FileCore - Phase 1
Functional Specification

Confidential Information

Document Ref:	None
Project:	Ursula
Revision:	0.05
Date:	02-07-1997
Author(s):	Simon Proven
Change:	None

Contents:

1.0 Overview

RISC OS 3.60 introduced enhancements to FileCore to increase the maximum disc size from 512M bytes; this work was largely successful, and discs of more than 8G bytes in size have successfully been used with the new addressing mechanism. This is documented in [1].

However, when large discs are used, the size of the minimum object increases proportionally, making the space allocation of small files on the disc very inefficient. This document specifies how FileCore will be enhanced so that this problem is reduced.

Alone, the changes specified here will allow roughly a halving of the minimum object size, for any given size of disc. These changes, in conjunction with a new directory format, specified elsewhere, will allow further reductions in the minimum size of a disc object.

This document assumes that the reader understands the information contained within [1] and [2].

2.0 Outstanding Issues

None.

3.0 Technical Background

FileCore's free space map governs the allocation of space on a FileCore disc. The operation of the free space map is described in the RISC OS 3 programmer's reference manual [2].

The current limitations in the free space map result in the doubling of the minimum object size every time that the disc size doubles. This limitation is caused by the number of bits that can be used to represent a disc object's id; each object on the disc must have a unique id. idlen gives the number of bits allocated to an object id. Currently, idlen cannot be more than 15. This limits us to 32765 objects on a disc (2^{idlen} possible object ids is 32768, but 3 object ids are reserved for special cases).

So we change FileCore to allow idlen to be larger. This requires that we alter our handling of the free space chain within a map zone; the free link at the start of the chain cannot be more than 15 bits. To handle this limitation, we change our handling of free links so that when idlen is more than 15 bits, free links are treated as 15 bits (previously, free links were treated as always being idlen bits long).

Note that the maximum required length for a free link is 13 bits. A zone is a single sector within the map. FileCore supports sector sizes of upto 1024 bytes, giving a maximum zone size of 8192 bits. We can thus represent the offset between two free spaces within a zone using no more than 13 bits; the maximum offset would be 8192-(idlen+1), since free spaces have to be at least idlen+1 bits in length.

3.1 Effects of Increasing idlen

Increasing idlen allows a reduction in the number of bytes represented by each map bit, which will increase the size of free space map required to cover a given disc. This table gives the relationship between disc size, idlen, and the minimum possible object sizes (assuming 512 byte sectors, and not considering an lfau of less than 512 bytes):

disc size	512M	1024M	2048M	4096M	8192M
idlen	minimum object size				
15	32K	64K	128K	256K	512K
16	8.5K	17K	34K	68K	136K
17	9K	9K	18K	36K	72K
18	9.5K	9.5K	9.5K	19K	38K
19	10K	10K	10K	10K	20K
20	10.5K	10.5K	10.5K	10.5K	10.5K

Note that the difference in allocation unit between idlen=15 and idlen=16 is around a factor of 4, unlike the others, which are a factor of 2. This is a side effect; the point at which the lfau changes as we increase the disc size moves when we increase idlen.

There are a number of side-effects which occur when we increase the allowed value of idlen; these will be addressed by other changes to the operation of FileCore and to the structure of the disc.

Firstly, the maximum number of zones on a disk will increase. Currently, the maximum number of zones is 127; this is because with 128 or more zones, it becomes impossible to correctly represent the disc address of the root directory, so we have to change the mechanism by which the root directory address is calculated and interpreted.

Also, the value of idlen affects the structure of a directory entry. Currently, directory entries contain a 24 bit field for the indirect disc address of the file. The bottom 8 bits of the address are the shared offset; the top 16 bits are the object id. Thus, if we want to allow idlen to be more than 16 bits, we must change the format of a directory entry. A new directory format should allow for longer object names (upto 255 characters) and variable size directories.

3.2 Free Chain Handling

The free chain entry gives the offset from the current free chain entry to the next entry in bits. This is the offset from the start of the current entry to the start of the next entry. The end of the free chain is marked by a zero offset value. Each zone has a separate free chain.

