# TANDOS 65 Disc Anatomy

## *Disc Drive Numbering*

There may be up to four disc drives on the Microtan system. Any or all of these may be double-sided. So far as the hardware is concerned, a two sided disc is the same as two single sided discs. To make the system consistent, a numbering scheme for discs has been adopted which gives a unique number to each disc surface. Thus, the first side of the first disc is 0, the second side of the first disc is 1 etc. up to the second side of the fourth disc which will be 7. If only single-sided disc drives are available then only the odd numbers are used (a typical two single-sided drive system would therefore contain units 0 and 2 only).

## *Disc Capacity*

Depending on the floppy disc drive capability, the Microtan can write either 40 or 80 Tracks of data on each side of a 5¼" Floppy Disc. The floppy disc specification will need to be 48 TPI (Tracks per inch) and 96 TPI respectively. The first (outer) track is labelled Track 0.

TANDOS is configured to operate in Single Density mode. The early Microtan TANDOS system created 9 sectors on each track. Latest versions create 10 sectors per track. However, TANDOS could be re-configured to operate at Double Density which would result in 18 sectors per track. In either case the first sector is labelled Sector 1.

Sector length is defined by the number of data bytes that each sector holds. All TANDOS versions write 256 bytes of data to each sector.

A single sided 40 track, 10 sector floppy disc can therefore hold 100Kbytes of data.

## Disc Formatting

The TANDOS FMAT10 utility lays down a pattern of digital information on the surface of the floppy disc to identify the start and end of each sector, track and sector identification information and allocate the space for the 256 bytes of user data.

The TANDOS FDC1793/MM8877 Floppy Disc Controller chips use the IBM3740 format and Frequency Modulation (FM) for single density storage.

The table below shows the sequence of data to be generated by the formatting utility for the Floppy Disc Controller to lay down on each track of the disc. The start of the sequence is aligned with when the Index hole in the disc is detected.

Once formatted the disc can be used as a FORTH data disc where 1024 Byte data Blocks are stored in designated sectors on the disc. Directories are not necessary as each specified block is always written to the same (4) sectors on the disc. For most other applications a file storage structure is required as described below. These structures are created using the TANDOS INIT utility. The CREATE utility combines both formatting and initialisation processes.

| | **Data Byte (Hex)** | **No. of Bytes** | **Comments** |
|---|---|---|---|
| **Pre-amble** | FF | 40 | Gap 0 (Post Index) |
| | 00 | 6 | |
| | FC | 1 | Index AM |
| **One Sector [Note 1]** | FF | 26 | Gap 1 |
| | 00 | 6 | |
| | FE | 1 | ID AM |
| | XX | 1 | Track Number |
| | 0X | 1 | Side Number |
| | 01 | 1 | Sector Number |
| | F7 | 1 | Sector Length (256 Bytes) |
| | FF | 11 | Causes 2-byte CRC to be written |
| | 00 | 6 | Gap 2 (ID Gap) |
| | FB | 1 | Data AM |
| | XX | 256 | Data Field |
| | F7 | 1 | Causes 2-byte CRC to be written |
| | FF | 27 | Gap 3 (Data Gap) |
| ***** | FF | 247 | Gap 4 (Pre-Index) [Note 2] |

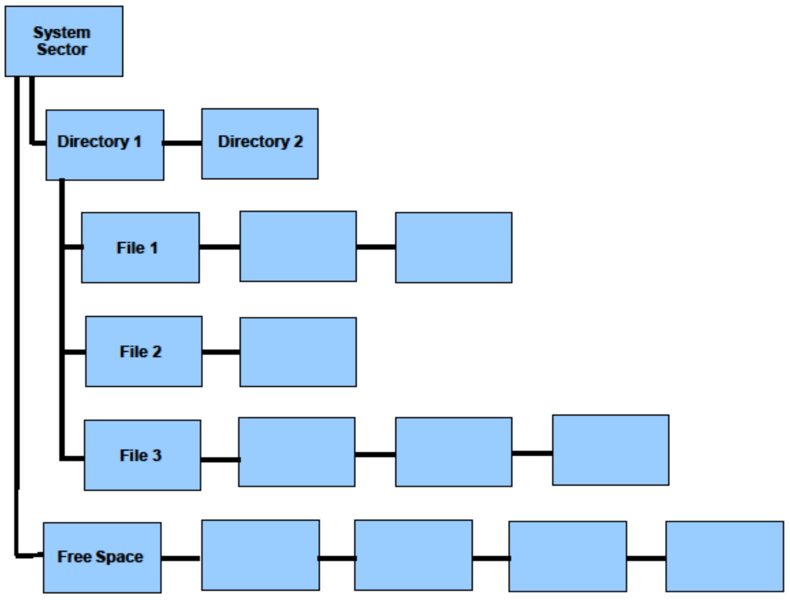
* Post-amble

Note 1. This pattern is repeated 10 times per track

Note 2. Hex FF are continued to be written until the FDC completes the sequence and generates an interrupt indicating the end of the track has been reached.

