# Exploring Garmin's IMG Format

# TRE, RGN, LBL, NET, NOD & DEM



#### (c) N Willink 21/08/2011

If you downloaded the manual from another source it will not be up to date.

more information on www.pinns.co.uk/osm/garmin.html

Latest Revision 02/03/2015

Exploring IMG Format

|   | ••••• |
|---|-------|
| TDB Editor 1.1  | 4     |
| Download & checkout the only tdb editor to:                     | 4     |
| Introduction  | 4     |
| Introduction  | 5     |
| Garmin NT   | 5     |
| IMG2TYP   | 6     |
| Garmin headers  | 7     |
| RGN header  | 7     |
| How are all the elements (pois, polylines and polygons) stored? | 8     |
| Map levels  | 8     |
| Subdivisions  | 9     |
| Pointers in a subdivision                                       | 10    |
| Finding number of pointers for each subdivision                 | 11    |
| Start of Subdivisions in RGN + &29                              | 13    |
| Locked TOPO maps  | 17    |
| LBL   | 18    |
| Lbl pointers NOT directly pointing to lbl1 label block          | 20    |
| Symbols   | 21    |
| More than 1 label in NET  | 21    |
| POIs  | 22    |
| POIs with subtypes  | 22    |
| POIs with no Subtypes   | 22    |
| POI labels  | 22    |
| POIs with extended types  | 23    |
| POLYLINES   | 24    |
| Polyline Labels   | 24    |
| Polylines with extended types 0 x 100+                          | 25    |
| Length of a polyline type 100+ block                            | 25    |
| POLYGONS  | 26    |
| Polygon Labels  | 26    |
| Polygons with extended types 0 x 100+                           | 26    |
| Length of a polygon type 100+ block                             | 26    |
| Plotting Coordinates  | 27    |
| Plotting POIs   | 27    |
| Plotting Polylines  | 28    |
| the first bitstream byte  | 28    |
| The 'official' algorithm:                                       | 29    |
| Starting to parse bitstreams                                    | 31    |
| Plotting Routable polylines                                     | 33    |
| Left_shifting Coordinates                                       | 34    |
| Left_shifting Coordinates                                       | 34    |
| Bits_per_coord  | 34    |
| Left Shifting   | 34    |
| Plotting Polygons   | 36    |
| Special cases in a bitstream                                    | 36    |
|   |       |

Exploring IMG Format

| ГRЕ from 0x4a         | . 38 |
|-----------------------|------|
| ГRЕ7                  | . 40 |
| ΓRE8                  | . 41 |
| ГRЕ9                  | . 41 |
| NET subfile           | . 42 |
| NET1                  | . 42 |
| Length of highways    | . 43 |
| NOD subfile           | . 44 |
| NOD 1                 | . 44 |
| Pointer               | . 45 |
| Flags at offset 1     | . 45 |
| Direction Coordinates | . 45 |
| Nodes Bytes           | . 45 |
| Flags A & B           | . 45 |
| Tables Header         | . 46 |
| Table A               | . 46 |
| NOD 2                 | . 47 |
| DEM subfile           | . 49 |
| Creating IMG files    | . 50 |
| NT POIs               | . 50 |
| Extra POI data stream | . 51 |

## **TDB Editor 1.1**

#### Download & checkout the only tdb editor to:

- lock & unlock your Mapsource/Basecamp maps.
- restore Mapsource when it crashes
- rename any mapsource/basecamp name
- change mapnames so they are easier to read on your gps
- Lock & Unlock your own maps
- add/remove routable option
- force direct routing when it doesn't work
- *add/remove profile option.*
- add or edit copyright description when creating your own maps
- create or remove 'Reduced map window for printing' and other (printing or viewing) messages
- *disable any form of printing.*
- add or resize Restricted Window View
- quickly identify & edit mapnames, useful when you want to isolate a particular IMG file.



## Introduction

For anyone who wants to delve more deeply into the IMG structure John Mechalas (JM)' 'Garmins IMG File Format' is a must read. We all owe a great deal to the author of cgpsmapper without whom the booklet could not have existed.

Apart from this excellent document there seems to be nothing else available. Unfortunately, there are some crucial errors in the JM's description, making it more even more challenging.

Our aim is to tie up some loose ends and use frequent examples to make the format more accessible. However, the IMG structure may to some still be a steep learning curve.

In this ongoing document I intend to focus on how to access and plot poi, polyline and polygon data within an IMG file. From the picture in the title you can see that, my humble efforts after 3 years of frustration, have finally been successful. There were many occasions when I felt like giving up as I was getting nowhere really fast.

*IMG Explorer* shows pointers and coordinates held in an IMG file and can be downloaded on request only.

We would welcome any additional information or queries using email address found at : <a href="http://www.pinns.co.uk/osm/">www.pinns.co.uk/osm/</a>

### Garmin NT

At present this document does not address Garmin's new NT format – more information about this subfile can be found at <u>openstreetmap</u> <u>GMP wiki</u>. Suffice it to say, the img structure of pointers is not much different and the offsets are calculated differently. With active routing additional information is attached to each line. Its routing compaction is also more efficient.

NT caters for postcode searches and routing using multiple highway lanes.



For this reason, there doesn't appear to be a viable incentive to reverse engineer this format.

*IMG Explorer* can ,however, extract all tiles contained in the file and gives the starting address for each tile. The LBL contains the ref names of tiles followed by pointers to each tile taken from the end of lbl1,

## IMG2TYP

I've written this GUI to read IMG files and list all element types found in TRE so that they can be saved as a TYP file. Check out on youtube



### TYPWiz

Use TYPWiz in connection with img2typ and you are well on your way to changing the rather drab TYP files which accompany CN 2012-2015 maps ! Check out on Youtube

| le Settings          | Icons         | Languag    | es Re   | gistration  | Help       |           |             |            |           |          |          |            |              |      |          |          |       |      |
|----------------------|---------------|------------|---------|-------------|------------|-----------|-------------|------------|-----------|----------|----------|------------|--------------|------|----------|----------|-------|------|
| • 128                |               | 77         |         | 225         | FID:       | 2635      | ] 0         | ode 125    | 2 🕶       | Crea     | ted 2683 | 2012 15:58 |              | L    |          |          |       |      |
| Polygons             | m m           | Wiz Lite 1 | L1 [Not | Registered  | I) K:\a_ty | pwiz3_9\A | IT 2013 C   | iountries) | 12 europe | U0000A4  | B.TYP    |            |              |      | X        |          |       |      |
| 1                    | File          | Settings   | Icons   | Languages   | Regist     | ration H  | elp         |            |           |          |          |            |              |      |          |          |       |      |
|                      |               | 128        | •       | 77          | .15        | 25        | FID: 2      | 635        | Code      | 1752     | . 0      | reated 26  | 8 201 2 15 5 | 18   |          |          |       |      |
| 5 5<br>x0100 0x0200  | Pa            | Amore      | TYP     | Viz Lite 1. | 1 [Not f   | Registere | d] K:\a_t   | typwiz3_   | 9\NT 201  | 13 Count | tries\12 | europe\l   | 0000A4B      | TYP  |          | l        |       | ) X  |
|                      |               | F          | ile Se  | ettings I   | cons L     | anguage   | es Regi     | stration   | Help      |          |          |            |              |      |          |          |       |      |
|                      |               | -          | +       | 128         | +          | 77        | •           | 225        | FID:      | 2635     |          | Code 12    | 252          |      | reated 2 | 6 8 2012 | 15:58 |      |
| x1600 0x1700         | 0x01          | 0:00       | Poly    | aons        | Lin        | es        | Poi         | is         | PID:      | 1        | Пне      | ader Al    | F            |      |          |          |       |      |
|                      |               | L          |         |             | L          |           |             |            |           |          | _        |            |              |      | -0-      |          | -     |      |
| 9 9                  |               | -          |         |             |            |           |             |            |           |          | 10       |            |              |      | 0        |          | O     |      |
| x4408 0x4600         | 0x14          | Oxt        |         |             |            |           | -           |            | ۳         |          | <u> </u> |            | •            |      |          | ري       |       | -    |
|                      | 1             | -T         | ZAUZ    | 2801        | 2802       | 2803      | 2804        | 2001       | 2002      | 2003     | 2004     | 2005       | 2006         | 2007 | 2008     | 2009     | ZCUA  | 2C0B |
| 6 6<br>x10900 0x1090 | -             |            | R       | 65          | Б          | <b>F</b>  | 4           |            | V         |          |          |            |              |      | യ്       | <b>a</b> | A     |      |
|                      | 0x109         | 00105      | 200E    | 2001        | 2002       | 2003      | 2004        | 2005       | 2006      | 2007     | 2008     | 2009       | 200A         | 2D0B | 2E02     | 2E04     | 2E05  | 2E0A |
| 12 5                 |               | _          | LOOL    | 2001        | 2002       | 2005      | 2004        | 2000       | 2000      | 2007     | 2000     | 2005       | 2007         | 2000 | ELUE     | 22.04    | 2205  | 2Lun |
|                      | Dx10E         | C6 0x108   | Ð       |             | Ŧ          | 不         | $\boxtimes$ | €          | -         | 3.       | P        | Î          | -            | -    |          |          |       |      |
|                      | ANO           | WIRLS A    | 2F01    | 2F02        | 2F03       | 2F04      | 2F05        | 2F06       | 2F07      | 2F09     | 2F0B     | 2F0C       | 2F0D         | 2F0E | 3001     | 3002     | 3003  | 3004 |
|                      |               |            | _       |             |            |           |             |            |           |          |          |            |              |      |          |          |       |      |
|                      | <b>b</b> x10E | 16 0×10    | 白       | $\Theta$    | E          | -         | 开           |            | i         | İ.       | 5        | 0          |              | X    |          | *+       | 놊     |      |
|                      |               |            | 3005    | 3006        | 3007       | 4100      | 4A00        | 4800       | 4C00      | 4E00     | 5000     | 5200       | 5400         | 6401 | 6402     | 6403     | 6404  | 6407 |
|                      |               | r          |         |             |            |           |             |            |           |          |          |            |              |      |          |          |       |      |
|                      |               |            | Δ.      | A           | A          | i.        | ٠           | +          | =         |          | ۰        | ٠          | ٠            | ٠    | 22       | ٠        | Q4    |      |
|                      |               |            | 640C    | 640D        | 6411       | 6415      | 6501        | 6502       | 6508      | 650D     | 6511     | 6512       | 6513         | 6601 | 6604     | 6606     | 660A  | 6616 |

