

Technical paper 2

Map Maker file formats



Last amended: 11th February 2001

The various file formats used by Map Maker fall into two categories:

- ASCII text files, these are designed to be editable by users using either the text editors supplied within Map Maker or else any word processor.
- Binary files, these are not designed for users to edit directly, however the formats are supplied here so that third party developers can create their own import and export filters allowing them to link their programs to Map Maker.

1 ASCII file formats - user definable

1.1 Location files (.LOC)

Location files are comma separated ASCII text files describing point locations. Each line describes one point, with a name followed by the XY co-ordinate in metres, e.g.:

```
ID, x, y
Hospital, 230456, 455343
Health post, 566343, 344675
```

A location file can, as an option, contain a third number on each line which represents a data value, eg:

```
ID, x, y, z
Hospital, 230456, 455343, 134.5
Health post, 566343, 344675, 87.6
```

A location file can also be used for polygon or line data, for instance:

```
ID, x, y, z
Hospital
230456, 455343
230461, 455355
230470, 455351
end, 134.5
Health post
566343, 344675
566380, 344681
566356, 344712
566343, 344675
end, 87.6
```

In this example the second object is interpreted as a polygon since the first and last co-ordinates are the same. Any attributes, such as here the z value, come after the word "end".

The use of LOC files for polygons is generally discouraged since they cannot handle polygons with holes in them. LOC files are primarily intended as a simple way to record point data.

1.2 XY survey files (.XY)

See Part 1 – Chapter 8 for the specification.

1.3 Project files (.GEO)

Project files use the Windows convention for configuration data stored in a text file in the form of groups of key-value pairs. A “group” is identified by a name in square brackets, and keys are followed by an equals sign and then the value:

```
[group]
key=va lue
```

The order of the groups within the file and of the key-value pairs within a group is not critical and should not be relied on. A project file can contain a wide variety of information. Here we will just describe the basics of a conforming project file. Such a file could be generated by a third party program and displayed in Map Maker. As a minimum the file requires a “project” group, a “map” group, and a group for each layer in the map. E.g.:

```
[proj ect]
type=Map
title=Afghani stan study
author=John Smi th

[map]
x mi n li mi t=320979. 000000
y mi n li mi t=415320. 000000
x ma x li mi t=1798836. 000000
y ma x li mi t=1529243. 000000
centre x=1066677. 349943
centre y=994327. 699884
metres per pixel =500. 000
layers=2

[Map-layer 1]
name=Terrain
di rectory=c: \Map Data\Countr y\AFGHAN\
files=4
file type=BMP
file 1=East. bmp
file 2=West. bmp
file 3=North. bmp
file 4=South. bmp

[Map-layer 2]
name=Provi nce boundari es
di rectory=c: \Map Data\Countr y\AFGHAN\
files=1
file type=DRA
```

```
file 1=PROVINCE.DRA
```

1.3.1 The project group

```
[project]
type=Map
title=Afghanistan study
author=John Smith
```

The project group contains descriptive data about the project. By convention it is usually the first group. The "type" key is necessary to identify the project as a map project to distinguish it from future project types, such a 3D images. The "Title" and "author" keys are self-explanatory.

1.3.2 The map group

```
[map]
x min limit=320979.000000
y min limit=415320.000000
x max limit=1798836.000000
y max limit=1529243.000000
centre x=1066677.349943
centre y=994327.699884
metres per pixel=500.000
layers=2
```

The map group describes the extent of the map. These limits are likely to be more than you normally see on screen, they describe the minimum and maximum extents that the user can scroll to using the scroll bars or using the guide map. The "centre x" and "centre y" values describe the centre of the view when the user first opens the project. Similarly, the "metres per pixel" value determines how much the user is zoomed in. "metres per pixel" is used instead of scale in order to make the project independent of a specific computer and display settings. Finally, "layers" describes how many layers the project has. Remember that layers is not necessarily the same as the number of files since on layer can contain several files.