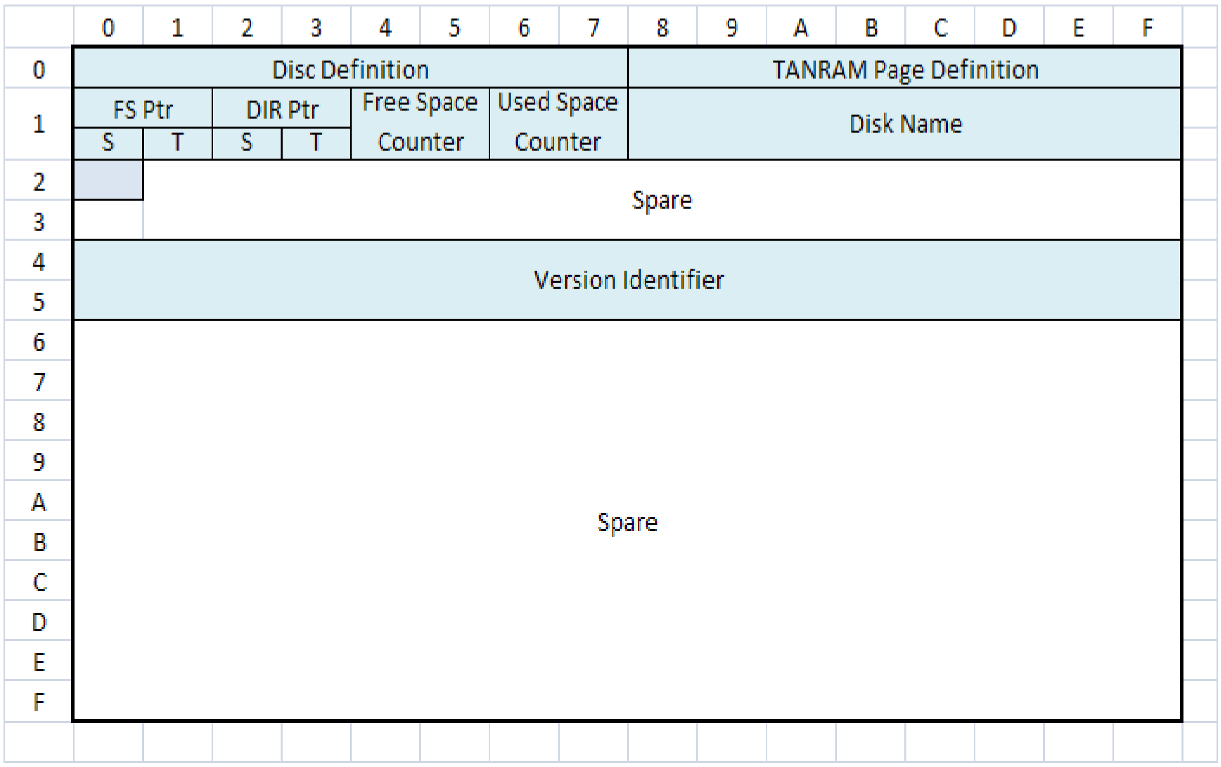
## *Disc Organisation and File Structures*

This diagram shows the various data structures created by the initialisation process and how they interact to complete the disc file structure. Each block in the diagram is one sector.  The System Sector points to both the Free Space chain and the Directory. The Free Space maybe as long or short as in appropriate (to suit the disc size and the number of sectors already used). The Directory points to each file (both start and end in fact, - this makes deleting a file a more straightforward and reliable task). Each file may be as long or short as necessary and available space permits.



## The System Sector

The very first sector on each disc (Track 0 Sector 1) is reserved for special system information. This sector is referred to as the System Sector (SS).
The information held in the System Sector is in two parts: System related and disc related.
The system related information consists of details of discs connected to the system and the amount of memory in each page.
The disc information includes such things as: where the directory is stored, how much free space is available, where it is and the name of the disc. The disc name is a string of up to 9 alphanumeric characters, provided by the user during INITialisation. It is useful in identifying a particular disc.



