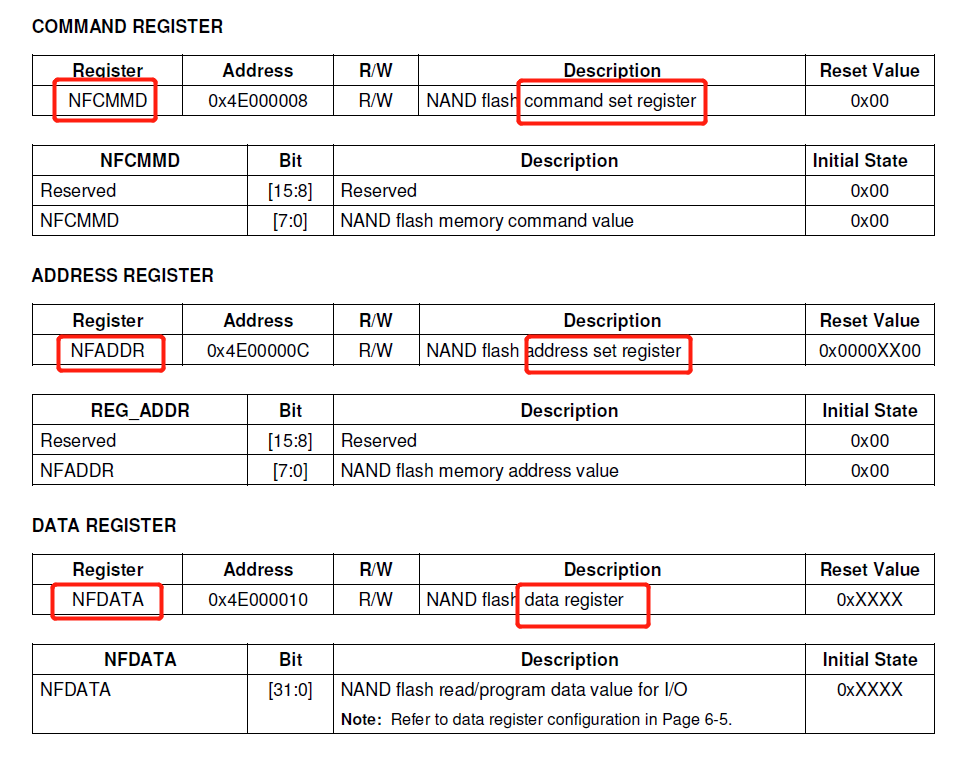
## nand\_flash ID读取

前面没有开启片选

这里先写nand\_select,nand\_diselect两个函数



通过时序图可以了解操作时序，nand\_flash控制器的存在简化了很多操作，只需对相应的寄存器进行操作就可以达到时序图一样的效果



可以写出nand\_cmd,nand\_addr,nand\_data三个函数

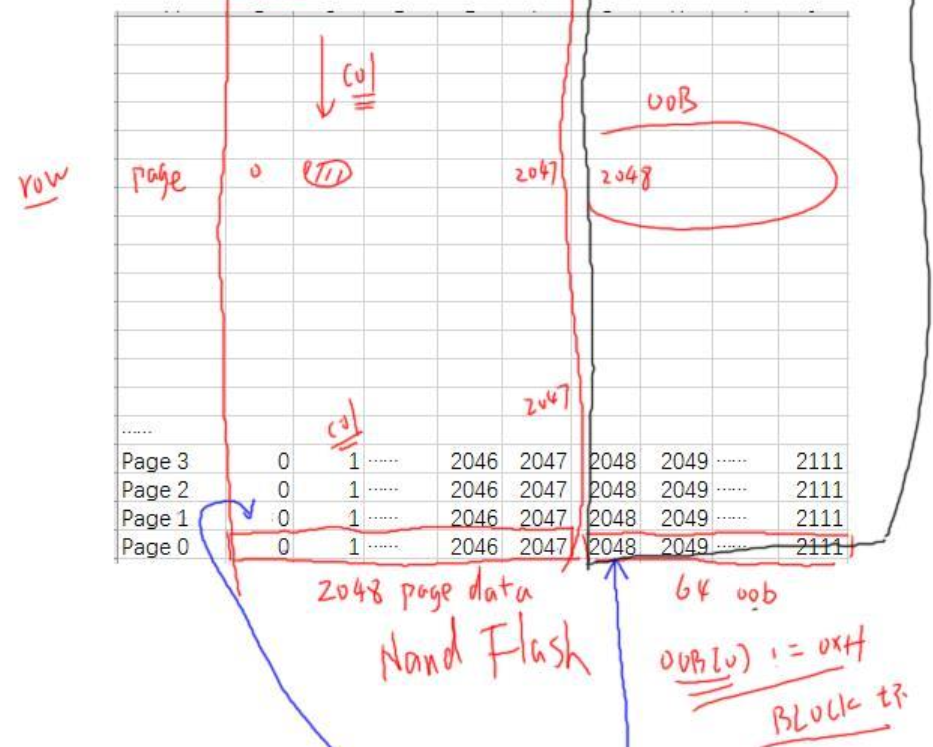