1.3.3 The layer group(s)

```
[Map-layer 1]
name=Terrain
file type=BMP
directory=c:\Map Data\Country\AFGHAN\
files=4
file 1=East.bmp
file 2=West.bmp
file 3=North.bmp
file 4=South.bmp
```

For each layer there is a group named "Map-layer N", where N is the number of the layer, with layer 1 being at the bottom of the stack. The "name" value is simply a descriptive title. "file type" identifies the format of the files being displayed, e.g: BMP, TIF, DRA, SHP, DXF.

The "directory" value identifies where the files are stored. In this case the full path has been given, however there are other options:

Directory=

If directory is blank then it is assumed that the files are in the same directory as the project file.

Directory=afghanfiles

In this case the program looks for the files in a sub-directory of the directory containing the project files called "afghanfiles".

Directory=<parent>afghanfiles

Here the key word "parent" in brackets tells the program to look in the parent directory of the directory containing the project then find a sub-directory of that called "afghanfiles". Similarly you can use the key words "grandparent" and "greatgrandparent".

Directory=http://www.mapmaker.com/afghanfiles/

Directory=ftp://mapmaker.com/pub/afghanfiles/

In these ways you can identify a directory on an internet location.

Having established the location of the files the project file uses the key word "files" to declare how many files are contained in the layer, in this case 4. For each file there follows a line identifying the name of the file.

1.4 Map Maker Export files (.MME)

Map Maker Export files (*.mme) can contain points, lines, polygons, text, notes and arrows. They can also contain attribute data. They are ASCII text files which can contain a greater variety of information than the simple LOC file but which are still simple enough to be readily read and generated by third party programs. Like the project file it uses the conventions of Windows configuration files (*.ini), that is to say data of the form:

```
[group]
key=value
```

The advantages of using this format are that user-defined data can be readily accommodated and many programming platforms can read and write such files easily. The disadvantage is that if you are writing a reader for an MME file you can not assume that groups come in any particular order or that keys within a group are in any order.

Some groups and keys are obligatory while others are optional.

Comment lines can be included by starting a line with an exclamation mark "!". Also by including an exclamation mark in a line everything in the line to the right of the exclamation will be ignored.

1.4.1 Header

The minimum requirement is for the header to contain the extent of the map and the number of objects contained in the map.

```
[header]
min x=123. 456
min y=456. 789
max x=3345. 678
max y=467. 890
```

objects=23

In addition a "units" field can be included. If omitted the units are assumed to be metres. Valid values for units are:

```
units=metres
units=feet
units=yards
units=degrees
```

A "min z" and "max z" field can also be included indicating that the vector co-ordinates include a valid Z value, e.g.

```
[header]
min x=123.456
min y=456.789
max x=3345.678
max y=467.890
min z=-10.107
max z=329.621
objects=23
3D
```

The header may include other information such as the date or other application specific data.

1.4.2 Point

Each object is in a group the name of which is a serial number. For instance:

```
[object 1]
type=point
id=object 1
coordinates=1
1=123.456, 456.789
```

If the file include Z data then each co-ordinate will have an extra value:

```
[object 1]
type=point
id=object 1
coordinates=1
1=123.456, 456.789, 22.421
```

As with the header extra information may be included, "name" is the display label of the object and "style" is the style number. Other application specific data may be included:

```
[object 1]
type=point
id=object 1
name=Health centre
style=22
coordinates=1
1=123.456, 456.789
```

1.4.3 Line

The line object is similar:

```
[object 1]
type=line
id=object 1
coordinates=3
1=123.456, 456.789
2=125.876, 450.123
3=127.009, 449.198
```

1.4.4 Polygon

Similarly the polygon

```
[object 1]
type=polygon
id=object 1
coordinates=4
1=123.456, 456.789
2=125.876, 450.123
3=127.009, 449.198
4=123.456, 456.789
```

1.4.5 Complex polygon

A complex polygon is one containing islands. It is made up of two or more loops. The first loop is the main outer loop which generally contains the other polygons, though you can also have satellite polygons.

```
[object 1]
type=complex polygon
loops=2
id=object 1
```

```
[object 1-loop 1]
coordinates=4
1=123.456, 456.789
2=125.876, 450.123
3=127.009, 449.198
4=123.456, 456.789
```

```
[object 1-loop 2]
coordinates=5
1=126.456, 451.453
2=128.643, 452.654
3=129.908, 451.321
4=127.001, 453.332
5=126.456, 451.453
```

