Enigma Raster map file format

This document describes the Enigma raster map file format.

This file format has been created to support the Enigma series of EFIS instruments from MGL Avionics.
MGL Avionics grants any interested party the right to use this airspace format whether this is in support of a MGL Avionics product or not.

Any party making use of this document and described file format will do so at their own discretion, responsibility and risk.

The party further acknowledges that any data it receives howsoever in this format may be faulty, incorrect, incomplete or outdated and that the party bears responsibility for any consequence that may arise out of this.
Data may have been compiled from untraceable sources and is included without assumptions of fitness for any purpose. All data should be treated with circumspection and verified with independent sources before acceptance and use.

MGL Avionics places this document and the data format it describes in the public domain in order to foster the use of a common, compact raster file format that is particularly optimized for use with resource limited systems and/or systems that require fast access to individual items in the airspace file in order to increase overall systems performance.

In order to allow recognition of the format any party adopting this format agrees to refer to this format as the “Enigma map raster format”. All data using this format should be considered to be in the public domain and the implementer will make reasonable effort to allow any interested party to use the data free of cost with the exception of a reasonable fee for distribution and media.

Exceptions:
MGL Avionics reserves copyright and denies use of this data format for any military or related purpose.

Comments are welcome: info@delta-omega.com

Raster file format

The Enigma raster file format has been created to allow a seamless map image to be created from many different map files that may contain map data in different resolutions. In addition, several techniques have been incorporated to compress the raster map data in order to reduce file size. The compression technique used has been specifically selected for fast decompression as this is a major requirement for typical EFIS instrumentation.

Map Projection

Map data contained in these files is area and distance linear. The projection is a constantly variable projection, much like Mercator but with lines of longitude always exactly vertical at a particular point of interest. This means that in the northern hemisphere, lines of longitude would slope upwards and inwards as one moves west or east from the point of interest. In the southern hemisphere the slope would be downwards and inwards.

Maps are represented as a linear, two dimensional array of pixels. Each pixel is represented using eight bits. A few of the possible codes are reserved for special purposes, the remainder form an index to a fixed color palette. The colors chosen for the palette is a compromise to allow for a wide range of typical aviation maps from various countries. Map creation software (such as the MGL Avionics MapMaker program) uses a nearest color matching algorithm for every pixel from the source map image.

Maps are represented as degree tiles in the map file. A single map file contains at least one degree tile but in practice contains several tiles. A typical map format may be a 6x4 array of degree tiles for example.

Each tile has a fixed vertical pixel resolution dependent on the underlying map resolution. This resolution may be one of:

- 150 pixels/degree
- 300 pixels/degree
- 600 pixels/degree
- 1200 pixels/degree
- 2400 pixels/degree

Most maps tend to be created using 1200 or 600 pixels per degree.

The horizontal number of pixels per degree tile is identical to the vertical number of pixels only at the equator. As one moves away from the equator, the number of pixels per horizontal line reduces in order to remain area and distance correct. The Enigma format uses the following format to calculate the number of horizontal pixels for every line.

Number of horizontal pixels = cos(latitude)*vertical number of pixels.

Each map line of data is prefixed by the number of pixels contained in the line so this calculation may or may not be required in the map drawing software.

General file name format

The filename is of importance. A DOS 8:3 format has been adhered to. The filename is built up as follows: Latitude+Longitude of top left hand corner of map + map resolution +'.M'+width+height in degree tiles.

Example:

S23E026b.M42 This map file has a top left corner at position Latitude S23, Longitude E026 degrees. The map has a base resolution of 1200 pixels per degree. The file contains eight degree tiles arranged 4 horizontal in two rows.

Map resolution identifiers:

- a – 2400 pixels/degree
- b – 1200 pixels/degree
- c – 600 pixels/degree
- e – 300 pixels/degree
- f – 150 pixels/degree

Maps should not have more than 9 tiles horizontal and 9 tiles vertical.

File format

The file starts with a fixed header as follows:

FileID:	4 characters “MGLM”
PointerToRasterImage:	pointer to start of Raster data, 32 bits.
PointerToTerrainData:	pointer to start of terrain data, 32 bits.
PointerToVectorData:	pointer to start of Vector data, 32 bits.
PointerToVendordata:	pointer to start of Vendor data, 32 bits.
Latitude:	Top left corner latitude 16 bit signed.
Longitude:	Top left corner longitide 16 bit signed.
SizeDegreesHorizontal:	Horizontal number of degree tiles. 16 bit unsigned.
SizeDegreesVertical:	Vertical number of degree tiles. 16 bit unsigned.
Resolution:	Resolution. 2400 pix/degree = 0, 150 pix/degree = 4

Currently, only the PointerToRasterImage is used as data pointer to map data. Other pointers are reserved for possible future use.

The PointerToRasterImage points to the byte location of the first byte of raster image data.

This location contains a further list of pointers. Each degree tile has one pointer. If there are eight degree tiles in the file, you have eight pointers. The first pointer points to the top left degree tile. This is followed by the pointer to the degree tile to the right of the first one (if more than one horizontal tile). This is repeated for each row.

