软盘控制器的编程方法

赵炯　oldlinux.org (gohigh@sh163.net)

1.1 软盘驱动器的设备号

在Linux中，软驱的主设备号是2，次设备号 = TYPE*4 + DRIVE，其中DRIVE为0-3，分别对应软驱A、B、C或D；TYPE是软驱的类型，2表示1.2M软驱，7表示1.44M软驱，也即floppy.c中85行定义的软盘类型（floppy_type[]）数组的索引值：

表6.1 软盘驱动器类型

类型	说明
0	不用。
1	360KB PC 软驱。
2	1.2MB AT 软驱。
3	360kB 在 720kB 驱动器中使用。
4	3.5″ 720kB 软盘。
5	360kB 在 1.2MB 驱动器中使用。
6	720kB 在 1.2MB 驱动器中使用。
7	1.44MB 软驱。

例如，因为7*4 + 0 = 28，所以 /dev/PS0 (2,28)指的是1.44M A驱动器,其设备号是0x021c。

同理 /dev/at0 (2,8)指的是1.2M A驱动器，其设备号是0x0208。

1.2 软盘控制器

对软盘控制器的编程比较烦琐。在编程时需要访问4个端口，分别对应一个或多个寄存器。对于1.2M的软盘控制器有以下一些端口。

表6.2 软盘控制器端口

I/O 端口	读写性	寄存器名称
0x3f2	只写	数字输出寄存器（DOR）(数字控制寄存器)
0x3f4	只读	FDC 主状态寄存器(STATUS)
0x3f5	读/写	FDC 数据寄存器(DATA)
0x3f7	只读	数字输入寄存器（DIR）
	只写	磁盘控制寄存器(DCR)(传输率控制)

数字输出端口DOR（数字控制端口）是一个8位寄存器，它控制驱动器马达开启、驱动器选择、启动/复位FDC以及允许/禁止DMA及中断请求。

表6.3 数字输出寄存器定义

位	名称	说明

7	MOT_EN3	启动软驱D马达：1-启动；0-关闭。
6	MOT_EN2	启动软驱C马达：1-启动；0-关闭。
5	MOT_EN1	启动软驱B马达：1-启动；0-关闭。
4	MOT_EN0	启动软驱A马达：1-启动；0-关闭。
3	DMA_INT	允许DMA和中断请求；0-禁止DMA和中断请求。
2	RESET	允许软盘控制器FDC工作。0-复位FDC。
1	DRV_SEL1	00-11用于选择软盘驱动器A-D。
0	DRV_SEL0	

FDC的主状态寄存器也是一个8位寄存器，用于反映软盘控制器FDC和软盘驱动器FDD的基本状态。通常，在CPU向FDC发送命令之前或从FDC获取操作结果之前，都要读取主状态寄存器的状态位，以判别当前FDC数据寄存器是否就绪，以及确定数据传送的方向。

表6.4 FDC主状态控制器MSR定义

位	名称	说明
7	RQM	数据口就绪：控制器FDC数据寄存器已准备就绪。
6	DIO	传输方向：1- FDC CPU；0- CPU FDC
5	NDM	非DMA方式：1- 非DMA方式；0- DMA方式
4	CB	控制器忙：FDC正处于命令执行忙碌状态
3	DDB	软驱D忙
2	DCB	软驱C忙
1	DBB	软驱B忙
0	DAB	软驱A忙

FDC的数据端口对应多个寄存器(只写型命令寄存器和参数寄存器、只读型结果寄存器)，但任一时刻只能有一个寄存器出现在数据端口0x3f5。在访问只写型寄存器时，主状态控制的DIO方向位必须为0（CPU FDC），访问只读型寄存器时则反之。在读取结果时只有在FDC不忙之后才算读完结果，通常结果数据最多有7个字节。

数据输入寄存器（DIR）只有位7（D7）对软盘有效，用来表示盘片更换状态。其余七位用于硬盘控制器接口。

磁盘控制寄存器(DCR)用于选择盘片在不同类型驱动器上使用的数据传输率。仅使用低2位（D1D0），
00 - 500kbps，01 - 300kbps，10 - 250kbps。