1.4.6 Text objects

```
[object 1]
type=text
id=Mountain range
height=8
justification=left
coordinates=2
1=126.456, 451.453
2=228.643, 452.654
```

The "height" value is in real-world co-ordinates (by default metres). Options for the justification key are:

```
Justification=left
Justification=right
Justification=centre
Justification=stretch
Justification=curved
```

1.4.7 Arrow

```
[object 1]
type=arrow
id=This points to the town
coordinates=2
1=123.456, 456.789
2=125.876, 450.123
```

1.4.8 Note objects

```
[object 1]
type=note
id=My data !this value is shown on the note's flag
name=c:\my data\view.bmp ! contents of the note
style=2
colour=255, 255, 200
coordinates=1
1=126.456, 451.453
```

The type of note object is determined by the "style" value:

- 0 = simple text note
- 2 = bitmap image (bmp, jpg)
- 4 = document (doc, pdf, rtf)
- 6 = multimedia file (e.g. avi, wav)
- 8 = program (exe)
- 9 = web page (htm)
- 10 = enhanced metafile (emf)

1.4.9 Data in an export file

You can include data values in an export file. To do this the header should include the field definitions, e.g:

```
[ header ]
mi n x=123. 456
mi n y=456. 789
max x=3345. 678
max y=467. 890
obj ects=23
fi el ds=5
fi el d 1=I, SPECIES
fi el d 2=R, PH, 2
fi el d 3=B, TREATED
fi el d 4=D, INSPECTED
fi el d 5=S, NAME, 40
```

In this example there are five fields (columns) in the database. Field 1 is being declared as an integer field called "SPECIES". Field 2 is a "real" (ie decimal number) rounded to two decimal places in this case. Field 3 is a boolean (logical or Yes/no) value. Field 4 is a date. Field 5 is a string of maximum length 40 characters.

Note that when Map Maker extracts data from a MME file it create an additional field which contains the object ID. This field is called ID. So though, in this example, there are five fields there is an implicit sixth field containing the ID.

For each point, line, or polygon object there should be a corresponding entry:

```
[obj ect 1]
type=pol ygon
i d=obj ect 1
fi el d 1=5
fi el d 2=6. 78
fi el d 3=Y
fi el d 4=23/6/2000      !note this is in day, month, year format
fi el d 5=Lower meadow
coordi nates=4
1=123. 456, 456. 789
2=125. 876, 450. 123
3=127. 009, 449. 198
4=123. 456, 456. 789
```

1.5 Geo-referenced web pages

Map Maker can display geo-referenced web pages (*.HTM). For an HTM document to be compatible it requires one or more special GEO keys in the document header. A typical HTM document header looks something like:

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html ;
charset=windows-1252">
```

```
<META NAME="Generator" CONTENT="Microsoft Word 97" >
<META NAME="KeyWords" CONTENT="Wildlife, Trees" >
<TITLE>Today's news</TITLE>
</HEAD>
```

To make the document geo-referenced requires the addition of one line named "GEO REFERENCE":

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=windows-1252" >
<META NAME="Generator" CONTENT="Microsoft Word 97" >
<META NAME="KeyWords" CONTENT="Wildlife, Trees" >
<META NAME="GEO REFERENCE" CONTENT="456.78, 2012.89" >
<TITLE>Today's news</TITLE>
</HEAD>
```

You can also specify a date and time that can be used to automatically clear stale items.