Following the pointer to a degree tile we get:

Degree tile data format

Each degree tile starts with a list of pointers to each line of map image data. It follows that this list of pointers depends on the number of vertical pixels per degree (the map resolution). If the map resolution is 1200 pixels per degree, you have 1200 pointers in this list. The first pointer points to the northernmost line of map image data for this tile.

Please note: These pointers are 24 bits in size (three bytes) and are relative to the location of the first pointer in the file. Take the file address of the first pointer and add the relative address to obtain the absolute file address of the line data.

Line data format

Each line follows the following format:

- Number of pixels in line – 16 bit unsigned. This number decreases as you move towards the poles.
- Number of data bytes for line following – This number will likely be less than the number of pixels in the line as the data is compressed. Please note that in occasional cases the number of bytes may be slightly larger than the number of pixels in the line.
- Compression format identifier. 1 Byte. Currently set to 1 = RLE compression. A value of 0 means no compression (currently not used). A value of 2 means LZH compression. This is currently not used either as decompression takes too much CPU time.
- Data for line (number of bytes specified above).

RLE compression format

The RLE compression format is simple and results in fast decompression. Many map images compress well with this method if areas of identical color are present.

The data starts with a mode/length byte.

If the byte is >0x80 then the least significant 7 bits contain a count of bytes to write using the 8 bit color value following in the next byte.

If the byte is <0x80 then the byte is a count of bytes to read directly from the source (starting with the next byte) – no compression in this case.

This is repeated until the required number of bytes have been assembled in the target image line, i.e, if required another mode/length byte follows.

Further processing

After the source line has been decompressed, it may need to be scaled if any zoom is used on the image. For example, you might need only every second byte or need to use every byte twice or more times to create your final target image. For the same reason, you might not read all the source lines but skip lines as required or duplicate them in your target image. This should be done in consultation with the map resolution as you may well find two files with tiles next to each other that have been created in different base resolutions.

Color palette

The color palette for the map can be easily created using the following Pascal code:

The first 16 colors are identical to the standard VGA 16 color system.

```
 Colortable[0]:=Graphics.clBlack;
 Colortable[1]:=Graphics.clMaroon;
 Colortable[2]:=Graphics.clGreen;
 Colortable[3]:=Graphics.clOlive;
 Colortable[4]:=Graphics.clNavy;
 Colortable[5]:=Graphics.clPurple;
 Colortable[6]:=Graphics.clTeal;
 Colortable[7]:=Graphics.clGray;
 Colortable[8]:=Graphics.clSilver;
 Colortable[9]:=Graphics.clRed;
 Colortable[10]:=Graphics.clLime;
 Colortable[11]:=Graphics.clYellow;
 Colortable[12]:=Graphics.clBlue;
 Colortable[13]:=Graphics.clFuchsia;
 Colortable[14]:=Graphics.clAqua;
 Colortable[15]:=Graphics.clWhite;
 //This is followed by an array of calculated color values
 i:=16;
 for red:=0 to 13 do
   for green:=0 to 4 do
     for blue:=0 to 2 do
     begin
       Vblue:=blue*128;
       if VBlue>255 then VBlue:=255;
       Vgreen:=green*64;
       if Vgreen>255 then Vgreen:=255;
       Vred:=red*20;
       if Vred>255 then Vred:=255;
       c:=(Vblue shl 16)+(Vgreen shl 8)+Vred;
```

```
        if (c<$FFFFFF) and (c>0) then
        begin
          ColorTable[i]:=c;
          inc(i);
        end;
      end;
    //This is followed by gray-scale values
    for x:=2 to 15 do
    begin
       c:=(x*16 shl 16)+(x*16 shl 8)+x*16;
       colortable[i]:=c;
       inc(i);
    end;
    //finally this is followed by a few specific colors
    for x:=0 to 7 do
    begin
       c:=(DEFC[x,2] shl 16)+(DEFC[x,1] shl 8)+DEFC[x,0];
       colortable[i]:=c;
       inc(i);
    end;
    //where the specific colors are defined as follows:
const
  DEFC: array[0..7,0..2] of byte =
    ((200,150,50),
     (211,165,72),
     (217,179,90),
     (229,194,108),
     (239,213,133),
     (247,231,160),
     (245,250,171),
     (252,253,208));
```

General comments

In order to properly render a map in Enigma raster format, the programmer must understand the way each source line is assembled in the correct location on the display device. The method is chosen such that the line of longitude is exactly vertical at a specific location on the screen, usually the screen center. This means that as you draw a new line, the line data may need to shift left or right in order to make this possible.

You will notice that the amount of pixels per line decreases or increases with latitude (depending on your hemisphere). Your drawing routine must take this into account and will assemble an entire line from perhaps several degree tiles, each scaled correctly if tiles originate from different files at different resolutions.

Correctly done, you can assemble a seamless image that (at least theoretically) spans the globe.

Please note that the Enigma map format is not intended for use above 80 degrees of latitude as the number of horizontal pixels becomes very small per degree tile and vanishes to zero at the poles.