6

## **Garmin headers**

An IMG file consists of several blocks, or sub files, each doing a specific job:

TRE, RGN, LBL , NET , NOD

The one we're particularly interested in is the RGN subfile as we want to establish how all highways, polygons and points of interests are plotted.

## **RGN header**

In the RGN subfile we find the essential data for plotting our elements: coordinates, ,length of highways, number of sides in each polygon etc. The header looks like this:

| RGN Offset | RGN Header   | BYTES |
|------------|--|-------|
| 00         | Header Length  |       |
| 02         | GARMIN RGN   | 4     |
| 15         | Pointer to beginning of RGN1 data, ie first subdivision to include           | 4     |
|            | possible POIs, Indexed POIs, Polylines or Polygons or first map level        |       |
| 19         | Length of this block   | 4     |
| 1D         | Pointer to RGN2 data block, contains <b>extended</b> polygons with types 0 x | 4     |
|            | 100+; for undocumented details about its structure see further               |       |
| 21         | Length of this block   | 4     |
| 39         | Pointer to RGN3 data block ; this block contains extended polylines          | 4     |
|            | with types $0 \ge 100+$ ; for undocumented details about its structure see   |       |
|            | further  |       |
| 3D         | Length of this block   | 4     |
| 55         | Pointer to RGN4 block containing extended POIs ; for undocumented            | 4     |
|            | details about its structure see further                                      |       |
| 59         | Length of this block   | 4     |
|            |  |       |
| 65         | FF   |       |
| 66         | 3F (7F) (03)   |       |
| 67         | 0  |       |
| 68         | 20 3f f7 ff 3f   | 4     |
| 6D         |  | 4     |
|            |  |       |
| 75         | Block or length  | 4     |
| 79         | 0x E3 (E5)3f   |       |

## How are all the elements (pois,polylines and polygons) stored?

You could imagine that, as maps rely on individual nodes, pois are plotted first, followed by the polygons etc. Yes and no. To understand how they are stored we need to look at :

#### 1) maplevels

#### 2) subdivisions.

## Map levels

To save space, Garmin opted for a unique solution using maplevels and subdivisions.

To enable any kind of zooming, Garmin has decided to plot data in 'zoom chunks'; the deeper the zoom the more information it contains, ie the more highways etc are plotted.

At the lowest zoom, very few pois, if any and only a few highways are plotted - each created using only a limited number of nodes, thus making them look straighter and more ragged .

These zoom levels are called maplevels ,each with subdivisions.

| Maplevel 1 | Subdivision        |
|------------|--------------------|
|            | <b>Subdivision</b> |
| Maplevel 2 | <b>Subdivision</b> |
|            | <b>Subdivision</b> |
|            | Subdivision        |
| Maplevel 3 | <b>Subdivision</b> |

#### Regard map levels as groups of subdivisions .

Each map level is given a separate data block telling you:

- 1) what elements to plot
- 2) at what resolution /degrees of accuracy
- 3) how many subdivisions it has grouped together sometimes none!

### Subdivisions

Before we retrieve these maplevel blocks, let's explore the nitty gritty of sublevels. Each subdivision contains data visible at that level. Some subdivisions can share data with others. The main highways are often plotted at different zoom levels and hence at a different resolution/ accuracy, whereas hardly any pois are plotted at the lowest level when you zoom out. So there is a certain amount of doubling up, despite Garmin's main aim to reduce its IMG file size.

How are these subdivisions constructed?

Consider the following:

- 1) POIs don't need as much data as highways; in fact they only need one node
- 2) Not all highways are the same length
- 3) Polygons are not the same shape and their number of nodes can vary too.

Unfortunately, because the pois and highways data are not the same lengths, we need to know where the starting points are for each element data block .

Ideal situation : every object the same length:

| Poi 1      |  |      |      |  |      |
|------------|--|------|------|--|------|
| Poi 2      |  |      |      |  |      |
|            |  |      |      |  |      |
| Polyline 1 |  |      |      |  |      |
| Polyline 2 |  |      |      |  |      |
| Polygon 1  |  | <br> | <br> |  | <br> |
| Polygon 2  |  |      |      |  |      |

Garmin's approach:

| Poi 1    | <br> | <br>Poi 2 | <br> |           | Polyline1 | <br>     |
|----------|------|-----------|------|-----------|-----------|----------|
|          |      |           |      | Polyline2 |           |          |
|          |      |           |      |           |           | Polygon1 |
|          |      |           |      |           |           |          |
| Polygon2 |      |           |      |           |           |          |

You could not have the 'ideal' situation unless you limit the number of nodes for each highway or polygon to say 1000. That would be such a waste if the lines are short etc.

Note :The maximum length of a subdivision is finite – the exact length is not clear - , hence the existence of numerous subdivisions depending on the amount of data within a maplevel.

### Pointers in a subdivision

As a compromise, Garmin decided to plot all pois in a subdivision in one chunk, followed by highways, followed by polygons.

To make sure we know where each element chuck starts we need to be given some **pointers**. These pointers are not saved somewhere else but are given just before the start of a chunk. More specifically, *all pointers are given before the first chunk appears, at the beginning of each subdivision and are 2 bytes long* :

| Pointer to chunk2Pointer to chunk 3Chunk 1Chunk 2Chunk 3 |  |
|--|--|
|--|--|

You would expect:

| Pointer to | Pointer to | Pointer to | Chunk 1 | Chunk2 | Chunk3 |
|------------|------------|------------|---------|--------|--------|
| chunk1     | chunk 2    | chunk 3    |         |        |        |

To save space we don't need a pointer to the first element as it follows after the pointers, if any. We can skip the pointer to chunk1 ;as each pointer is 2 bytes long , we know where chunk 1 starts (6 bytes from the offset).

Example:

| Pointer to chunk2 | Pointer to chunk 3 | Chunk 1 POIs | Chunk 2   | Chunk 3  |  |
|-------------------|--------------------|--------------|-----------|----------|--|
|                   |                    |              | Polylines | Polygons |  |

So now we know the start of each element chunk.

There are 2 problems with this solution:

- 1) we need to know how many pointers there are in each subdivision
- 2) we need to know the length of each subdivision

# *If we didn't know how many pointers there were, we wouldn't know where our first element chunk started* – for an answer see later .

Again, supposing we knew the number of pointers, we could find the first element block, but we wouldn't know where the next *subdivision* started, ie with its pointers

We can't tell, unless the number of pointers & the start of each subdivision is stored elsewhere in the IMG. They are not found in the RGN but in another subfile, called the TRE - see below

### Finding number of pointers for each subdivision

The information is found in the TRE subfile, specifically at an offset found at TRE + &29, or &2429: ie offset value = &1BF. Add this to &2400.