Linux 0.11内核中，驱动程序与软驱中磁盘之间的数据传输是通过DMA控制器实现的。在进行读写操作之前，需要首先初始化DMA控制器，并对软驱控制器进行编程。对于386兼容PC，软驱控制器使用硬件中断IR6（对应中断描述符0x26），并采用DMA控制器的通道2。有关DMA控制处理的内容见后面小节。

1.3 软盘控制器命令

软盘控制器共可以接受15条命令。每个命令均经历三个阶段：命令阶段、执行阶段和结果阶段。

命令阶段是CPU向FDC发送命令字节和参数字节。每条命令的第一个字节总是命令字节（命令码）。其后跟着0--8字节的参数。

执行阶段是FDC执行命令规定的操作。在执行阶段CPU是不加干预的，一般是通过FDC发出中断请求获知命令执行的结束。如果CPU发出的FDC命令是传送数据，则FDC可以以中断方式或DMA方式进行。中断方式每次传送1字节。DMA方式是在DMA控制器管理下，FDC与内存进行数据的传输直至全部数据传送完。此时DMA控制器会将传输字节计数终止信号通知FDC，最后由FDC发出中断请求信号告知CPU执行阶段结束。

结果阶段是由CPU读取FDC数据寄存器返回值，从而获得FDC命令执行的结果。返回结果数据的长度为0--7字节。对于没有返回结果数据的命令，则应向FDC发送检测中断状态命令获得操作的状态。

由于Linux 0.11的软盘驱动程序中只使用其中6条命令，因此这里仅对这些用到的命令进行描述。

1. 重新校正命令（FD_RECALIBRATE）

该命令用来让磁头退回到0磁道。通常用于在软盘操作出错时对磁头重新校正定位。其命令码是0x07，参数是指定的驱动器号（0—3）。

该命令无结果阶段，程序需要通过执行“检测中断状态”来获取该命令的执行结果。

表6.5 重新校正命令（FD_RECALIBRATE）

阶段	序	D7	D6	D5	D4	D3	D2	D1	D0	说明
命令	0	0	0	0	0	0	1	1	1	重新校正命令码：0x07
	1	0	0	0	0	0	0	US1	US2	驱动器号
执行										磁头移动到0磁道
结果		无。								需使用命令获取执行结果。

2. 磁头寻道命令（FD_SEEK）

该命令让选中驱动器的磁头移动到指定磁道上。第1个参数指定驱动器号和磁头号，位0-1是驱动器号，位2是磁头号，其它比特位无用。第2个参数指定磁道号。

该命令也无结果阶段，程序需要通过执行“检测中断状态”来获取该命令的执行结果。

表6.6 磁头寻道命令（FD_SEEK）

阶段	序	D7	D6	D5	D4	D3	D2	D1	D0	说明
命令	0	0	0	0	0	1	1	1	1	磁头寻道命令码：0x0F
	1	0	0	0	0	0	HD	US1	US2	磁头号、驱动器号。
	2	C								磁道号。
执行										磁头移动到指定磁道上。
结果		无。								需使用命令获取执行结果。

3. 读扇区数据命令（FD_READ）

该命令用于从磁盘上读取指定位置开始的扇区，经DMA控制传输到系统内存中。每当一个扇区读完，参数4（R）就自动加1，以继续读取下一个扇区，直到DMA控制器把传输计数终止信号发送给软盘控制器。该命令通常是在磁头寻道命令执行后磁头已经位于指定磁道后开

始。

返回结果中，磁道号C和扇区号R是当前磁头所处位置。因为在读完一个扇区后起始扇区号R自动增1，因此结果中的R值是下一个未读扇区号。若正好读完一个磁道上最后一个扇区（即EOT），则磁道号也会增1，并且R值复位成1。

表6.7 读扇区数据命令（FD_READ）

阶段	序	D7	D6	D5	D4	D3	D2	D1	D0	说明
命令	0	MT	MF	SK	0	0	1	1	0	读命令码：0xE6（MT=MF=SK=1）
	1	0	0	0	0	0	0	US1	US2	驱动器号。
	2	C								磁道号
	3	H								磁头号
	4	R								起始扇区号
	5	N								扇区字节数
	6	EOT								磁道上最大扇区号
	7	GPL								扇区之间间隔长度（3）
	8	DTL								N=0时，指定扇区字节数
执行										数据从磁盘传送到系统
结果	1	ST0								状态字节0
	2	ST1								状态字节1
	3	ST2								状态字节2
	4	C								磁道号
	5	H								磁头号
	6	R								扇区号
	7	N								扇区字节数