Bytes $0-$7 : Disc Definition

These 8 bytes define the number of tracks available on each of the 8 disc drives permissible. Byte ø refers to disc ø etc. A value of H28 indicates a 40 track disc drive. 'DSCDEF' is loaded into RAM from the system disc (drive ø) and is used to determine which devices are available. (ø tracks indicate no drive).

Bytes $8-$F : TANRAM Page Definition

These 8 bytes define the maximum amount of memory available on each TANRAM page. Byte $8 refers to TANRAM Page 0 etc. The value contained in the byte is the high byte of the maximum address on the page i.e. if the maximum address is $A1FF then the relevant byte will contain $A1. The least significant byte is always assumed to be $FF.

Bytes $10-$11 : Free Space Pointer

Byte $10 contains the sector number and Byte $11 the track number of the start of the free space chain. On a freshly INITialised disc, this will be Track 0 Sector 7. This pointer will be updated whenever sectors are allocated to files or directory use, and when files are deleted.

Bytes $12-$13 : Directory Pointer

Byte $12 contains the sector number and Byte $13 the track number of the first sector of the disc directory. This will usually be Track 0, Sector 4 but may point to anywhere on the disc (particularly after a long period of heavy use creating and deleting files). A sector number of 0 (which is an illegal number, as far as the disc hardware is concerned) indicates that there is no directory on the disc.

Bytes $14-$15 : Free Space Count

These bytes together (byte $14 least significant) contain the total number of sectors remaining unallocated. This number will be the total number of sectors on the disc minus two on a freshly INITialised disc and is updated whenever sectors are allocated to files or directory.

Bytes $16-$17 : Used Space Count

These bytes together (Byte $16 least significant) contain the total number of sectors allocated to files on the disc. The number will be on a freshly INITialised disc and is updated whenever sectors are allocated to file usage. The count is not changed when sectors are allocated to directory use.

Bytes $18-$20 : Disc Name

This is a nine character ASCII string which is used to store an arbitrary disc name. This name is specified by the user during INITialisation. It is used by DIRectory and COPY.

Bytes $21-$3F : Spare.

Bytes $40-$5F : Version identifier

This area is reserved for an ASCII string identifying the particular version of TANDOS 65 used to create the disc. It is written by INITialise.

Bytes $60-$FF Spare.

## Directories.

Each disc is capable of holding many files (about 350 very short files for single-sided, single density, 40 track discs). The operating system needs to know a certain amount of information about each of these files. In particular: what it is called (filename), where it is kept on the disc and how big it is. All this information is held in the directory. In fact the directory is nothing more than another file on the disc which just happens to contain information about every other file.
The operating system uses the directory in much the same way as you would use the index to a book. Using the directory, the system has rapid access to file information.

The disc directory consists of a number of directory sectors (often only 1). The first directory sector is pointed to by the directory pointer in the system sector. Two bytes in each directory sector are reserved as pointers to the next directory sector. The last directory sector is indicated by a 0 in the next sector byte. By this means the directory maybe as long, or as short as required.
Each directory sector may hold up to 15 file entries.
Each directory sector is organised into three parts:-
Next directory Sector Pointer: Bytes $0 and $1 these point to the next directory sector. Byte $0 is the track and Byte $1 is the actual sector. A value of 0 in Byte $1 indicates that the current directory sector is also the last.
The remaining 7 bits are undefined but care should be taken to leave them unchanged when modifying the Write Protect bit to preserve compatibility with future systems.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Next Dir | | No. Dir | Filename | | | | | | | | | Number of Sectors | | First | |
| | T | S | | | | | | | | | | | | | S | T |
| 1 | Last | | ATT | Additional Entries | | | | | | | | | | | | |
| | S | T | | | | | | | | | | | | | | |
| 2 | | | | Additional Entries | | | | | | | | | | | | |
| 3 | | | | Additional Entries | | | | | | | | | | | | |
| 4 | | | | Additional Entries | | | | | | | | | | | | |
| 5 | | | | Additional Entries | | | | | | | | | | | | |
| 6 | | | | Additional Entries | | | | | | | | | | | | |
| 7 | | | | Additional Entries | | | | | | | | | | | | |
| 8 | | | | Additional Entries | | | | | | | | | | | | |
| 9 | | | | Additional Entries | | | | | | | | | | | | |
| A | | | | Additional Entries | | | | | | | | | | | | |
| B | | | | Additional Entries | | | | | | | | | | | | |
| C | | | | Additional Entries | | | | | | | | | | | | |
| D | | | | Additional Entries | | | | | | | | | | | | |
| E | | | | Additional Entries | | | | | | | | | | | | |
| F | | | | Empty | | | | | | | | | | | | |