The dark blue line points to blocks of 16 bytes, each block representing a subdivision containing a number of elements.

The first 3 numbers indicate offsets in the RGN (in pink), the fourth number of each block of 16 bytes tells you what elements you can find each map level, ie pois and or highways etc, underlined in light blue – ie the first map level = 0, second CO, then CO, DO,DO.

| 2400 | BC | 00 | 47 | 41 | 52  | 4D | 49 | 4E | 20 | 54 | 52 | 45  | 01  | 00  | DB  | 07  | 4 GARMIN TRE -0.   |
|------|----|----|----|----|-----|----|----|----|----|----|----|-----|-----|-----|-----|-----|--|
| 2410 | 07 | 15 | 12 | 12 | 3A  | 06 | 15 | 24 | 4D | 88 | FD | 97  | 12  | 24  | 83  | 88  | • <sup>⊥</sup> <sup>†</sup> <sup>†</sup> :- <sup>⊥</sup> SMcÿ <sup>†</sup> <sup>†</sup> <sup>†</sup> |
| 2420 | FD | A7 | 01 | 00 | 00  | 18 | 00 | 00 | 00 | BF | 01 | 00  | 00  | 62  | 00  | 00  | ýs t   |
| 2430 | 00 | 21 | 02 | 00 | 00  | 09 | 00 | 00 | 00 | 03 | 00 | 00  | 00  | 00  | 00  | 00  | -19 6  |
| 2440 | 0A | 00 | 00 | 01 | 03  | 11 | 00 | 01 | 00 | 00 | 31 | 02  | 00  | 00  | 10  | 00  | La   |
| 2450 | 00 | 00 | 02 | 00 | 00  | 00 | 00 | 00 | 5B | 02 | 00 | 00  | 10  | 00  | 00  | 00  |  |
| 2460 | 02 | 00 | 00 | 00 | 00  | 00 | 2A | 02 | 00 | 00 | 15 | 0   | 00  | 00  | 03  | 00  | 9 · · · · · · · · · · · · · · · · · · ·  |
| 2470 | 00 | 00 | 00 | 00 | 1E  | 21 | 9A | 00 | 00 | 00 | 00 | 0   | 6B  | 02  | 00  | 00  | ····!#····/@   |
| 2480 | 5B | 00 | 00 | 00 | OD  | 00 | 07 | 06 | 00 | 00 | C6 | 02  | 00  | 00  | 08  | 00  | [·····   |
| 2490 | 00 | 00 | 04 | 00 | 02  | 00 | 00 | 00 | 00 | 00 | B1 | OE  | E6  | 10  | 44  | 0E  |  |
| 2480 | 84 | FF | 80 | OE | DB  | FF | BO | OE | DB | FF | 00 | 00  | 10  | 00  | 00  | 00  |  |
| 24B0 | 00 | 00 | 00 | 00 | 00  | 00 | 00 | 00 | 00 | 00 | 00 | 00  | 41  | 70  | 65  | 6E  | Öpen   |
| 2400 | 53 | 74 | 72 | 65 | 65  | 74 | 4D | 61 | 70 | 20 | 61 | 6E  | 64  | 20  | 63  | 6E  | StreetMap and co   |
| 24D0 | 6E | 74 | 72 | 69 | 62  | 75 | 74 | 65 | 72 | 73 | 00 | 77  | 77  | 77  | 2E  | 6.F | ntributors www.o   |
| 24E0 | 70 | 65 | 6E | 73 | 74  | 72 | 65 | 65 | 74 | 6D | 61 | 70  | 2E  | F   | 72  | 67  | penstreetmap.org   |
| 24F0 | 00 | 4D | 61 | 70 | 20  | 64 | 61 | 74 | 61 | 20 | 6C | 69  | 63  | 6   | 6E  | 63  | Map data licenc  |
| 2500 | 65 | 64 | 20 | 75 | 6E  | 64 | 65 | 72 | 20 | 43 | 72 | 65  | 61  | 74  | 69  | 76  | ed under Creativ   |
| 2510 | 65 | 20 | 43 | 6F | 6D  | 6D | 6F | 6E | 73 | 20 | 41 | 74  | 74  | 72  | 6.9 | 62  | e Commons Attrib   |
| 2520 | 75 | 74 | 69 | 6F | 6E  | 20 | 53 | 68 | 61 | 72 | 65 | 41  | 6C  | 69  | 6B  | 65  | ution ShareAlike   |
| 2530 | 20 | 32 | ZE | 30 | 00  | 68 | 74 | 79 | 70 | 3A | ZF | ZF  | 6.3 | 72  | 15  | 61  | 2.0 http://crea  |
| 2540 | 74 | 69 | 76 | 65 | 63  | 6F | 6D | 6D | 6F | 6E | 73 | 2E  | 6F  | 72  | 47  | 2F  | tivecommons.org/   |
| 2550 | 6C | 69 | 63 | 65 | 6E  | 73 | 65 | 73 | 2F | 62 | 79 | 2D  | 73  | 61  | 21  | 32  | licenses/by-sa/2   |
| 2560 | 2E | 30 | 21 | 00 | 4D  | 61 | 70 | 20 | 63 | 72 | 65 | 61  | 74  | 65  | 64  | 20  | .0/ Map created  |
| 2570 | 77 | 69 | 74 | 68 | 20  | 60 | 6B | 67 | 6D | 61 | 70 | 2D  | 72  | 31  | 39  | 31  | with mkgmap-r191   |
| 2580 | 36 | 20 | 20 | 20 | 20  | 20 | żö | 00 | 50 | 72 | 6F | 67  | 72  | 61  | 6D  | 20  | 6 Program  |
| 2590 | 72 | 65 | 6C | 65 | 61  | 73 | 65 | 69 | 20 | 75 | 6E | 64  | 6.5 | 72  | 20  | 19  | released under t   |
| 2580 | 68 | 65 | 20 | 47 | 50  | 4C | 00 | 85 | OF | 01 | 00 | 04  | 10  | 01  | 00  | -0. | he GPL . I .It .I  |
| 2580 | 12 | 01 | 00 | 02 | 14  | 01 | 00 | 01 | 16 | 01 | 00 | 00  | 18  | 01  | 00  | 00  | 1  |
| 2500 | 00 | 00 | 00 | 00 | 87  | FD | D2 | 13 | 24 | 01 | 80 | 01  | 00  | 0.2 | 0.0 | 00  | \$yò!!\$ €   |
| 25D0 | 00 | 00 | CO | 00 | 82  | FD | D2 | 18 | 24 | 02 | 80 | 02  | 00  | 03  | 00  | 67  | - A-Syd!!\$161 - 0   |
| 25E0 | 00 | 00 | CO | 00 | 8A  | FD | D2 | 13 | 24 | 06 | 80 | 0.5 | 00  | 04  | 00  | 9B  | À-Šýô‼\$-€  -1 ->  |
| 25F0 | 01 | 00 | DO | 00 | 874 | FD | D2 | 13 | 24 | 15 | 80 | 14  | 00  | 05  | 00  | 12  | -D -ŠýÒ‼\$∔€¶ -  -1  |
| 2600 | 04 | 00 | DO | 00 | 8A  | FD | D2 | 13 | 24 | 54 | 80 | 4D  | 00  | 06  | 00  | C8  | - D - Šýčijstem È  |
| 2610 | 08 | 00 | DQ | 00 | 82  | FD | D2 | 13 | 24 | 4D | 81 | 34  | 01  | EB  | 14  | 00  | D -D -SyollsM4 69 -  |
| 2620 | 00 | 24 | 00 | 00 | SA  | 00 | 00 | 52 | 00 | 00 | 12 | 01  | 02  | 12  | 01  | 07  | -\$: ··R1 -1 •   |

Let's examine the bytes highlighted in blue.

The table below shows you what these numbers mean:

| code            | pois | Indexes | polylines    | polygons     | Pointers in |
|-----------------|------|---------|--------------|--------------|-------------|
|                 | -    | pois    |              |              | RGN         |
|                 |      | _       |              |              | subdivision |
| <mark>10</mark> |      |         |              |              | 0           |
| 20              |      |         |              |              | 0           |
| <mark>40</mark> |      |         |              |              | 0           |
| <mark>80</mark> |      |         |              |              | 0           |
| C0              |      |         |              |              | 1           |
| D0              |      |         | $\checkmark$ | $\checkmark$ | 2           |
| E0              |      |         | $\checkmark$ | $\checkmark$ | 2           |
| F0              |      |         |              |              | 3           |

Notice that the last subdivision contains all the elements (&FO) and so we need (4-1) 3 pointers.

