Gmail **Mark Lemmert <mark.lemmert@gmail.com>**

driver file read question

Peter Ferrie <peter.ferrie@gmail.com> Mon, Aug 8, 2016 at 5:37 PM
To: Mark Lemmert <mark.lemmert@gmail.com>

Hi Mark,

>> It required a fairly substantial change internally, owing to the way
>> that the code grew over time. Initially, it was based on a
>> sector-oriented DOS 3.3 driver. Then it became a sector-oriented
>> ProDOS driver, and has finally become a block-oriented ProDOS driver.
>
> Ah, ok. Sounds like efficiency probably increased with each step as the
> design became closer to the format of the raw data?

It's become simpler to read in blocks, but more complicated to allow reads to resume, because of the need to keep a pointer into a cache of the last read block.
Still, it will be faster than ProDOS.

>> Since I'm making significant changes right now, is there any chance
>> that you'd want a seek routine? Or will you always read initially
>> from the start of the file and discard what isn't interesting?
>
> I have some ideas on how that might be useful. Thanks! A few questions.
>
> With a seek routine would that mean I could specify a range of bytes in the
> file I want to read in? Like read byte $100-$180, skipping the first page
> of data? Or would it be more like DOS RWTS where I'd specific the block
> range to read and the driver returns whatever is in those blocks?

The seek would be a separate command and a separate call from the read. You specify the number of bytes to skip in one call, and the number of bytes to read from the new position using another call.

> Is the advantage of seek routine speed and/or simplicity in the higher
> level code? For example, if the driver does a read/seek to return bytes
> $5000-$10000 do you think the driver run time would be significantly faster
> than if the driver just read bytes $00 - $10000 and let the higher level
> code discard bytes $00-$4FFF?

Yes, the seek reads only one block (the final one where the file pointer ends up, so that the next read comes from the cache initially), instead of reading and decoding everything in between. That's a huge saving if you're seeking a long way.

> Would an upgrade the driver be available first with just the read length
> added, so I can start working with it, or does the seek need to be done at
> the same time?

I've added the whole thing. It's ready to use right now. You have to specify the size value for all of the commands now, not just the write.

Some notes about it:
;- writes can be performed beginning at any block in a file, by seeking to the block first
;- file size can be retrieved by seeking to position 0 within a file and then reading bleft
;- to read a complete file, set the size to something large, eg $ffff

```
;constants
cmdseek   = 0
cmdread   = 1
cmdwrite  = 2

;zpage usage
tmpsec    = $3c
reqsec    = $3d
A1L       = $3c        ;only during init
A1H       = $3d        ;only during init
A2L       = $3e        ;only during init
A2H       = $3f        ;only during init
A3L       = $40        ;only during init
A3H       = $41        ;only during init
curtrk    = $40

command   = $42        ;ProDOS constant
unit      = $43        ;ProDOS constant
adrlo     = $44        ;ProDOS constant
adrhi     = $45        ;ProDOS constant
bloklo    = $46        ;ProDOS constant
blokhi    = $47        ;ProDOS constant

bleftlo   = $ef        ;(internal) bytes left in file
blefthi   = $f0        ;(internal) bytes left in file
blkofflo  = $f1        ;(internal) offset within cache block
blkoffhi  = $f2        ;(internal) offset within cache block
status    = $f3        ;returns non-zero on error
auxreq    = $f4        ;set to 1 to read/write aux memory
sizelo    = $f5        ;size of request
sizehi    = $f6        ;size of request
entries   = $f7        ;(internal) total number of entries
reqcmd    = $f8        ;0=seek, 1=read, 2=write
ldrlo     = $f9        ;set to load address
ldrhi     = $fa        ;set to load address
namlo     = $fb        ;set to name of file to open
namhi     = $fc        ;set to name of file to open
step      = $fd        ;(internal) state for stepper motor
tmptrk    = $fe        ;(internal) temporary copy of
current track
phase     = $ff        ;(internal) current phase for seek
reloc     = $d000
dirbuf    = reloc+$500  ;$200 bytes
encbuf    = dirbuf+$200 ;$200 bytes
```

Memory usage is now $D000-D8FF.
The openfile entrypoint is $D003. It supports seek/read/write as well.
The general seek/read/write entrypoint is $D000 (open the file first).
There is no "close" operation. You can have only one file open at a time, so opening another file will close the current one.

```
;open file and read 3 bytes to $1234
        ldx #cmdread
        stx reqcmd
        dex
        stx auxreq
        lda #3
        sta sizelo
        lda #0
        sta sizehi
        lda #$34
        sta ldrlo
        lda #$12
        sta ldrhi
        lda #<noxmain
        sta namlo
        lda #>noxmain
        sta namhi
        jsr $d003

;seek to position $192 in opened file

        lda #cmdseek
        sta reqcmd
        lda #$92
        sta sizelo
        lda #1
        sta sizehi
        jsr $d000

;read $500 bytes to $6000 from opened file
        ldx #cmdread
        stx reqcmd
        lda #0
        sta sizelo
        lda #5
        sta sizehi
        lda #0
        sta ldrlo
        lda #$60
        sta ldrhi
        jsr $d000
```

Entries no longer returns the number of pages read, but you can track it yourself since you know exactly how many bytes you are reading, and you have access to the file size.

Another thing - file entries in the file system aren't "free". There are 13 files per block. The 14th file will require one more block on the disk. For each file that is larger than a block, there will be a block that holds the block list for that file. If you have, say, three files that are 513 bytes long, that's six blocks on the disk (three block lists, three data blocks).
With the new driver, you can pack many files into a single one if you want to. It could save a lot of space. We can talk more about that if it's interesting to you.

NOXARCH.zip
2K