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=windows-1252" >
<META NAME="Generator" CONTENT="Microsoft Word 97" >
<META NAME="KeyWords" CONTENT="Wildlife, Trees" >
<META NAME="GEO REFERENCE" CONTENT="456.78, 2012.89" >
<META NAME="Geo DateTime" CONTENT="15 February 2001 10:30" >
<TITLE>Today's news</TITLE>
</HEAD>
```

2 Binary file formats

2.1 Drawing files (.DRA)

Drawing files are binary files describing vector lines, polygons, symbols, text, and arrows. Each DRA file starts with a file header object. Every object starts with a 24 byte object header of the following form:

Offset	Size in bytes	Content
0	1	signature byte, always equal to 237 + the current version number, which is 3
1	1	object type 1 = Line 2 = Polygon 3 = Point 4 = Text 5 = Arrow 10 = File header 12 = Note

2	4	a 32 bit integer describing the total object size in bytes
6	4	rounded down Minimum X as 32 bit integer
10	4	rounded down Minimum Y as 32 bit integer
14	4	rounded up Maximum X as 32 bit integer
18	4	rounded up Maximum Y as 32 bit integer
22	2	size as a word in bytes of the attribute record or in file headers this value gives the number of objects that it contains

Other object types can be found in DRA files. The types described here are just the common types. If you write your own code to read DRA files they should be able to deal with objects where the object type (identified in the second byte of the object header) is not one of the standard ones by discarding and stepping over them.

An object can consists of four parts: Object header, Attribute record, Geometry records, and Label record.

Object type	Object header	Attribute record	Geometry records	Label record
1. Line	x	x	N	x
2. Polygon	x	x	N	x
3. Point	x	x	1	x
4. Text	x	x	N	
5. Arrow	x	x	2	
10. File header	x			
12. Note	x	x	1	

After the object header there is an attribute record. After the attribute record there follows a number of 13-byte geometry records each of the form:

Offset	Size in bytes	Content
0	6	X ordinate as a six-byte real
6	6	Y ordinate as a six-byte real
12	1	attribute flag – mixture of these values 16 = corner rather than simple vertex 128 = polygon corner before an island starts

Note that the X and Y values use six byte reals. These are of the “pascal” format.

The attribute records vary with the different object types:

Line and polygon attribute record

Offset	Size in bytes	Content
0	1	a byte describing the display style
1	1	Attribute flags 128 = contains islands
2	8	Eight one-byte data values (user-defined)
10	1	Size in bytes of object name - N1
11	N1	Object name as a string
11+N1	1	Size in bytes of object ID - N2

12+N1	N2	object's Unique ID as a string
12+N1+N2	1	spare

After the header there follows a number of geometry records, calculated from the object size and the attribute size.

Point attribute record

Offset	Size in bytes	Content
0	1	a byte describing the display style
1	1	Attribute flags Not used
2	8	Eight one-byte data values (user defined)
10	1	Label justification code
11	1	Size in bytes of object caption - N1
12	N1	Object caption as a string
12+N1	1	Size in bytes of object ID - N2
13+N1	N2	object's Unique ID as a string
13+N1+N2	1	spare

After a point attribute record there is always one geometry record.

Text attribute record

Offset	Size in bytes	Content
0	1	a byte describing the display style
1	1	spare
2	1	Justification code
3	6	Text height in metres as a six-byte real
9	2	Rotation in tenths of a degree, 2 byte integer
11	1	Size in bytes of object caption - N1
12	N1	Object caption as a string

The number of geometry records varies with the justification code:

Justification code	Number of geometry records	Description of type
0	1	Left justified text
1	1	Centre justified text
2	1	Right justified text
3	2	Stretched text
4	3	Curved text
5	4	Bezier curved text
6	N	Bezier curved text

Arrow attribute record

Offset	Size in bytes	Content
0	1	a byte describing the display style
1	1	reserved
1	1	reserved
3	1	Size in bytes of object caption - N1
4	N1	Object caption as a string

Note attribute record

Offset	Size in bytes	Content
0	1	a byte describing the note style 0= simple note 2= bitmap (bmp or jpg) 4= document (doc, pdf, rtf) 6= media (avi,qt,wav) 8=program 9=web page (htm) 10=emf 11=rich text note
1	1	Attribute flags Not used
2	4	32-bit integer for the flag colour
6	5	spare
11	1	Size in bytes of the note contents - N1 (see note below)
12	N1	Note contents as a string (eg filename)
12+N1	1	Size in bytes of object ID - N2
13+N1	N2	object's Unique ID as a string