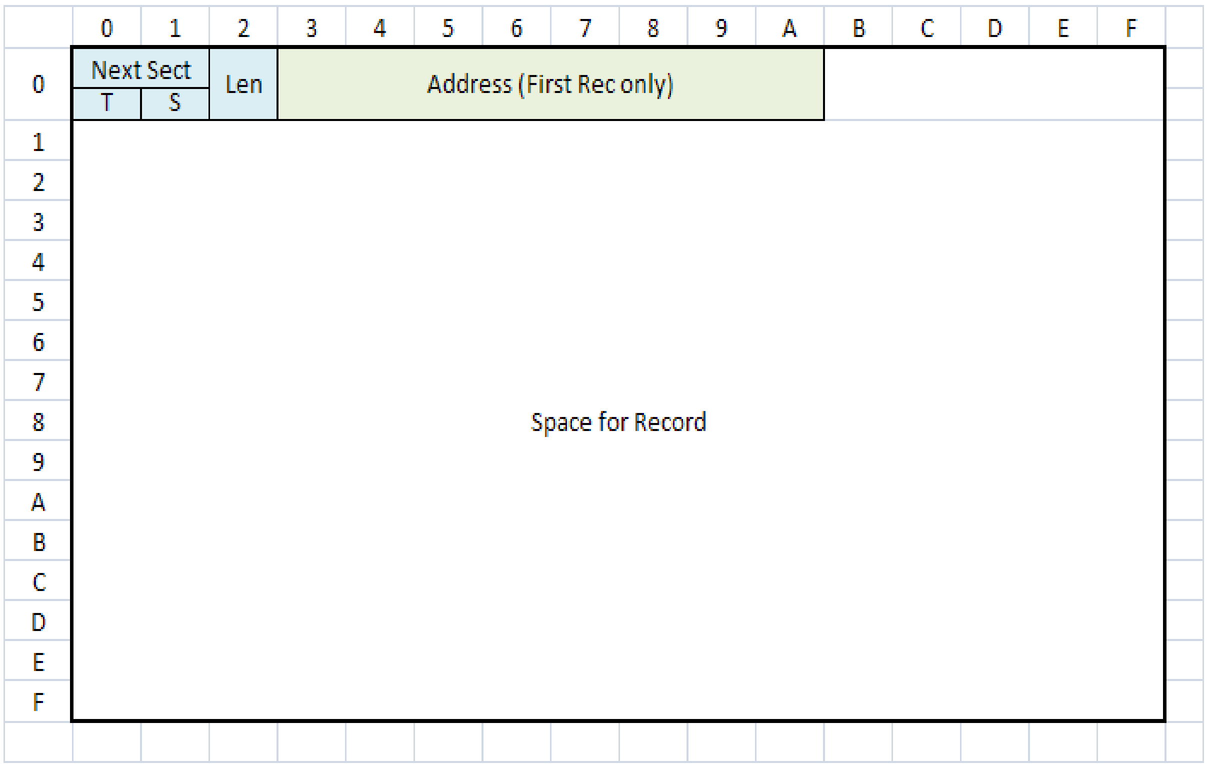
The directory sector is organised as follows:
The position of the first directory sector is given in bytes $12 (sector) and $13 (track) of the system sector. When the directory consists of more than one sector, the first two bytes of each sector indicate the track and sector numbers where the next sector of the

directory has been stored. The last sector of the directory has these bytes both set to 0. The third byte in a directory sector indicates how many file names it contains. A full directory sector holds 15 entries, but if files have been deleted, the directory entry is overwritten with zeros. The other entries are not 'closed up' to remove the gap.
The first directory entry starts in the next byte. Each entry has 16 bytes, the first 6 for the filename, and the next 3 for the extension. After these come a series of numbers. First, the file length (in sectors), given as two bytes, low byte first (A file can occupy more than 256 sectors). Then comes the sector number, followed by the track number, where the first sector of the file is saved. Next come the sector and track numbers of the last sector of the file. Finally comes a single byte which holds the file attributes. This is 80 (Hex) for a write-protected file and 0 for non-protected files.
This sequence is repeated for successive entries, and finally the last 13 bytes of the directory sector are filled with zeros.

So, in summary:
Byte $0 : TRACK of next directory sector
Byte $1 : SECTOR of next directory sector
Byte $2 : Number of entries in this sector
The rest of the sector is filled with file entries in the following format.
6 bytes Filename
3 bytes Filename extension
2 bytes File length (sectors). Low byte first.
1 byte SECTOR First Sector of file
1 byte TRACK
1 byte SECTOR Last sector of file
1 byte TRACK
1 byte ATTRIBUTE

## File Organisation

The sectors which form an individual file are linked together into a chain (in a similar way to the Free-Space). The first sector of a file is pointed to by information contained in the File Information Block in its directory entry. The first sector contains information which points to the second sector and so on. The pointers are contained in Bytes $0 and $1 of each file sector. Byte $0 is the Track and Byte $1 the Sector number of the next sector in the file. A value of $0 in Byte $0 indicates that the current sector is also the last sector of the file.