其中MT、MF和SK的含义分别为：
MT表示多磁道操作。MT=1表示允许在同一磁道上两个磁头连续操作。
MF表示记录方式。MF=1表示选用MFM记录方式，否则是FM记录方式。
SK表示是否跳过有删除标志的扇区。SK=1表示跳过。
返回的个状态字节ST0、ST1和ST2的含义见下面所示。

表6.8 状态字节0 （ST0）

位	名称	说明
7 6	ST0_INTR	中断原因。00 - 命令正常结束；01 - 命令异常结束； 10 - 命令无效；11 - 软盘驱动器状态改变。
5	ST0_SE	寻道操作或重新校正操作结束。(Seek End)
4	ST0_ECE	设备检查出错（零磁道校正出错)。(Equip. Check Error)
3	ST0_NR	软驱未就绪。(Not Ready)
2	ST0_HA	磁头地址。中断时磁头号。(Head Address)
1 0	ST0_DS	驱动器选择号（发生中断时驱动器号)。(Drive Select) 00 - 11分别对应驱动器0—3。

表6.9 状态字节1 （ST1）

位	名称	说明
7	ST1_EOC	访问超过磁道上最大扇区号EOT。(End of Cylinder)
6		未使用（0）。
5	ST1_CRC	CRC校验出错。
4	ST1_OR	数据传输超时，DMA控制器故障。(Over Run)
3		未使用（0）。
2	ST1_ND	未找到指定的扇区。(No Data - unreadable)
1	ST1_WP	写保护。(Write Protect)
0	ST1_MAM	未找到扇区地址标志ID AM。(Missing Address Mask)

表6.10 状态字节2 （ST2）

位	名称	说明
7		未使用（0）。
6	ST2_CM	SK=0时，读数据遇到删除标志。(Control Mark = deleted)
5	ST2_CRC	扇区数据场CRC校验出错。
4	ST2_WC	扇区ID信息的磁道号C不符。(Wrong Cylinder)
3	ST2_SEH	检索（扫描）条件满足要求。(Scan Equal Hit)
2	ST2_SNS	检索条件不满足要求。(Scan Not Satisfied)
1	ST2_BC	扇区ID信息的磁道号C=0xFF，磁道坏。(Bad Cylinder)
0	ST2_MAM	未找到扇区数据标志DATA AM。(Missing Address Mask)

4. 写扇区数据命令（FD_WRITE）

该命令用于将内存中的数据写到磁盘上。在DMA传输方式下，软驱控制器把内存中的数据串行地写到磁盘指定扇区中。每写完一个扇区，起始扇区号自动增1，并继续写下一个扇区，直到软驱控制器收到DMA控制器的计数终止信号。见下表所示，其中缩写名称的含义与读命令中的相同。

表6.11 写扇区数据命令（FD_WRITE）

阶段	序	D7	D6	D5	D4	D3	D2	D1	D0	说明
命令	0	MT	MF	0	0	0	1	0	1	写数据命令码：0xC5（MT=MF=1）
	1	0	0	0	0	0	0	US1	US2	驱动器号。
	2	C								磁道号
	3	H								磁头号
	4	R								起始扇区号
	5	N								扇区字节数
	6	EOT								磁道上最大扇区号
	7	GPL								扇区之间间隔长度（3）
	8	DTL								N=0时，指定扇区字节数
执行										数据从系统传送到磁盘
结果	1	ST0								状态字节0
	2	ST1								状态字节1
	3	ST2								状态字节2

	4	C	磁道号
	5	H	磁头号
	6	R	扇区号
	7	N	扇区字节数

5. 检测中断状态命令（FD_SENSEI）

发送该命令后软驱控制器会立刻返回常规结果1和2(即状态ST0和磁头所处磁道号PCN)。它们是控制器执行上一条命令后的状态。通常在一个命令执行结束后会向CPU发出中断信号。对于读写扇区、读写磁道、读写删除标志、读标识场、格式化和扫描等命令以及非DMA传输方式下的命令引起的中断，可以直接根据主状态寄存器的标志知道中断原因。而对于驱动器就绪信号发生变化、寻道和重新校正（磁头回零道）而引起的中断，由于没有返回结果，就需要利用本命令来读取控制器执行命令后的状态信息。