13+N1+N2	1	spare

After a note attribute record there is always one geometry record.

Note that when the note contents is longer than 254 characters (such as is commonly the case with a rich text note object then byte 11 is 255 and is followed by a two byte word giving the length of the contents that follow.

Label location record

The label location record used by point, polygon, and line objects is 13 bytes long

Offset	Size in bytes	Content
0	4	100 x Label X as a rounded integer
4	4	100 x label Y as a rounded integer
8	2	Label rotation in tenths of a degree as a two-byte integer
10	1	Justification code 0 = left justified 1 = centre justified 2 = right justified
11	1	100 x multiplication of font size as a rounded byte
12	1	attribute flag – always 64 so that it can be distinguished from a geometry record.

Label X and Label Y are a displacement of the label from the default label insertion point. For a point object this is the co-ordinate of the point. For a line it is the centre point of the middle segment of the line (where there is an even number of segment the segment used is the one before the middle vertex). For a polygon the default insertion point is the centre of the bounding box of the polygon.

2.2 Terrain files (.TER)

Terrain files are used by Map Maker's 3D module (see Part 5). They describe altitude as a continuous surface over a rectangular area by heights at the grid intersections of a square grid.

Offset	Size	Data type	Description
0	1	byte	Currently 3
1	1	Byte	Flags 0 = values are encoded 2 byte words (see below) 2 = values are 4 byte reals (singles)
2	1	Byte	Unites 0 = metres 1 = lat/long
3	1	Byte	Reserved
4	4	single	Left X in metres
8	4	Single	Bottom Y in metres
12	4	Single	Minimum Z in metres
16	4	Single	Maximum Z in metres
20	4	Single	Cell width in the X direction in metres
24	4	Single	Cell width in the Y direction in metres
28	4	Integer	Number of rows (e.g. 100 cells is 101 rows)
32	4	Integer	Number of columns (e.g. 100 cells is 101 columns)
36	If flags=0 then Rows x Columns x 2 else Rows x Columns x 4		Data values, if flags =0 or 1 then the result is in metres, if flags=2 then the result is in millimetres.

Encoded altitudes

The purpose of encoded altitudes was to use simple 2 byte unsigned integers (words) to achieve a finer resolution at low altitudes than at high altitudes (or great depths) while keeping a compact file size. The encoding results in a resolution of 0.1 metres at low altitudes (less than 1,000m). The altitude range is designed to encompass the range found on the earth's surface. The encoded integer can be decoded as follows.

If code > 60,000 then decoded altitude = $7,000 + (\text{code}-60,000) / 2.5$
 Else If code > 40,000 then decoded altitude = $3,000 + (\text{code}-40,000) / 5$
 Else If code > 10,000 then decoded altitude = $(\text{code}-10,000)/10$
 Else If code > 9000 then decode altitude = $(\text{code} - 10,000) / 10$
 Else If code > 5000 then decoded altitude = $(\text{code} - 9,200) / 2$
 Else Decode altitude = $\text{code}-7,100$

2.3 Calibrated TIFF files (.TIF)

A normal Aldus TIFF (Tagged Image File Format) file fresh from a scanner has no scale, though its header may or may not have a figure containing the dots per inch at which it was scanned. The calibration process of Map Maker adds a data block to the end of the TIFF file of the following format:

Offset	Size	Data type	Content
file size - 24	6	Pascal Real	Left X as a global ordinate as a six-byte real
file size - 18	6	Pascal real	Bottom Y as a global ordinate as a six-byte real
file size - 12	6	Pascal real	Scale on paper of the original
file size - 6	2	Word	Dots per inch of the scan
file size - 4	2	Word	reserved
file size - 2	1	Byte	signature currently 4
file size - 1	1	byte	signature always 237

Sometimes there is an extra calibration block though this is not currently used:

Offset	Size	Data type	Content
file size - 50	6	Pascal Real	Reserved
file size - 44	6	Pascal real	Reserved
file size - 38	6	Pascal real	Reserved
file size - 32	6	Pascal real	Reserved
file size - 26	1	Byte	signature currently 4
file size - 25	1	byte	signature always 237

2.4 Calibrated BMP files (.BMP)

A normal Windows BMP (Bitmap) file fresh from a scanner or screen grab has no scale, though its header may or may not have a figure containing the dots per inch at which it was scanned. The calibration process of Map Maker adds a data block to the end of the BMP file which is the same as for the TIF file described above.