Have a look at the picture below; you can see the RGN subfile starts at C00 .

Offset &15 (&C15) (4 bytes) always points to the start of the first subdivision chunk , ie it starts at 7D (C00+7D).

This number is underlined in blue.

| C00 | 7D | 00 | 47 | 41 | 52 | 4D  | 49 | 4E  | 20 | 52 | 47 | 4E | 01 | 00 | DB | 07  | } GARMIN RGN -Û.      |
|-----|----|----|----|----|----|-----|----|-----|----|----|----|----|----|----|----|-----|-----------------------|
| C10 | 80 | 14 | OA | 38 | 24 | 7D  | 00 | 00  | 00 | 34 | 06 | 00 | 00 | B1 | 06 | 00  | 98\$) ···· 4-·· ±-·   |
| C20 | 00 | 00 | 00 | 00 | 00 | 00  | 00 | 00  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00  |                       |
| C30 | 00 | 00 | 00 | 00 | 00 | 00  | 00 | 00  | 00 | B1 | 06 | 00 | 00 | 00 | 00 | 00  | ••••• <u>+</u> =••••• |
| C40 | 00 | 00 | 00 | 00 | 00 | 00  | 00 | 00  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00  |                       |
| C50 | 00 | 00 | 00 | 00 | 00 | B1  | 06 | 00  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00  | ·····±                |
| C60 | 00 | 00 | 00 | 00 | 00 | 00  | 00 | 00  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00  |                       |
| C70 | 00 | 00 | 00 | 00 | 00 | 00  | 00 | 00  | 00 | 00 | 00 | 00 | 00 | 28 | 00 | 35  |                       |
| C80 | 01 | 64 | 00 | 00 | 80 | 5E  | FF | BD. | FF | 17 | 64 | CF | 00 | 80 | 7A | 00  | d€^949 dI .€z .       |
| C90 | 94 | FF | OF | 12 | 00 | 0.0 | 80 | FE  | FF | DE | FF | 0B | 64 | 00 | 00 | 80  | "9#1 · ·€þ9Þ92d · ·€  |
| CAO | 92 | FF | 2A | 00 | 17 | 2C  | 00 | 00  | 00 | EC | FE | C1 | FF | 0C | 24 | F9  | '9* - , ipág?\$ù      |
| CBO | A4 | 3D | 09 | DO | F1 | D1  | SF | F9  | CO | 87 | 05 | 16 | 00 | 00 | 00 | 01  | H=ĐNN ùÀ‡ T           |
| 000 | 00 | -  | -  | 00 |    |     | 00 | 00  |    |    | 00 | -  | -  | -  | -  | 0.0 | 10 441 m 32-00        |

Now look at C7D : underlined in green are 2 pointers , 28 00 and 35 01 – each pointer is always 2 bytes long .

Next, underlined in yellow we find some pois, ie the first one is 64 00 00 80 5E FF BD FF 17 – more later.

Question: How do I know there are only two (green) pointers? Why is 64 00 not a pointer?

We know, because in TRE where the pointers are kept, we found this subdivision to contain &D0 types of elements, giving us 3 elements and hence 2 pointers.

### Start of Subdivisions in RGN + &29

The first 3 bytes show pointers to a subdivision in the RGN file. Remember to add the offset found at RGN + &15 to each of these values.

| 25B0 | 12 | 01  | 00 | 02 | 14 | 01 | 00 | 01 | 16 | 01 | 00 | 00 | 18 | 01 | 00 | 00 | \$ |
|------|----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2500 | 00 | 00  | 00 | 00 | 8A | FD | D2 | 13 | 24 | 01 | 80 | 01 | 00 | 02 | 00 | 00 | 1  |
| 25D0 | 00 | 00  | CO | 00 | 8A | FD | D2 | 13 | 24 | 02 | 80 | 02 | 00 | 03 | 00 | 67 |    |
| 25E0 | 00 | 00  | CO | 00 | 8A | FD | D2 | 13 | 24 | 06 | 80 | 05 | 00 | 04 | 00 | 9B |    |
| 25F0 | 01 | 0.0 | DO | 00 | 8A | FD | D2 | 13 | 24 | 15 | 80 | 14 | 00 | 05 | 00 | 12 |    |
| 2600 | 04 | 00  | DO | 00 | 8A | FD | D2 | 13 | 24 | 54 | 80 | 4D | 00 | 06 | 00 | C8 | ٦  |
| 2610 | 08 | 00  | DO | 00 | 8A | FD | D2 | 13 | 24 | 4D | 81 | 34 | 01 | EB | 14 | 00 | C  |
| 2620 | 00 | 24  | 00 | 00 | 3A | 00 | 00 | 52 | 00 | 00 | 12 | 01 | 02 | 12 | 01 | 07 | 0  |

| subdivision | Offset           | Type of<br>Data | Number of<br>Pointers in | Zoom level | Real map<br>Levels |
|-------------|------------------|-----------------|--------------------------|------------|--------------------|
|             |                  |                 | zoom chunk               |            |                    |
| 0           | 00 00 00 or 0    | 0               | 0                        | A1         | 5                  |
| 1           | 00 00 00 or 0    | C0              | (2-1) =1                 | B1         | 4                  |
| 2           | 67 00 00 or &67  | C0              | (2-1)=1                  | C1         | 3                  |
| 3           | 9b 01 00 or &19b | D0              | (3 – 1)=2                | D1         | 2                  |
| 4           | 12 04 00 or &412 | D0              | (3 – 1)=2                | E1         | 1                  |
| 5           | C8 08 00 or &8C8 | D0              | (3-1)=2                  | E2         |                    |
| 6           | etc              | etc             | etc                      | F1         | 0                  |
| 7           | etc              |                 |                          | F2         |                    |

In our example, we should find a new subdivision at offset 67, ie RGN+ &7D + &67The next new subdivision is at RGN+ &7d + &19B etc

We have named the levels A - F for the sake of clarity.

We can see that zoom level 'E' has 2 pointers, and zoom level F has 5 pointers , remember the above found sequence: 1,1,1,1,2,5

Important: The subdivisions in all mapevels apart from the last one have 16 bytes; all subdivisions in the last level, in our case 'F', have only 14 bytes.

To obtain the address for each subdivision we add the offset to (RGN + pointer found in (RGN+&15)).

Example: in our case subdivision 5 has an offset of &8C8 . Supposing RGN starts at &E00 and the offset pointer found at E00+&15 points to &7D This subdivision, therefore, is found at :&8C8 + &E00 + &7D

Note: the size of a subdivision appears to be a critical; perhaps that is why IMGs created using cgpsmapper seem to include a subdivision for almost every highway!

#### Grouping subdivisions into map levels

We mentioned earlier that maplevels are groups of subdivisions. If only we could find which subgroups belonged to which maplevel . Fortunately we can, but NOT in the RGN subfile.

The information is found in TRE offset 0 x 21.

It contains information about maplevels, not where they are, but how many there are and also how many subdivisions each map level has and also its resolution( bits\_per\_coord).

It doesn't actually give you pointers to an address in the RGN but the 3rd column (green) indicates the number of subdivisions in each maplevel.

So we can see that map level 6 starts at subdivision 7

| 5,F, <mark>1</mark> ,0,  |
|--------------------------|
| 4,10 <mark>,1</mark> ,0, |
| 3,12, <mark>1</mark> ,0, |
| 2,14, <mark>1</mark> ,0, |
| 1,16, <mark>2</mark> ,0, |
| 0,18, <mark>3</mark> ,0, |
| _                        |

| maplevel | subdivision |
|----------|-------------|
| 1        | 1           |
| 2        | 2           |
| 3        | 3           |
| 4        | 4           |
| 5        | 5           |
|          | 6           |
| 6        | 7           |
|          | 8           |
|          | 9           |