When idlen is more than 15, the number of bits in a free chain entry is treated as a 15 bit offset value. Although free chain ids are always treated as at most 15 bit values, we never allow free chain entries to become shorter than idlen+1 bits (ie they have the same constraints as for file frgaments). There are two reasons for this:

- Fixing broken free space maps is simpler if we can assume that all gaps and fragments are at least 15 bits.
- FileCore's space allocation strategy may not be able to cope with the concept of gaps of less than idlen+1.

Given these constraints, for at least the initial implementation, the size of free fragments will be kept to a minimum of idlen+1. There are potential advantages to allowing free fragments to become shorter, but the risk involved is too high, given the time available.

3.3 Free Space Map Sizes

With increased idlen, the maximum size of the free space map will grow. The approximate maximum size of the allocation bytes in the map, in bytes, is:

$$\frac{2^{idlen} \times (idlen + 1)}{8}$$

Note that this excludes the spare bits in each zone. If we allow for these, and also for the extra bits in zone 0 which hold the disc record, we can calculate the maximum free space map size, and the corresponding maximum disc sizes and minimum object sizes:

estimated maximum disc size and minimum object size for a given lfau

(note: only 1K-8K lfau shown - larger or smaller is possible)

		lfau=1K		**lfau=2K**		**lfau=4K**		**lfau=8K**	
idlen	**max size of map**	**max disc**	**min object**	**max disc**	**min object**	**max disc**	**min object**	**max disc**	**min object**
	current maximum:								
15	64.5K	511 M	16K	1,023 M	32K	2,046 M	64K	4,092 M	128K
	with big maps, old directory format:								
16	137K	1,086 M	17K	2,173 M	34K	4,347 M	68K	8,695 M	136K
	with new directory format:								
17	291K	2,309 M	18K	4,618 M	36K	9,237 M	72K	18,474 M	144K
18	615K	4,881 M	19K	9,762 M	38K	19,524 M	76K	39,048 M	152K
19	1,291K	10,246 M	20K	20,493 M	40K	40,987 M	80K	81,974 M	160K

As can be seen from the table, we can trade off the free space map size against the lfau, to achieve the best possible optimisation of performance. With a larger free space map, RAM usage will increase, and some operations on the free space map will become slower. In general operations within a single zone will not become slower, but those operations which operate on the entire map will tend to be slower with a larger map.

3.4 Root Directory Disc Address

The representation of the root directory disc address is altered to allow free space maps larger than 127 zones. At present, the root directory disc address is held in the format &2xx. This is a FileCore internal disc address, ie it consists of an object id (2) and a sharing offset, which is a value from 1-255, representing offsets 0-254, in sharing units. Sharing units are normally sectors, or can be power-of-2 multiples of a sector (specified by the ShareSize field in the disc record, introduced in RISC OS 3.60; ShareSize is the log to the base 2 of the unit of sharing).

The root directory is stored on the disc in the same object as the free space map, directly after the free space map itself. If ShareSize is 0, then we cannot represent the sharing offset of the root directory, given that the free space map may now be much larger than previously. Increasing ShareSize only to allow correct representation of the root directory disc address is not a good idea (it makes all shared objects have a larger granularity, reducing overall space efficiency).

We instead make the root directory object a separate disc object. The root directory can be in any zone in the free space map, and can be in an object of any id, except the two reserved object ids. This change is more future proof than the previous suggested change, which was to make the disc id of the root direcory &201, and make the map/root directory address calculation a special case. This has the disadvantage that when we move to extendable directories, there will be no guarantee that we can safely extend the root directory object as required.

The disadvantage with this method is that disc fixing operations may become more involved, since in theory the root directory could be anywhere on the disc. In practice, the root directory will always exist in the same map zone as the free space map, as the object cannot be moved between zones by a compaction operation. However, a compaction operation may try to move the root directory object to another place within the same zone, in order to optimise the space utilisation within that zone.