表6.12 检测中断状态命令（FD_SENSEI）

阶段	序	D7	D6	D5	D4	D3	D2	D1	D0	说明
命令	0	0	0	0	0	1	0	0	0	检测中断状态命令码：0x08
执行										
结果	1	ST0								状态字节0
	2	C								磁头所在磁道号

6. 设定驱动器参数命令（FD_SPECIFY）

该命令用于设定软盘控制器内部的三个定时器初始值和选择传输方式,即把驱动器马达步进速率（STR)、磁头加载/卸载（HLT/HUT）时间和是否采用DMA方式来传输数据的信息送入软驱控制器。

表6.13 设定驱动器参数命令（FD_SPECIFY）

阶段	序	D7	D6	D5	D4	D3	D2	D1	D0	说明
命令	0	0	0	0	0	0	0	1	1	设定参数命令码：0x03
	1	SRT（单位2ms）				HUT（单位32ms）				马达步进速率、磁头卸载时间
	2	HLT（单位4ms）							ND	磁头加载时间、非DMA方式
执行										设置控制器，不发生中断
结果		无								无

1.4 软盘控制器编程方法

在PC机中，软盘控制器一般采用与NEC PD765或Intel 8287A兼容的芯片，例如Intel的82078。由于软盘的驱动程序比较复杂，因此下面对这类芯片构成的软盘控制器的编程方法进行较为详细的介绍。

典型的磁盘操作不仅仅包括发送命令和等待控制器返回结果,的软盘驱动器的控制是一种低级操作，它需要程序在不同阶段对其执行状况进行干涉。

命令与结果阶段的交互

在上述磁盘操作命令或参数发送到软盘控制器之前,必须首先查询控制器的主状态寄存

器（MSR），以获知驱动器的就绪状态和数据传输方向。软盘驱动程序中使用了一个output_byte(byte)函数来专门实现该操作。该函数的等效框图如下所示。

该函数一直循环到主状态寄存器的数据口就绪标志RQM为1，并且方向标志DIO是0（CPU FDC），此时控制器就已准备好接受命令和参数字节。循环语句起超时计数功能，以应付控制器没有响应的情况。本驱动程序中把循环次数设置成了10000次。对这个循环次数的选择需要仔细，以避免程序作出不正确的超时判断。在Linux内核版本0.1x至0.9x中就经常会碰到需要调整这个循环次数的问题，因为当时人们所使用的PC机运行速度差别较大（16MHz -- 40MHz），因此循环所产生的实际延时也有很大的区别。这可以参见早期Linux的邮件列表中的许多文章。为了彻底解决这个问题，最好能使用系统硬件时钟来产生固定频率的延时值。

对于读取控制器的结果字节串的结果阶段，也需要采取与发送命令相同的操作方法，只是此时数据传输方向标志要求是置位状态（FDC CPU）。本程序中对应的函数是result()。该函数把读取的结果状态字节存放到了reply_buffer[]字节数组中。