0 1 2 3 4 5 6 7 8 9 A B C D E F
0 1 2 3 4 5 6 7 8 9 A B C D E F
Next Sect
T
S
Len
Address (First Rec only)
Space for Record

Files are saved in the following format.
The first two bytes hold the track and sector numbers of succeeding sectors, or 0 if this is the last (or the only) sector of a file.
The next byte holds the length of the first record in this sector. In the first sector, the first record will normally include an address record, see DOS users manual page 5.6.
This occupies 8 bytes, making a total of 11 allocated. So there are 245 bytes left in the sector or $F5 (in hex) bytes left for saving data.
Succeeding sectors will have track and sector pointers to the following sector, then a record length byte, leaving 253 bytes for saving data. So the third byte of the other sectors in a file is normally $FD (in hex), and the rest of the sector is filled with data.
The last sector in a file is not normally occupied completely by data, so the record length byte is less than $FD. The unused part of the sector is filled out with zeros in normal files, i.e. files saved with a DSAVE command.

Load Modules
Load modules are files, usually created by the DSAVE command, which are intended to be loaded into memory.
The first record in the file is always an address record. Address records in load modules

are non-standard in that the record length byte is $FF but the record is only 7 bytes long. Further address records may occur anywhere in a load module, (hence the requirement for some special indicator).

Address Records

| | |
|---|---|
| Byte 0 Record length: | always $FF |
| Byte 1 TANRAM Page: | TANRAM Page number to which the load module should be loaded, |
| Bytes 2-3 Start Address: | the address of the first byte of data in the load module. Byte 2 is least significant. |
| Bytes 4-5 End Address: | the address of the last byte of data in the load module. Byte 4 is least significant. |
| Bytes 6-7 Transfer Address: | the address to which control should be transferred on completion of loading. Byte 6 is least significant. If both bytes 6 and 7 are zero then control will not be transferred to the load module but will continue with TANBUG on completion of loading |

The data from subsequent records is loaded at the start address upwards until either the data is exhausted or a new address record is encountered.
DLOAD will not load files correctly with records which cross sector boundaries.
DSAVE generates load modules with no more than one data record per sector.

Data files saved under Basic have a different structure. In particular, they do not have address records, since data files have no fixed address. Rather, they appear to be stored as a series of records on the disc, each record with just a length byte, leaving it to DBASIC to decide the addressing. For this reason, basic data files cannot be read by normal DOS commands.

## Free Space

Unused sectors on the disc are linked together to form a 'free-space chain'. Thus, the first free sector is pointed to by the 'free space pointer' in the system sector. The first sector points to the second, the second to the third, and so on. This organisation is created initially by the INITialise command. During this process, all unused sectors are filled with zeroes (except the two bytes used as pointers).
INITialise creates the free space chain in such a way that the pointers never point to an adjacent sector but to a sector about 1/3rd of a disc revolution 'later'. This considerably reduces the time to access files as there is just sufficient time to process each sector and request the next before that sector appears under the head. If the sectors were adjacent on the disc, then the 'next' sector would have gone past too quickly and the system would have to wait for an entire disc revolution.
This optimum arrangement is gradually eroded as a disc has files created and deleted in a random way.
During normal TANDOS 65 operations, sectors are used for files and then, perhaps, returned to the free space chain. Under these circumstances the sector contents will be undefined.
The pointers in each sector of free space are contained in Bytes $0 and $1. Byte $0 is the track and Byte $1 the sector number of the next sector in free space. A value of 0 in Bytes $0 and $1 indicates that the current sector is also the last sector.

Note that the first two bytes of any sector are always the 'Linking' pointers in the free space chain. They are always in the order TRACK, SECTOR. What happens when a file finishes to these bytes is important for full understanding. Should a file use a sector as a last entry on the disc and not spill onto the next available sector, then the contents of these two bytes are transferred into bytes $11 & $10 of the system sector and they are then set to zero to signify the last module of this file. As these 2 bytes were pointing to the next free sector for dumping then the free space chain is always kept track of by the system sector. As you can see, the system sector is very important for holding the whole act together and should it get accidentally corrupted as can happen on a power cut etc. when the head happens to be over track 0, then irreparable damage seems to have been done. It is quite feasible to 'repair' this damage with DZAP.