| Sub      | division.001     | 119F    | Bits/Cod        | nd.22 F   | 7 single |         | F PO   | la la       | Polyle    | xes .    | P P        | Polygons      |  |  |
|----------|------------------|---------|-----------------|-----------|----------|---------|--------|-------------|-----------|----------|------------|---------------|--|--|
| Mapi.    | Address          | Eim     | Longitude       | Latitude  | Width    | Address | Longi  | ude Latitu  | se Tvo    | e LBL    | Name       |               |  |  |
| 0        | 000000           | 00      | -3.450801       | 50.711281 |          | 2422    | 1451   | 2514 50.71  | 16453 120 | 2 0.0.00 | 1          |               |  |  |
| 1        | 000000           | 40      | -3.450801       | 50.711302 |          | 2428    | 3.453  | 008 50.71   | 88235 2F1 | 7 40.00  |            | -             |  |  |
| 2        | 0000D4           | CO      | -3.450801       | 50.711302 |          | 2434    | -3.463 | 3113 50.71  | 62801 400 | 0 8040   |            |               |  |  |
| 3        | 000258           | FO      | -3.450801       | 50.711302 |          | 2430    | 3.457  | M029 50 72  | 05717 261 | 7 0000   | 10         |               |  |  |
| 4        | 000883           | FO      | -3.457625       | 50,710616 |          | 2445    | 3.473  | 5252 5571   | 59368 120 | 2 15.0   |            |               |  |  |
| 4        | 00119F           | FO      | -3.459964       | 50,711302 |          | 2445    | 3.457  | 7964 55 71  | 31044 120 | 2 196    |            |               |  |  |
| 4        | 002128           | DO      | -3.441617       | 50.711281 |          | 2457    | 3.483  | 8247 50.71  | 10888 120 | 2 29.0   |            |               |  |  |
| 5        | 002944           | FÖ      | -3.457625       | 50.710616 |          | 2460    | 3.454  | 5998 5071   | 14477 120 | 2 20.0   |            |               |  |  |
| 5        | 00361B           | FO      | -3.459964       | 50,711302 |          | 7400    | 3 457  | 18604 50.77 | 07433 640 | E 31.0   |            |               |  |  |
| 5        | 005115           | DO      | -3.441617       | 50.711281 |          | 1       | 11     | 0004 00.12  | 01400     |          |            |               |  |  |
|          |                  |         |                 |           |          | Addr    | Type   | Longitude   | Lattude   | LBL      | Name       | bitstream 4   |  |  |
|          |                  |         |                 |           |          | 26AD    | 0E     | -3.4655429  | 50,717654 | 00 00    | describe - | 2 10 D1 3     |  |  |
|          |                  |         |                 |           |          | 2689    | 10     | -3.4662296  | 50.720915 | 150_     |            | E 43 C 0 43 5 |  |  |
|          |                  |         |                 |           |          | 26D1    | 16     | -3.4683754  | 50.715508 | 2A.0     |            | 1055          |  |  |
|          |                  |         |                 |           |          | 260C    | 05     | -3.4727527  | 50.720057 | 3F 0     |            | 2 20 ED 1     |  |  |
|          |                  |         |                 |           |          | 2658    | 45     | -3.4669162  | 50 714907 | 54.0     |            | 21450         |  |  |
|          |                  |         |                 |           |          | 26F4    | 14     | -3.445232   | 50 714735 | A110     |            | 2 51 5D 16    |  |  |
|          |                  |         |                 |           |          | 2700    | 05     | -3.4669162  | 50,714907 | 60.0     |            | 3 14 F7 E8 23 |  |  |
|          |                  |         |                 |           |          | 2700    | 06     | -3.468032   | 50,714650 | 86.0     |            | 4 12 4 59 FE  |  |  |
|          |                  |         |                 |           |          | 2718    | 06     | -3.4670579  | 50.714478 | 98.0     |            | 4 13 AA C5 4  |  |  |
|          |                  |         |                 |           |          | 2729    | 10     | -3 4658004  | 50 716881 | B0.0     |            | 13 42 4 80 57 |  |  |
| < 100 F  | =                |         |                 |           | 1        | 2746    | 29     | -3.4615947  | 50 722460 | 000      |            | 2 23 30 D     |  |  |
| 1 COUN   | VTRYABC          |         |                 |           |          | 2752    | 10     | -3.4678604  | 50,716967 | C5 0     |            | 1017          |  |  |
| A CLYS   | T ST MARY        |         |                 |           | 131      | 2750    | 29     | -3.4615947  | 50,722460 | 000      |            | 2 33 AD 15    |  |  |
| 14 SOW   | TON INDUST       | TRIAL E | STATE           |           |          | 2769    | 06     | -3.4728386  | 50,720744 | DA 0     |            | 2 20 AD 1     |  |  |
| 27 STAR  | FF PARKING       | in all  |                 |           | +        | 2775    | 07     | -3.4642555  | 50 718512 | EF 0     |            | 4 12 E8 26 4F |  |  |
| 31 CLY   | ST ST MARY       | POST    | OFFICE          |           |          | 1       |        |             |           | S Note-  |            |               |  |  |
| 44 EX5   | 1BR<br>MALTSTERI | s       |                 |           |          | Addr    | Туре   | Longtude    | Latitude  | LBL      | Name       | bitstream -   |  |  |
| 54 EX2   | 595              |         |                 |           |          | 3180    | 5      | -3.465285   | 50,717053 | 000      |            | E 32 30 0 F0  |  |  |
| SA COS   | TA COFFEE        |         |                 |           |          | 3198    | 4F     | -3.462796   | 50.713278 | 000      |            | 7 42 F4 C0 E/ |  |  |
| 64 OVE   | RELOW CAR        | PARK    |                 |           |          | 31A9    | 2      | -3.452067   | 50,722460 | 000      |            | 5 23 D8 C6 2  |  |  |
| 71 TRA   | VELODGE          |         |                 |           |          | 3188    | 5      | -3.464513   | 50.717654 | 1840     |            | E 23 14 A0 F  |  |  |
| 79 EX2   | 130              |         |                 |           |          | 31D0    | 24     | -3.452754   | 50.721859 | 000      |            | 9 23 D8 44 30 |  |  |
| TF TEL   | EPHONE           |         |                 |           |          | 31E3    | 12     | -3.468976   | 50,714049 | 000      |            | C 34 C8 E7 3  |  |  |
| BE MAR   | KS AND SPE       | NCER    | 5               |           |          | 31F9    | 8      | -3.468461   | 50.715079 | 000      |            | 9 34 EC EB 4  |  |  |
| 94 EXE   | TER              |         | S               |           |          | 320C    | 50     | -3.463998   | 50.712161 | 000      |            | 1E 32 74 BF 1 |  |  |
| 99 FY2   | THE              |         |                 |           |          | 3234    | 2      | -3.473783   | 50.710186 | 000      |            | 8 32 80 6D DI |  |  |
| SE EVE   | TER SERVIC       | ES      |                 |           |          | 3246    | 17     | -3.466058   | 50.712590 | 000      |            | 8 13 18 2 88  |  |  |
| AR THE   | CAT AND F        | DOLE    |                 |           |          | 3258    | 26     | -3.462539   | 50.717654 | 1120     |            | 19 43 FC EF [ |  |  |
| RO THE   | RIUE RALL        |         |                 |           |          | 3278    | 5      | -3.464599   | 50.718340 | 000      |            | 8 24 20 0 20  |  |  |
| C3 THE   | HALF MOOT        |         |                 |           |          | 3280    | C      | -3.466744   | 50.714907 | 1400     |            | 1A 56 74 37 i |  |  |
| 00.000   | EV AND SOM       | TON E   | INT OFFAT WE    | STERN     |          | 3281    | 5      | -3.463054   | 50.716881 | 2E 4 0   |            | 7 13 D4 3 88  |  |  |
| 100 0100 | ar who dow       | and the | THE STORE STORE | arrante.  |          | 2202    | ~      | 0 170100    | CA 745705 | 0100     |            | 110100.00     |  |  |

Below, an IMG file with 10 subdivisions and maplevels 0 to 5. Maplevels 4 & 5 both have 3 subdivisions. Maplevel 0 contains no elements, Maplevel 1 only polylines (40) etc.

Highlighted are the contents of subdivision 119F at maplevel 4.

Notice F0 meaning pois, polylines and polygons. The creamy list contains all the pois and indexed pois, blue polylines and purple polygons, plotted below including nodes.



### 'the RGN block and beyond'

So, now we know how many map levels there are, but we still don't know how long the last map level block in the RGN is.

Fortunately, Garmin gives us the size of the RGN block, ie from the first subdivision containing pointers and pol/highway/polygon data, to the end of the last one.

This is found at RGN + &2D

Now, interestingly there is some special data BEYOND this block not documented by Mechalas.

So, in addition to the subdivisions mentioned above, the RGN contains other chunks of data found at pointers beyond &15.

| Offset in RGN |                  | bytes |
|---------------|------------------|-------|
| 1d            | pointer          | 4     |
| 21            | Length of block  | 4     |
| 39            | Pointer to block | 4     |
| 3d            | Length of block  | 4     |
| 55            | Pointer to block | 4     |

Some imgs store highways with types &100+ in the 'green' block . These types are used to perform overlays ; this in theory ensures that certain highways, ie bridges, when given types &100+, acquire the highest draw order and can overlap say Motorways. This, for some reason does not always happen. More later. – see Polylines

Orange block is reserved for polygons with types &100+ - see Polygons.