All disc formatting software, to give consistency of formatting of discs, will place the root directory as the first object after the free space map object in the map zone. The object id allocated will be the first object id for that zone. The object id allocated to the root directory should not be changed by any operation which FileCore may perform on the disc, even if the root directory does move within the zone.

3.5 Memory Allocation

Currently, FileCore holds the directory cache, the free space maps, and the random access file cache for any FileCore instance in the RMA. This leads to a maximum RMA allocation for each FileCore instance of 8*64K (512K) for the free space maps, plus a maximum of around 512K for the directory cache and random access file cache together. If we have idlen=18, however, then we end up with around 8*516K potential for RMA used for the free space maps. Further, there may be multiple instances of FileCore, further increasing such problems.

This could be very problematic, as the RMA can only grow to around 11M bytes, and RMA fragmentation is also an issue. In order to resolve this, FileCore will hold each free space map in a dynamic area of its own. The dynamic area will be protected from USR mode write access to avoid accidental damage by erronous code. Each dynamic area will be named according to the filing system - e.g. ADFS free space maps would be in dynamic areas "ADFS map".

3.6 Backwards Compatability

Backwards compatability will be maintained with applications which use FileCore based filing systems, and those which legally use documented FileCore SWIs. Further, FileCore will maintain backwards compatability with old FileCore E format hard discs.

Full compatability will be maintained with existing E and F format floppies, and image filing systems such as DOSFS will continue to operate as present. Compatability with D format floppies and hard discs will be retained, but the use of such devices is deprecated. Few users require access to such devices, and their usefulness is limited. Testing will concentrate on E and F format floppies, and E format hard discs of both old and new formats. S, M and L format floppies are no longer supported.

Where software attempts to directly access the disc (ie where software modifies the low-level structures on the disc, such as the free space map), compatability is unlikely to be maintained. It should be noted that there is a risk of software which operates in such a manner failing in a way which causes potentially catastrophic data loss on the disc being accessed. It is therefore important that users are made fully aware of this.

Older versions of FileCore will refuse to mount new-format discs. Two of the tests which FileCore performs to check that discs are valid will fail, producing the error "Disc not understood - has it been formatted?":

- The value of idlen is tested. It must be in the range 1 to 15 for old versions of FileCore to accept the disc.
- The root directory disc access on old discs must be in the form &2xx, where xx is the offset of the root directory from the start of the free space map. This test will also fail. (For discs with idlen more than 15, the root directory disc address will be &00xxxx01).

Software such an old version of fsck may erronously attempt to fix a disc which has increased values of idlen. This could result in data loss. If, however, a new directory format is adopted, fsck will probably fail safely, since it will think that the root direcory is broken, and stop there.

In the case of software which does attempt to modify the disc format:

- Software which modifies old format free space maps should have no problems as long as it doesn't attempt to modify a new format free space map.
- Software which modifies the free space map of a new format disc is likely to have serious problems. In the worst case, the software could damage the free space sufficiently that it cannot be recovered by any means other than restoration from a backup. Software that falls into this category includes fsck, Oregan Disc Doctor, and other disc fixing tools. Note that such software may appear to work on partially filled discs, and only fail once the disc is becoming full.
- Software which reads the free space map to determine information about the layout of information on the disc, but does not attempt to write the map, will probably fail, but in a harmless, manner.

4.0 User Interface

4.1 The *map Command

*map currently displays the following information line before displaying the map information:

```
(   start,  length) new map, new directories
```

When idlen is more than 15, to distinguish the new format discs, the line will change to:

```
(   start,  length) big map, new directories
```

i.e the word 'new' is replaced with 'big'.

5.0 Programming Interface

The FileCore SWI interfaces will not be changed. Note however, that the disc record structure is altered, and this structure is passed to/from some FileCore SWIs (FileCore_DiscOp, FileCore_MiscOp, FileCore_SectorOp, FileCore_FloppyStructure and FileCore_DescribeDisc).