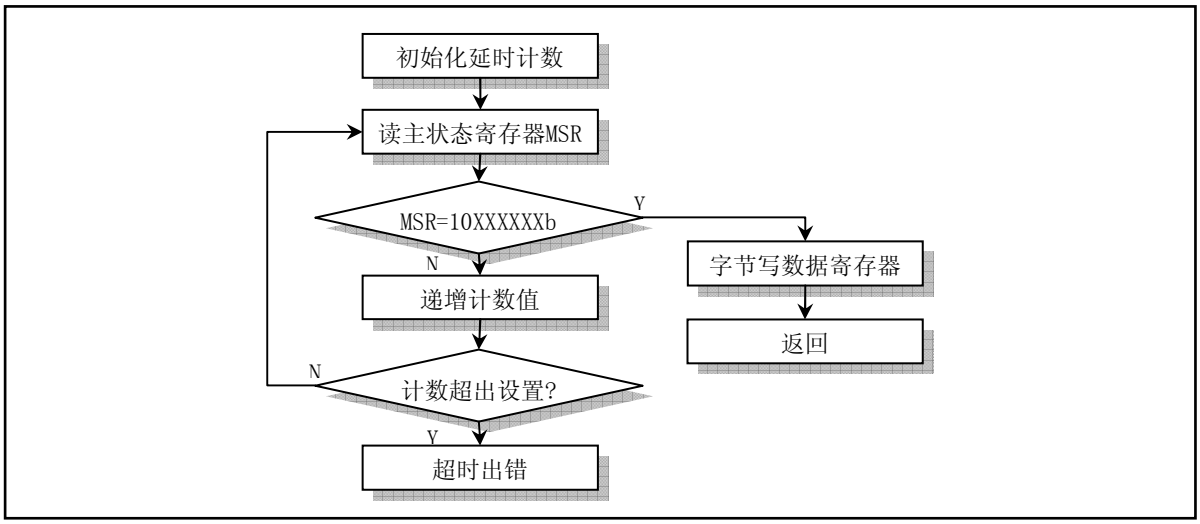

图6.1 向软盘控制器发送命令或参数字节

软盘控制器初始化

对软盘控制器的初始化操作包括在控制器复位后对驱动器进行适当的参数配置。控制器复位操作是指对数字输出寄存器DOR的位2（启动FDC标志）置0然后再置1。在机器复位之后，“指定驱动器参数”命令SPECIFY所设置的值就不再有效，需要重新建立。在floppy.c程序中，复位操作在函数reset_floppy()和中断处理C函数reset_interrupt()中。前一个函数用于修改DOR寄存器的位2，让控制器复位，后一个函数用于在控制器复位后使用SPECIFY命令重新建立控制器中的驱动器参数。在数据传输准备阶段，若判断出与实际的磁盘规格不同，还在传输函数transfer()开始处对其另行进行重新设置。

在控制器复位后，还应该向数字控制寄存器DCR发送指定的传输速率值，以重新初始化数据传输速率。如果机器执行了复位操作（例如热启动），则数据传输速率会变成默认值250Kpbs。但通过数字输出寄存器DOR向控制器发出的复位操作并不会影响设置的数据传输速率。

驱动器重新校正和磁头寻道

驱动器重新校正（FD_RECALIBRATE）和磁头寻道(FD_SEEK)是两个磁头定位命令。重新校正命令让磁头移动到零磁道，而磁头寻道命令则让磁头移动到指定的磁道上。这两个磁头

定位命令与典型的读/写命令不同，因为它们没有结果阶段。一旦发出这两个命令之一，控制器将立刻会在主状态寄存器（MSR）返回就绪状态，并以后台形式执行磁头定位操作。当定位操作完成后，控制器就会产生中断以请求服务。此时就应该发送一个“检测中断状态”命令，以结束中断和读取定位操作后的状态。由于驱动器和马达启动信号是直接由数字输出寄存器（DOR）控制的，因此，如果驱动器或马达还没有启动，那么写DOR的操作必须在发出定位命令之前进行。流程图见下图所示。