## Locked TOPO maps

You may find that these locked maps do not display the word 'Garmin' in a hex editor. If so, data needs to be unscrambled. Find the first hex number and use the XOR function, 'xorring' every byte by the first byte found. This will, ofcourse, still keep the file locked, but at least it's readable.

Locked TOPO maps have the subdivisions pointers in the TRE encrypted. (For those in the know, bytes &24 - &27 are used to unscramble the zoom level chunk and replaced with new values)

This means you need to guess where each zoom level chunk starts and ends in the RGN. This RGN data is itself not encrypted, so you can still read the first RGN map level chunk, including its pointers, however you have no idea where it ends.

The problem is that pointers could be read as POIs etc and pois can be mistaken for polylines. However, you could in theory when parsing , build in, best fit scenarios to check if the next piece of data is a pointer, a POI or a polyline given the fact that coordinates couldn't suddenly escape the given boundaries.

If an IMG file contains pois, polylines or polygons with extended types, ie 0x100+ (see below), then these can be read and plotted without any problems. Even pointers to different subdivisions remain unscrambled – see TRE7.

## LBL

Where are all the names of streets, pois etc kept?

They are stored in the LBL subfile. However when you examine it, you more than likely won't recognize any labels ,as they are all encrypted.

Lets examine a POI data stream found in a RGN zoom level

#### <mark>46</mark> DF 00 00</mark> 49 FF BB FF <mark>02</mark>

The first and last 'green' byte denote type and subtype, so this POI's type is 4602

The next three yellow bytes DF 00 00 are and offset in LBL, ore more precisely in LBL1 which is an offset from the start of the LBL subfile.

However, bits 7 & 8 in the 'third byte' are reserved for something else:

Sometimes the third byte is 80, which implies that it is also found in the NET subfile if it's a highway. More importantly, if the third byte is a 80,40 C0 ,82,C2 etc then the pointer **DOES NOT** point to an offset from the beginning of the lbl1 labels/names block. Instead, it points to a **different** block where other information is found, including a pointer to the label lbl1 block – more later

For our purpose, we could just look at the first two bytes (you also need to check if bits 1 to 6 are set in the third byte - only used if labels are found a long way from the beginning of this block).

It tells me that the name of the poi is found at offset 00DF (ie &DF). The other 4 bytes following the 'yellow block' provide the x and y coordinates.

However, it is not an offset starting from the beginning of the LBL subfile.

Instead, it is an offset from where the label block within the LBL starts ; the start of this label block is defined in LBL + &15 or LBL1.

Each name in the label block ends with 00 if it's not encoded.

Let's have a look at an example:

| 1400 | C4 | 00 | 47 | 41 | 52 | 4D | 49 | 4E | 20 | 4C | 42 | 4C | 01 | 00 | DB | 07 | A GARMIN LBL -0.                             |
|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
| 1410 | 07 | 14 | 0C | 27 | 11 | D1 | 00 | 00 | 00 | 27 | 00 | 00 | 00 | 00 | 06 | F8 | • 9 ? · 48 · · · · · · · · · · · · · · · · · |
| 1420 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 00 | 00 | FB | 00 | 00 | L  |
| 1430 | 00 | 00 | 00 | 00 | 00 | 05 | 00 | 00 | 00 | 00 | 00 | FB | 00 | 00 | 00 | 00 | ·····  |
| 1440 | 00 | 00 | 00 | 05 | 50 | 00 | 00 | 00 | 00 | FB | 00 | 00 | 00 | 00 | 00 | 00 | ····   ····· à ······                        |
| 1450 | 00 | 04 | 00 | 00 | 00 | 00 | 00 | FB | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .J <u>6</u>                                  |
| 1460 | 00 | 00 | 00 | 00 | FB | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | ···· b····· b····                            |
| 1470 | 00 | 00 | FB | 85 | 00 | 00 | 00 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 00 | 00 | ··@······                                    |
| 1480 | FB | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 06 | 00 | 00 | 00 | 00 | 00 | FB | 00 | Q · · · · · · · · · · · · · · · · · · ·      |
| 1490 | 00 | 00 | 00 | 00 | 00 | 00 | 05 | 00 | 00 | 00 | 00 | 00 | FB | 00 | 00 | 00 | ·····  |
| 14A0 | 00 | 00 | 10 | 00 | 03 | 00 | 00 | 00 | 00 | 00 | E4 | 04 | 00 | 00 | 00 | 80 | L  |
| 1480 | C4 | 00 | 00 | 00 | OD | 00 | 00 | 00 | FB | 00 | 00 | 00 | 00 | 00 | 00 | 00 | Ä · · · · · û · · · · · ·                    |
| 14C0 | 00 | 00 | 00 | 00 | 44 | 65 | 66 | 61 | 75 | 6C | 74 | 20 | 73 | 6F | 72 | 74 | ····Default sort                             |
| 14D0 | 00 | 00 | 00 | F5 | 4E | 51 | 26 | 5D | 04 | 20 | FF | 41 | 23 | C7 | 48 | 13 | ··· \$5NQ4] YA‡ÇH!!                          |
| 14E0 | 40 | 30 | 90 | C5 | 38 | 31 | 44 | 01 | 53 | 84 | 15 | 20 | 07 | 40 | CO | 16 | 00Å81D S. + .0ÅT                             |
| 14F0 | 88 | 41 | 23 | C7 | 48 | 13 | 10 | 03 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | <a#çh!!81< td=""></a#çh!!81<>                |
| 1500 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |  |

Exploring IMG Format

We know all label data starts at offset &15 from the beginning of the LBL subfile. Strangely, it points to 00 ( LBL + &D1). Add 1 and you get the sequence OC F5 4E 51

When you cross-check in the right column, below Default sort, it just shows gobbledy gook. Let me tell you that all the values underlined in green read :'COUNTRY' It looks as if the word is zipped in some way and most likely 6 bit encoding has been used.

First you need to change each of the 4 bytes into a binary format of zeros and ones . We don't know if it's just 4 bytes, it could be more - see later.

In basic this would be:

```
For n = 7 to 0 step -1
If (byte and 2^n) = 2 ^n then string=string+"1" else
string=string+"0"
Next
```

Do this for each byte until you get a string which begins like this :

#### 000011001111010101001110.....

Now split them into chunks of 6, as this is how most of the IMGs are encoded.

#### 000011 001111 010101 001110 .....

#### Chunk 1:00 0011

Read from *right* to left and you get:

 $1\ 1\ 0\ 0\ 0$  or  $2^0 + 2^1 + 0 + 0 = 3$  which is the third letter ie 'C'

### Chunk 2:00 1111

Read from *right* to left and you get:

1 1 1 0 0 0 or  $2^{0} + 2^{1} + 2^{2} + 2^{3} + 0 + 0 = 15$  which is the 15th letter, ie 'O'

(The reason why we had to add 1 was because in our example the offset (+ 1) was given in the Country records offset at lb + 1F- see further)

How do we know we've come to the end of a word? Garmin have though of this: if the 6 bit number is > &2f then you've reached the end of your label. The next label starts in the next byte, so any superfluous bits are just discarded and not used as the beginning of the next word.

## Lbl pointers NOT directly pointing to lbl1 label block

Pointers are always 3 bytes, but sometimes the third byte ends in 80,82,C0,C6 etc

If bit 7 of the third byte is set then your lbl label refers to an offset in NET1 where you can find a 3 byte pointer to LBL1. Remember : Pointer =  $1^{st}$  byte +  $2^{nd}$  byte +  $(3^{rd}$  byte ) mod 32 Net1 starts at offset 0x15 from the beginning of the NET subfile.