6.0 Data Interchange

The FileCore data interchange flows will not be changed.

7.0 Data Formats

7.1 Disc Record

The Disc Record structure is currently defined in the RISC OS PRM (page 2-202) with extensions added in RISC OS 3.60, defined in PRM volume 5a.

The disc record format will be extended to represent more than 255 zones. This will be represented by having a second byte for the number of zones, specifying the top 8 bits of the number of zones. This value will be stored at offset 42 in the disc record. This is currently a reserved byte, which should always be zero, for existing discs. The complete new disc record format is as follows:

offset	name	meaning
0	*log2secsize*	$\log_2$ (sector size of disc in bytes)
1	*secspertrack*	Number of sectors per track
2	*heads*	Number of disc heads if sides interleaved
3	*density*	0 hard disc 1 single density 2 double density 3 double+ density 4 quad density 8 octal density
4	*idlen*	Length of id field of a map fragment, in bits
5	*log2bpmb*	Log_2 (number of bytes per map bit)
6	*skew*	Track to track sector skew for random access file allocation
7	*bootoption*	Boot option (as *Opt 4,n)
8	*lowsector*	bits 0-5: lowest numbered sector id on a track bit 6: if set, treat sides as sequenced (rather than interleaved) bit 7: if set, disc is 40 track
9	*nzones*	least sig. byte of number of zones in the map
10	*zone_spare*	Numer of non-allocation bits between zones
12	*root*	Disc address of root directory
16	*disc_size*	least sig. 32 bits of disc size, in bytes
20	*disc_id*	disc cycle id
22	*disc_name*	Disc name
32	*disctype*	File type given to disc
36	*disc_size_2*	most sig. 32 bits of disc size, in bytes
40	*share_size*	$\log_2$ (sharing granularity in sectors)
41	*big_flag*	bit 0: set-> RISC OS partition >512MB bits 1-7: reserved, must be 0
42	*nzones_2*	most sig. byte of number of zones in the map
43-59		Reserved; must be 0

8.0 Dependencies

8.1 External

No external dependencies.

8.2 Other

New versions of the disc tools shipped on the hard disc (!HForm) and the tools use in production and testing must be updated to handle the larger free space maps.

14.0 Glossary

bpmp	Bytes per map bit. Also known as lfau.
fragment	A single contiguous allocated block on the disc.
fragment id	The identification number of a given fragment in the free space map. When a file consists of a number of fragments, all the fragments are given the same id.
idlen	The length, in bits, of a free space map **fragment id**.
lfau	Large File Allocation Unit. The lfau is the unit of space allocation on a disc. Each bit in the free space map corresponds directly to one lfau's worth of disc. The lfau is always a power of 2.
smallest fragment	The size, in map bits, of the smallest fragment, is idlen+1. This means that the smallest fragment that can be created, is lfau*(idlen+1).
object	A collection of fragments, which, when concatenated, contain the data for that object.
shared object	An object which often has a directory at the start of the object; this followed by a number of files which are children of that directory. Alternatively, a shared object can have a number of files; all the files share a common parent directory.
zone	A section of the free space map. Each zone is one sector in length. Within each zone, a free space chain lists all the unused space in that zone.

15.0 References

1. *RISC OS 3 Programmer's Reference Manual, Vol. 5a*, pp 5a-165 to 5a-168, Acorn Computers Ltd, 1995

2. *RISC OS 3 Programmer's Reference Manual, Vol. 2*, pp 2-199 to 2-209, Acorn Computers Ltd, 1992

16.0 History

Revision	Who	Date	Comment
0.00	SBP	30-04-1997	Started.
0.01	SBP	13-05-1997	Added detail.
0.02	SBP	24-06-1997	Tidied up ready for review.

0.03	SBP	25-06-1997	Alan saw it. Added references.
0.04	SBP	26-06-1997	Clarified acceptance criteria. Added section 3.6 on backwards compatability. Minor other changes.
0.05	SBP	02-07-1997	Updated from Review comments.