Atari Image File Formats

DrCoolZic V1.0 – September 2014

This document provide a detailed description of several file formats used by Atari Emulators.

Table of Contents

ST FILE FORMAT

The .ST image files format has been created originally for PaCifiST emulator and is extremely simple. A .ST file contains the raw data of the FD content with no header or compression. The sectors are stored in the expected logical order. So, on a sector basis the images run from sector 1 to however many sectors are on the disk. Tracks are stored in alternating side order, e.g. a double sided disk is stored as:

- TRACK 0, SIDE 0
- TRACK 0, SIDE 1
- TRACK 1, SIDE 0
- TRACK 1, SIDE 1
- TRACK 2, SIDE 0
- TRACK 2, SIDE 1
- ...

MSA FILE FORMAT

The .MSA image file format is made up as follows:

Header:

- Word ID marker, should be $0E0F
- Word Sectors per track
- Word Sides (0 or 1; add 1 to this to get correct number of sides)
- Word Starting track (0-based)
- Word Ending track (0-based)

Individual tracks follow the header in alternating side order, e.g. a double sided disk is stored as:

- TRACK 0, SIDE 0
- TRACK 0, SIDE 1
- TRACK 1, SIDE 0
- TRACK 1, SIDE 1
- TRACK 2, SIDE 0
- TRACK 2, SIDE 1
- ...

Track blocks are made up as follows:

- Word Data length
- Bytes Data

If the data length is equal to 512 x the sectors per track value, it is an uncompressed track and you can merely copy the data to the appropriate track of the disk. However, if the data length value is less than 512 x the sectors per track value it is a compressed track.

Compressed tracks use simple a Run Length Encoding (RLE) compression method. You can directly copy any data bytes until you find an $E5 byte. This signals a compressed run, and is made up as follows:

- Byte Marker - $E5
- Byte Data byte
- Word Run length

So, if MSA found six $AA bytes in a row it would encode it as: $E5AA0006

What happens if there's an actual $E5 byte on the disk? Well, logically enough, it is encoded as: $E5E50001

This is obviously bad news if a disk consists of lots of data like $E500E500E500E500... but if MSA makes a track bigger when attempting to compress it, it just stores the uncompressed version instead.

MSA only compresses runs of at least 4 identical bytes (after all, it would be wasteful to store 4 bytes for a run of only 3 identical bytes!). There is one exception to this rule: if a run of 2 or 3 $E5 bytes is found, that is stored appropriately enough as a run. Again, it would be wasteful to store 4 bytes for every single $E5 byte.

The hacked release of MSA that enables the user to turn off compression completely simply stops MSA from trying this compression and produces MSA images that are completely uncompressed. This is okay because it is possible for MSA to produce such an image anyway, and such images are therefore 100% compatible with normal MSA versions (and MSA-to-ST of course).

DIM FORMAT

The file format of normal .DIM image files are quite the same as the .ST image files (see .ST FILE FORMAT) - the .DIM image files just have an additional header of 32 bytes. However, there are also "compressed" images which only contain the used sectors of the disk. It is necessary to parse the FAT to "uncompress" these images.

The header contains following information:

0x0000	Word	ID Header (0x4242('BB'))
0x0002	Byte	1 = disk configuration has been detected automatically 0 = the user specified the disk configuration
0x0003	Byte	Image contains all sectors (0) or only used sectors (1)
0x0004	Word	$0000
0x0006	Byte	Sides (0 or 1; add 1 to this to get correct number of sides)
0x0008	Byte	Sectors per track
0x000A	Byte	Starting Track (0 based)
0x000C	Byte	Ending Track (0 based)
0x000D	Byte	Double-Density(0) or High-Density (1)
0x000E	18 Bytes	A copy of the Bios Parameter Block (BPB) of this disk.

STT Disk Image Format

All unused flags must be 0 for future compatibility, do not fail if a flag is set that isn't supported.

- STT version: 1=First version (1st May 2001). The version numbers must remain the same for all compatible formats
- STT flags: None yet defined
- Data flags: Bit 0=Sectors and ID field data
- Bit 1=Raw track data
- Sectors flags: None yet defined
- Raw flags: None yet defined

Header

All LONGs and WORDs in little endian format.

- L Magic: 'STEM' ($53 $54 $45 $4d)
- W STT Version
- W STT Flags - flags that affect this header or all tracks
- W All Track Data Flags - specifies what data is contained for every track, individual tracks may have more but can't have less.
- W Number of Tracks - maximum 86
- W Number of Sides - maximum 2
- L Track 0 Side 0 Data Section Offset
- W Track 0 Side 0 Data Section Length
- L Track 1 Side 0 Data Section Offset
- W Track 1 Side 0 Data Section Length
- L Track 2 Side 0 Data Section Offset
- W Track 2 Side 0 Data Section Length
- ....
- L Track NumTracks-1 Side NumSides-1 Data Section Offset
- W Track NumTracks-1 Side NumSides-1 Data Section Length

Track Data - cannot exceed 65535 bytes

- L Magic: 'TRCK' ($54 $52 $43 $4b)
- W Track Data Flags

..Sectors section (bit 0 of data flags set)

W Offset to End of Section - From start of track data.

All sections must start with this so unsupported sections can be skipped

W Sectors Flags

W Number of Sectors

--

B First Sector Track Number \

B First Sector Side Number \

B First Sector Sector Number \ The address id field

B First Sector Sector Length Index / of the sector

B First Sector Address Field CRC byte 1 /

B First Sector Address Field CRC byte 2 /

W First Sector Data Offset (from start of track data)

W First Sector Data Length

-- Repeat up to and including Number of Sectors-1

(any future data should be added here)

SECTORS DATA (in one big block)

..Raw section (bit 1 of data flags set)

W Offset to End of Section - From start of track data
W Raw Flags
W Raw Data Start Offset (from start of track data)
W Raw Data Length

(any future data should be added here)

RAW DATA (in one big block)

..Undefined section (if bit 2 of data flags set)

W Offset to End of Section - From start of track data

..Undefined section (if bit 3 of data flags set)

W Offset to End of Section

..Undefined section (if bit 4 of data flags set)

W Offset to End of Section

..Undefined section (if bit 5 of data flags set)

W Offset to End of Section

..Undefined section (if bit 6 of data flags set)

W Offset to End of Section

..Undefined section (if bit 7 of data flags set)

W Offset to End of Section

..Undefined section (if bit 8 of data flags set)

W Offset to End of Section

..Undefined section (if bit 9 of data flags set)

W Offset to End of Section

..Undefined section (if bit 10 of data flags set)

W Offset to End of Section

..Undefined section (if bit 11 of data flags set)

W Offset to End of Section

..Undefined section (if bit 12 of data flags set)

W Offset to End of Section

..Undefined section (if bit 13 of data flags set)

W Offset to End of Section

..Undefined section (if bit 14 of data flags set)

W Offset to End of Section

..Undefined section (if bit 15 of data flags set)

W Offset to End of Section

STX Pasti Format

Separate document – See Pasti File documentation by DrCoolZic

IPF Format

Separate Document - See Interchangeable Preservation Format (IPF) Documentation by DrCoolZic

References

Format of .ST and .MSA files from Damien Burke 31st August 1997

Format DIM Ggn (George) & Darren Birks