#### Example: 60 00 C1

60 00 01 (&C1 mod 32) points to offset &010060 from the NET1 block. Here we will find a pointer to the main label block. Bit 7 of this byte may also be set! So, pointer NET1 Byte1+ NET1 byte + (NET1 byte 3) mod 32 should point to somewhere in LBL1, even if it is a blank label.

| LBL Offset |                | size |
|------------|----------------|------|
| 1F         | Country        |      |
| 2D         | Region         |      |
| 3B         | City           |      |
| 49         |                |      |
| 57         | POI            |      |
| 64         |                |      |
| 72         | ZIP Post Codes |      |
| 80         | Highway        |      |
| 8E         | Exit           |      |
| 9C         | Highway data   |      |

There are several offsets/pointers found in the LBL header :

|       | 1400 | C4 | 00  | 47 | 41 | 52 | 4D | 49 | 4E | 20 | 4C    | 42 | 4C | 01 | 00 | DB | 07 | A GA    |
|-------|------|----|-----|----|----|----|----|----|----|----|-------|----|----|----|----|----|----|---------|
|       | 1410 | 07 | 17  | 10 | 03 | 13 | D1 | 00 | 00 | 00 | 87    | 00 | 00 | 00 | 00 | 06 | 58 | •1+L    |
|       | 1420 | 01 | 00  | 00 | 03 | 00 | 00 | 00 | 03 | 00 | 00    | 00 | 00 | 00 | 5B | 01 | 00 |         |
|       | 1430 | 00 | 00  | 00 | 00 | 00 | 05 | 00 | 00 | 00 | 00    | 00 | 5B | 01 | 00 | 00 | 05 |         |
|       | 1440 | 00 | 20  | 00 | 05 | 00 | 00 | 00 | 00 | 00 | 60    | 01 | 00 | 00 | 00 | 00 | 00 |         |
|       | 1450 | 00 | 04  | 00 | 00 | 00 | 00 | 00 | 60 | 01 | 00    | 00 | 08 | 00 | 00 | 00 | 00 |         |
|       | 1460 | OE | 00  | 20 | 00 | 68 | 01 | 00 | 00 | 00 | 00    | 00 | 00 | 04 | 00 | 00 | 00 | 11 1    |
|       | 1470 | 00 | 00  | 6  | 01 | 00 | 00 | 03 | 00 | 00 | 00    | 03 | 00 | 00 | 00 | 00 | 00 | · ·h    |
| ×     | 1480 | 6B | 01  | 00 | 00 | 00 | 00 | 00 | 00 | 06 | 00    | 00 | 00 | 00 | 00 | 6B | 01 | k ·     |
| 1     | 1490 | 00 | 00  | 00 | 10 | 00 | 00 | 05 | 00 | 00 | 00    | 00 | 00 | 6B | 01 | 00 | 00 |         |
|       | 14A0 | 00 | 00  | 00 | 0  | 03 | 00 | 00 | 00 | 00 | 00    | E4 | 04 | 00 | 00 | 00 | 80 |         |
|       | 14B0 | C4 | 00  | 00 | 00 | OD | 00 | 00 | 00 | 6B | 01    | 00 | 00 | 00 | 00 | 00 | 00 | Ä · · · |
|       | 1400 | 00 | 00  | 00 | 00 | 14 | 65 | 66 | 61 | 75 | 6C    | 74 | 20 | 73 | 6F | 72 | 74 |         |
| -1    | 14D0 | 00 | 00  | oc | FS | 41 | 51 | 26 | 5D | 04 | 20    | FF | 38 | 55 | CO | 24 | E3 | · · ¥ō  |
|       | 14E0 | BF | 09  | 23 | C1 | 10 | 33 | 19 | 4D | 4F | 15    | 89 | 40 | 8C | 26 | 3F | 5C | 2#A+3   |
|       | 14F0 | 82 | 4D  | 40 | C1 | 40 | 18 | FO | 44 | FC | 41    | 23 | C7 | 48 | 13 | 40 | 30 | , MQÁ   |
|       | 1500 | 90 | C5  | 38 | 31 | 44 | 01 | 53 | 84 | 15 | 20    | 07 | 40 | CO | 16 | 8B | 3D | Å81D    |
|       | 1510 | 01 | 4E  | 4D | 44 | 85 | 15 | 43 | 41 | 41 | C3    | 8F | 48 | 70 | 03 | 3C | E5 | NM      |
|       | 1520 | 12 | 24  | 25 | 54 | 3D | 24 | 17 | 4C | 51 | 5C    | 68 | 02 | 14 | 51 | 07 | 1A | 1\$%T   |
|       | 1530 | 70 | F7  | OF | SC | 92 | C9 | 70 | E3 | DO | 14    | E4 | D4 | 48 | 51 | 54 | 34 | p÷#\    |
|       | 1540 | 14 | 10  | 38 | F4 | 87 | 70 | F2 | 7. | 10 | 56    | 10 | :9 | 02 | 10 | 70 | FO | \$80\$  |
|       | 1550 | 54 | 51  | 22 | 42 | 55 | 42 | 4F | 3B | 01 | 00    | 00 | 10 | 00 | 00 | 01 | 40 | TQ"B    |
|       | 1560 | OA | 00  | 00 | 1E | 00 | 00 | 01 | 01 | 18 | 00    | 00 | 00 | 00 | 00 | 00 | 00 |         |
|       | 1570 | 00 | 00  | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00    | 00 | 00 | 00 | 00 | 00 | 00 |         |
| i i i |      |    | 1.1 |    |    |    |    |    |    |    | 1.0.0 |    |    |    |    |    |    |         |

Look at LBL + 1F, underlined in blue: it reads 58 01 - so offset is &158 from LBL, ie &1558

There you will find underlined in green a 3 byte pointer : 01 00 00, ie 1 and this is our 'elusive' offset.

There are only 3 bytes to look at (see green 03 at &1422). If there were 2 more regions then the length (03) will be multiples of 3, ie 06.

Next, there appears to be some more pointers at 155B, underlined in yellow.

This is a yellow 5 byte chunk, 10 00 00 01 40

The first 3 bytes point to the label data segment (ie 10), the next 2 bytes are to do with 'city information'

## Symbols

If you parse the LBL1 from the label offset you should get all the labels. When labels are encoded in 6 bits, the start of the next label is always the next byte.

You may get funny letters but check Mechalas' section on symbol letters– it's fairly straight forward except that 0x1B should be 0x1C.

## More than 1 label in NET

A highway can have up to four label pointers in NET, ie

 $6C\ 00\ 00\ 72\ 00\ 00\ 91\ 00\ 00\ 12\ 01\ 80$ 

The last label is terminated with bit 7 being set , ie 80 in our case. This shows 4 labels, one starting at  $62\ 00\ 00$  and the last at  $12\ 01\ 80$ .

This principle of the 7<sup>th</sup> bit when set terminating a list is used in other subfiles as well, ie NOD.

For more information regarding other highway properties see NET.

## POIs

Points of interest all have a type number and a subtype number, ie type = 30 and subtype=01. The subtypes are supposed to be subsets of the main type: ie main type = amenity restaurant and subtype = French cuisine

The length of each block varies depending on whether the subtype = 0 or not.

### POIs with subtypes

| type | Lbl I | Lbl II | Lbl III | longitude | latitude | subtype |
|------|-------|--------|---------|-----------|----------|---------|
|      |       |        |         |           |          |         |

#### POIs with no Subtypes

| type | lbl | lbl | lbl | longitude | latitude |  |
|------|-----|-----|-----|-----------|----------|--|
|      |     |     |     |           |          |  |

How do we know if the poi has a subtype or not?

Answer: if (lbIIII and 128)=128 then it contains a subtype.

This is where Mechalas is incorrect: it is not the bit 8 of first byte, but bit 8 of the 4<sup>th</sup> byte.

#### **POI** labels

If bit 7 in the above LBLIII block is set then the labels for all pois are kept in a block named LBL6 ie found at LBL + 4 byte pointer at (LBL + &57)

This marks the beginning of series of 3+ byte chunks, the first 3 bytes of each chunk shows pointers to LBL1 label blocks. The other bytes indicate presence of phone-numbers etc.

Generally, we find 3 byte chunks showing pointers to LBL 1

Example :

POI: 46 39 00 C0 B5 FD 53 FE 11

Exploring IMG Format

Bit 8 of the lbIII pointer is set, ie &C0 : this means it has a subtype : Bit 7 is also set ; this means 00 39 is pointing to LBL 6. ( &C0 = &40 + &80)

Now start from the beginning of LBL 6 ,underlined in blue and add &39 . It points to 3F 00 00

| 40 | 00 | 00 | 10 | 01 | 00 | 00   | 01 |    | 00 | 01 | 00 | 01 | 00 | 00 | 01 |   |
|----|----|----|----|----|----|------|----|----|----|----|----|----|----|----|----|---|
| CO | 00 | 00 | 00 | 00 | 00 | 00   | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | À |
| 18 | 00 | 00 |    | 00 | 00 | 00   | 00 | 00 | 00 | 00 | 00 | 1E | 00 | 00 | 28 | 1 |
| 00 | 00 | 00 | 00 | 00 | 00 | - 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 2F | 00 |   |
| 00 | 00 | 00 | 00 | 00 | 00 | 00   | 36 | 00 | -  | 3F | 00 | 00 | 3F | 00 | 00 |   |
| 47 | 00 | 00 | 00 | 00 | 00 | 00   | 00 | 00 | 00 | 00 | 00 | 50 | 00 | 00 | 52 | G |

This, finally, is our lb1 pointer and hey presto we get the name of an amenity at offset 3F from LBL1!