结合时序图最后一行进行读取id操作nand\_chip\_id

再提供一个目录，按键调用函数nand\_test，赋值nor\_test作修改

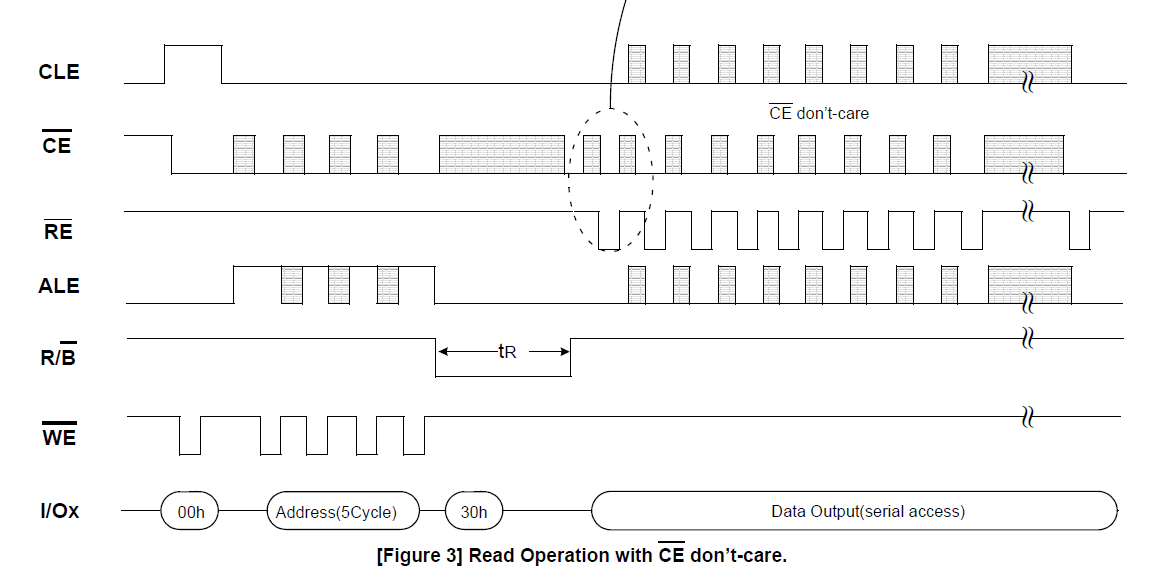
## nand\_flash的数据读取

nand\_flash 有个缺点就是位反转，为了解决这个问题，nand引入oob区，在写page数据时，会在out of bank 区生成校验码，读数据时根据校验码来判断数据是否出错