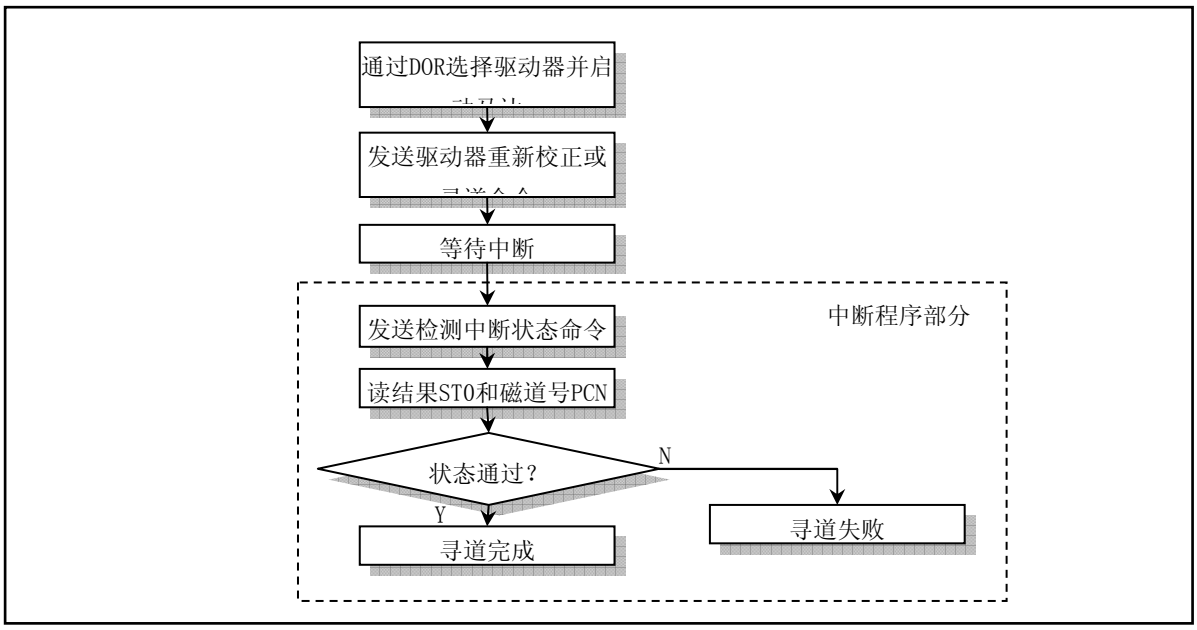

图6.2 重新校正和寻道操作

数据读/写操作

数据读或写操作需要分几步来完成。首先驱动器马达需要开启，并把磁头定位到正确的磁道上，然后初始化DMA控制器，最后发送数据读或写命令。另外，还需要定出发生错误时的处理方案。典型的操作流程图见下图所示。