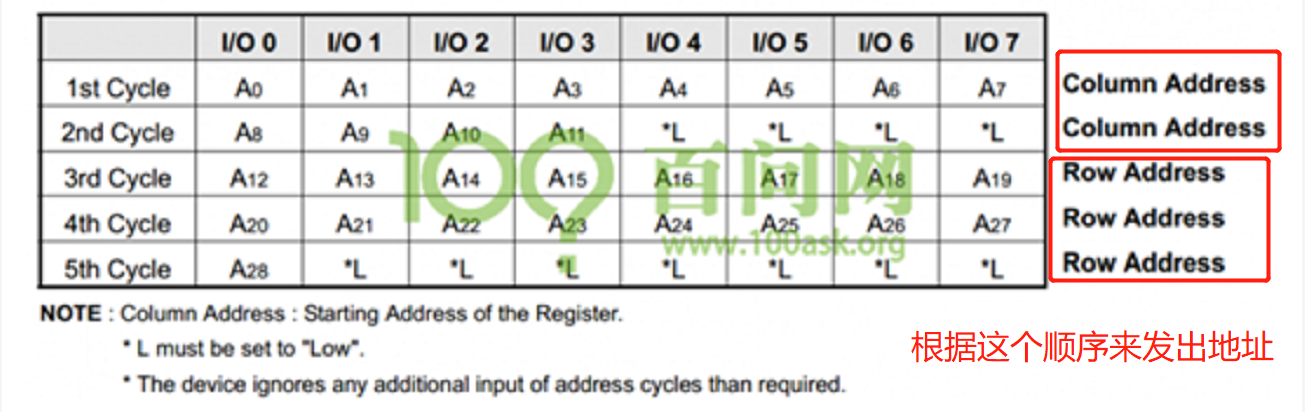
cpu不操作oob区，

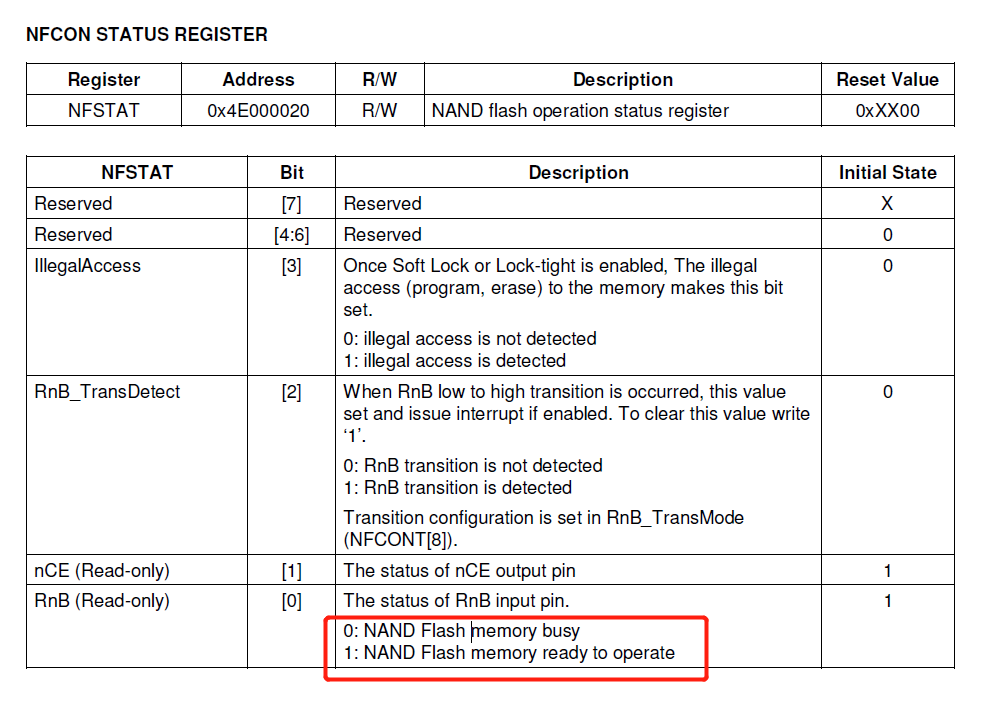


读操作序列，直接看最后一行，



发出00后发出address，上面有一张图





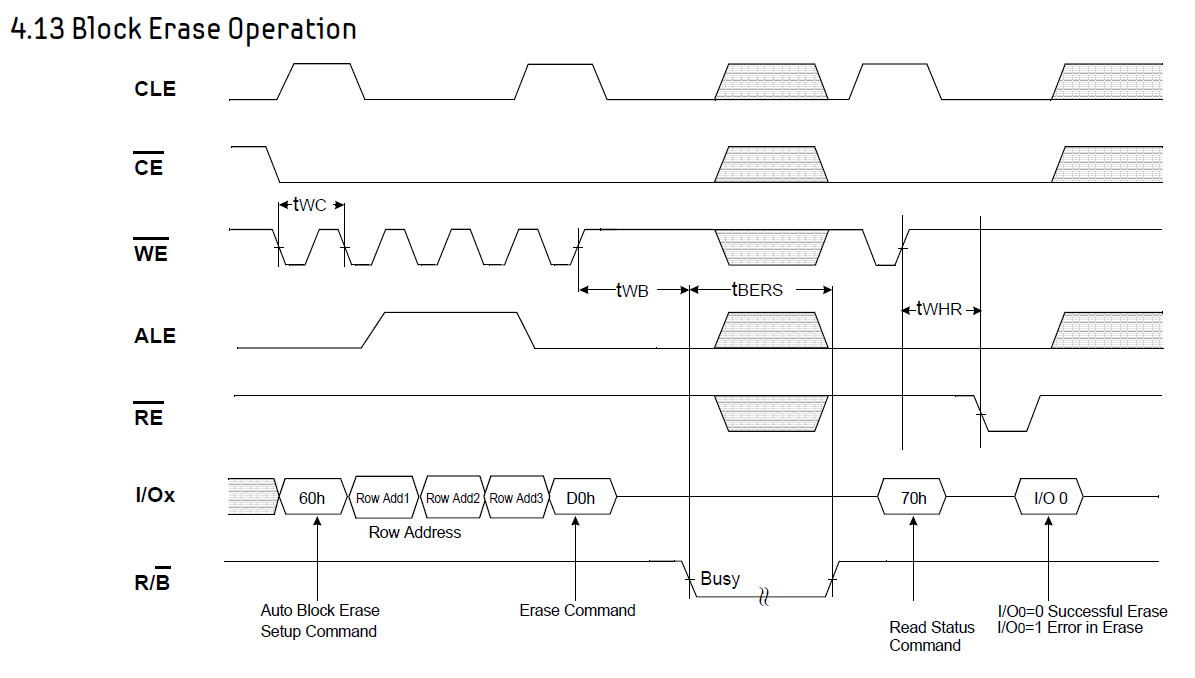
再根据这个寄存器写wait函数

最后来写nand\_read, do\_nand\_read函数

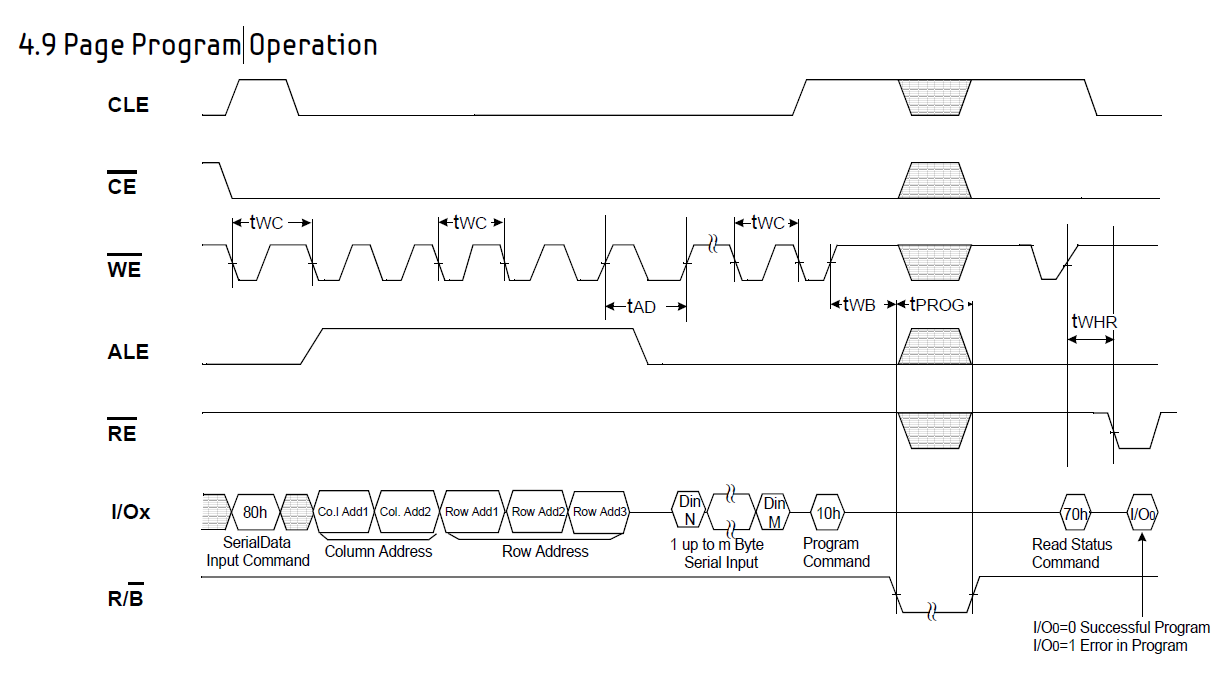
因为之前的代码重定位是在nor执行的，现在的程序大于4k，所以这里要在init.c中添加一个判断是否为nand启动的函数isBootFromNorFlash，如果是nand启动，那重定位的代码也要改，

## nand\_flash的擦除与烧写

### 擦除



nand擦除是一个快一个块（128k）擦除的，



先写nand\_w\_data,把要写的val值传递到NFDATA寄存器里，然后再根据时序图操作，传递完成地址后调用nand\_w\_data函数，逐个写