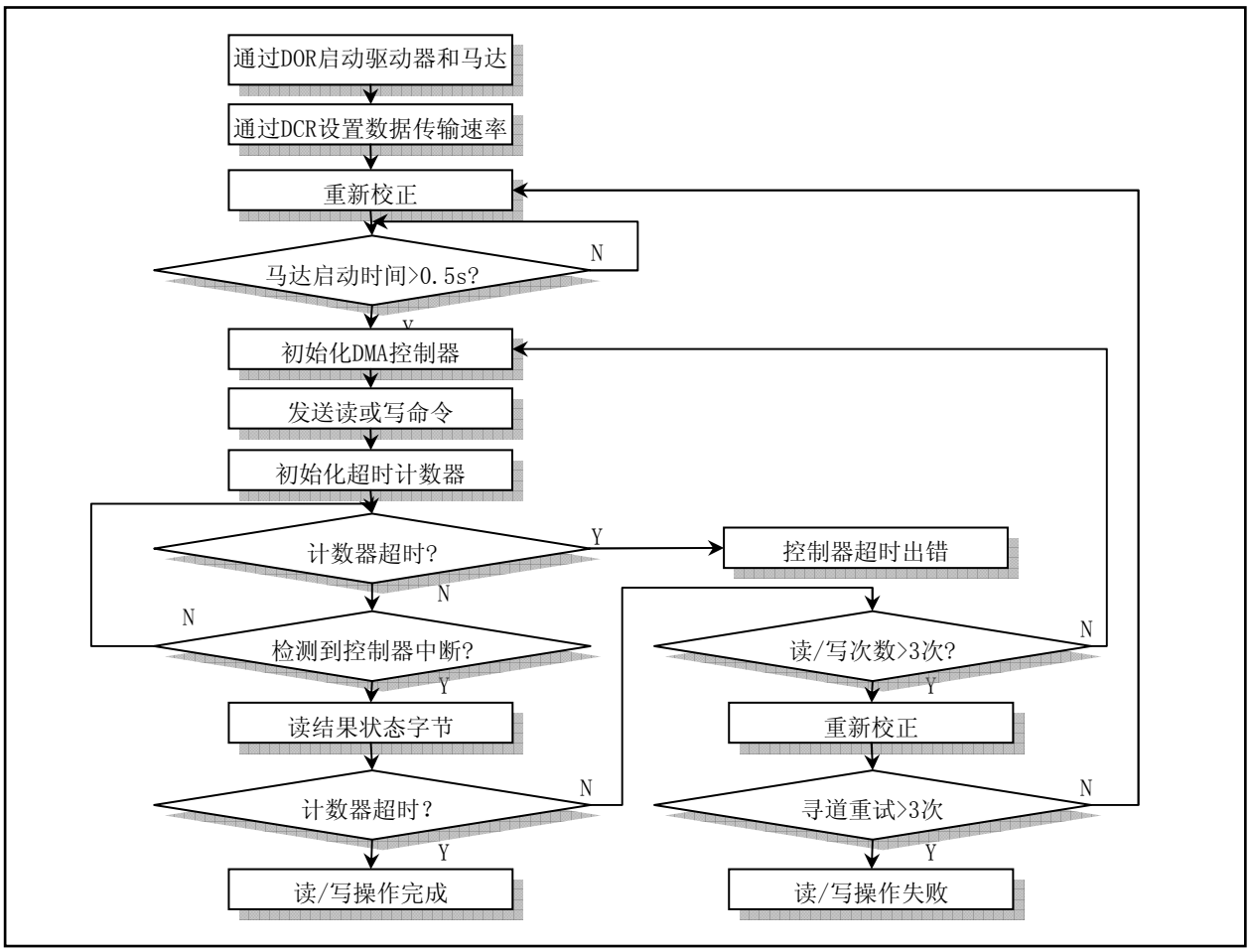

图6.3 数据读/写操作流程图

在对磁盘进行数据传输之前，磁盘驱动器的马达必须首先达到正常的运转速度。对于大多数3½英寸软驱来讲，这段启动时间大约需要300ms，而5¼英寸的软驱则需要大约500ms。在floppy.c程序中将这个启动延迟时间设置成了500ms。

在马达启动后，就需要使用数字控制寄存器DCR设置与当前磁盘介质匹配的数据传输率。

如果隐式寻道方式没有开启，接下来就需要发送寻道命令FD_SEEK，把磁头定位到正确的磁道上。在寻道操作结束后，磁头还需要花费一段到位（加载）时间。对于大多数驱动器，这段延迟时间起码需要15ms。当使用了隐式寻道方式，那么就可以使用“指定驱动器参数”命令指定的磁头加载时间（HLT）来确定最小磁头到位时间。例如在数据传输速率为500Kbps的情况下，若HLT=8，则有效磁头到位时间是16ms。当然，如果磁头已经在正确的磁道上到位了，也就无须确保这个到位时间了。

然后对DMA控制器进行初始化操作，读写命令也随即执行。通常，在数据传输完成后，DMA控制器会发出终止计数（TC）信号，此时软盘控制器就会完成当前数据传输并发出中断请求信号，表明操作已到达结果阶段。如果在操作过程中出现错误或者最后一个扇区号等于磁道最后一个扇区（EOT），那么软盘控制器也会马上进入结果阶段。

根据上面流程图，如果在读取结果状态字节后发现错误，则会通过重新初始化DMA控制器，再尝试重新开始执行数据读或写操作命令。持续的错误通常表明寻道操作并没有让磁头到达指定的磁道，此时应该多次重复对磁头执行重新校准，并再次执行寻道操作。若此后还是出错，则最终控制器就会向驱动程序报告读写操作失败。

磁盘格式化操作

Linux 0.11内核中虽然没有实现对软盘的格式化操作，但作为参考，这里还是对磁盘格式化操作进行简单说明。磁盘格式化操作过程包括把磁头定位到每个磁道上，并创建一个用于组成数据字段（场[1]）的固定格式字段。

在马达已启动并且设置了正确的数据传输率之后，磁头会返回零磁道。此时磁盘需要在500ms延迟时间内到达正常和稳定的运转速度。

在格式化操作期间磁盘上建立的标识字段（ID字段）是在执行阶段由DMA控制器提供。DMA控制器被初始化成为每个扇区标识场提供磁道（C）、磁头（H）、扇区号（R）和扇区字节数的值。例如，对于每个磁道具有9个扇区的磁盘，每个扇区大小是2（512字节），若是用磁头1格式化磁道7，那么DMA控制器应该被编程为传输36个字节的数据（9扇区 x 每扇区4个字节），数据字段应该是：7,1,1,2，7,1,2,2，7,1,3,2，...，7,1,9.2。因为在格式化命令执行期间，软盘控制器提供的数据会被直接作为标识字段记录在磁盘上，数据的内容可以是任意的。因此有些人就利用这个功能来防止保护磁盘复制。

在一个磁道上的每个磁头都已经执行了格式化操作以后，就需要执行寻道操作让磁头前移到下一磁道上，并重复执行格式化操作。因为“格式化磁道”命令不含有隐式的寻道操作，所以必须使用寻道命令SEEK。同样，前面所讨论的磁头到位时间也需要在每次寻道后设置。

[1] 关于磁盘格式的说明资料，以前均把filed翻译成场。其实对于程序员来讲，翻译成字段或域或许更顺耳一些。