## POIs with extended types

These are special POIs which may not show up on your GPS or Mapsource/Basecamp.

They could have types of 0x101,  $0 \times 10101$  etc and are found at RGN4 + Offset &55 with length of block at RGN4 +&68 containing additional information. The length of each block is variable depending on bits 5 - 7 of the 2<sup>nd</sup> byte.

In its simplest form its structure is:

| Type + | Sub type |           |           | latitude | latitude | lbl | lbl | lbl |
|--------|----------|-----------|-----------|----------|----------|-----|-----|-----|
| &100   | mod 32   | longitude | longitude |          |          |     |     |     |

Example : 14 35 45 00 23 01 06 2E 00

This is a poi with type: &100 + &14 + &35 mod 32 = &11415

It has a label (yellow) at offset &2e06 from LBL1:. We know it has a label as bit 5 was set , ie &20 + &15

Another example encountered has bit seven set:

Example

02 A1 90 FF 9F FF CD 00 00 E0 09 00 00 00 00 01

The bit stream following the yellow lbl label is at present unclear. It is possible that E0 marks beginning of a possible bitmap stream/index with 09 being a flag and next 3/4 bytes a pointer to its location. (The MDR has similar 2 byte codes, telling us how many bytes we should expect next, ie number of characters and number of bytes to denote the index.)

### For more information see TRE7

## POLYLINES

Polylines ,like polygons, have a more complicated variable length.

The first 9 bytes are fixed :

| 0    | 1    | 2     | 3      | 4         | 5         | 6        | 7        | 8      |
|------|------|-------|--------|-----------|-----------|----------|----------|--------|
| type | lblI | lblII | lblIII | longitude | longitude | latitude | latitude | length |

Because the highway node information can exceed 256 bytes we need to know when the length is > 256 and when it is not.

Answer: if the (type byte and 128 = 128) then we know the length is determined by 2 bytes.

The first 8 bytes gives us the starting point coordinates of the highway – however, it's as an offset from the centre of a box defined by each subdivision in TRE .

## **Polyline Labels**

If an IMG is not routable then the 3 LBL byte chunks behave normally and show pointers to lbl 1

If bit 8 in the above LBLIII block is set then these highways are routable. In which case the labels of these highways are kept in a different block named NET1 ie found at NET + 4 byte pointer at (NET + &15)

Railway lines, (type 0 x 14) are not routable (in theory) so lblIII should not have bit 8 set.

## Polylines with extended types 0 x 100+

These are located in RGN3 offset &39 and are not routable; length of this block is found in RGN offset &3d.

Their structure is quite different from polylines with types  $< 0 \times \&100$ .

| 0      | 1        | 2         | 3         | 4        | 5        | 6       | 7+     | 3 bytes |
|--------|----------|-----------|-----------|----------|----------|---------|--------|---------|
| Type + | Sub type | longitude | longitude | latitude | latitude | pointer | length | lbl     |
| &100   | mod 32   |           |           |          |          |         | +data  |         |

Example : first 2 bytes are : 12 24

Add &100 to 12 and 24 mod 32 = 4: this makes it type 11204 If the second byte, ie 24, has bit 5 set (ie &20) then the polyline has text; in which case, the 3 byte LBL pointer is stuck at the end of its data stream.

No text :

003F8590: 0F 04 23 FF 10 00 11 30 A6 10 12 85 F0 02 01

With text:

003F8590: 0F 24 23 FF 10 00 **11** 30 A6 10 12 85 F0 02 01 003F85A0: **38** 70 02

## Length of a polyline type 100+ block

Byte 7+ defines the length of data to draw the polyline.

In the example before length is highlighted in green : ie &11 or decimal 17

If byte 7 is even then an extra byte will be used to calculate its length.

In our case it's odd so only one byte is needed.

| Single Byte Algorithm: | (byte 7 -1) / 2            |
|------------------------|----------------------------|
| Two Bytes Algorithm:   | (byte7 + byte 8 * 256) / 4 |

Note: the length refers to the data steam to define the complete polyline, so it remains the same with or without LBL pointers. For more information see TRE7.

## POLYGONS

Data structure is the same as for polylines.

## Polygon Labels

Again see polylines

### Polygons with extended types 0 x 100+

These are located in RGN2 offset &1D ; length of this block is found in RGN offset &21.

## Length of a polygon type 100+ block

See Polylines.

## **Plotting Coordinates**

Finally we come to plotting our elements.

The coordinates are stored in 2 byte offsets from the centre of your current zoom level map

| Earlier on we gave the following example                 | 85 F 1 0  |
|--|-----------|
| for map zoom levels found in TRE where                   | 100,1,1,0 |
| the last two digits determine the number of subdivisions | +,10,1,0  |
| in each map level.                                       | 5,12,1,0  |
|  | 2,14,1,0  |
|  | 1,16,2,0  |

This tells us that we have 5 different map levels. Remember, at each level elements may be plotted. The first level you encounter tends to be empty, not referring g to any elements and is used for determining boundaries.

0,18,5,0,

Look at the last level : 0, 18, 5,0

The second number (&18) is vital to the way elements are plotted. It is referred to as the bits per coordinate (bpc)

The formula for getting latitude and longitude degrees : 1 garmin unit =  $360/(2^24)$ 

24 (or &h18) represents the bpc

At each zoom level you swap 24 for the second number in our table . Remember, this will be different for each IMG.

In our case the various bpcs are : &F, &10,&12,&14,&16,&18, increasing the accuracy of the coordinates.

### **Plotting POIs**

#### 64 15 03 C0 1F 00 1C 00 0F

You need to add these two byte chucks in green to the 3 byte latitude + longitude centres defined in TRE +&29 – see earlier. Add the first two to the latitude, and the second 2 to the longitude.

Next, multiply each by  $360/(2^24)$  to obtain degrees

Exploring IMG Format

Check if the value of your two byte chunks is >&7FFF:

If it is, then the result is negative, ie -(65536 - value),

## **Plotting Polylines**

Before attempting to plot polylines or polyhedrons, you must be familiar to some extent with Mechalas description of bitstream parsing. This is heavy going but a few additional examples may help:

#### 06 E0 01 00 5A FF 76 FF 11 23 A5 C2 52 93 4A 77 EA 5C F0 F0 8C 21 2A 5A 4B 14 01

This represents a residential highway (06) with LBL1 ref at 00 01 E0 and longitude and latitude starting at offset 5A FF ,76 FF respectively from the centre of your zoom level map.

We know the start of the highway, but not the end, or any other nodes along its line. They are all compacted in a bit stream, highlighted in blue.

The length of this bitstream always follows the latitude byte and is highlighted in green :&11 ie 17 bytes. The next byte,23, in yellow tells us 'something' about how many bits are grouped together to determine our longitude and latitude. This may not be the same number – see further.

So far, so good. Now the fun starts:

We need to 'translate' the bit stream into binary, reading from left to right, starting at A5 and finishing at 01. Lats and longs are now determined by bits, NOT bytes, to save space. So we now have to find out how many bits are needed for each!

Part of this information is also held in the yellow byte, part in the first 'blue' byte of the bit stream.

#### the first bitstream byte

Before we examine the meaning of our A5 here are 2 examples of first bytes of a bitstream, starting LSB, from bit 0.

- a) 1011 00101 ..... etc
- b) 1110 01101 ..... Etc

The first bit is most significant;

if it is set then

1) all the longitudes in this line have the same sign ;

- 2) the second bit tells you whether they are all positive (0) or negative (1).
- 3) The third bit tells you about latitudes; if set then the fourth bit gives you the sign

Exploring IMG Format

| Longitude | Longitude | Latitude  | Latitude | Effect     |
|-----------|-----------|-----------|----------|------------|
| 1= Has    | 1= -      | 1= Has    | 1= -     |            |
| same sign | 0 = +     | same sign | 0 = +    |            |
| 1         | 0         | 1         | 1        | Going east |
|           |           |           |          | and south  |
| 1         | 1         | 1         | 0        | Going west |
|           |           |           |          | and north  |

you can see that both a and b start with the first and third byte being set (=1)

Now examine the first byte in our example above , ie A5:

In binary this is 1010<mark>0101</mark> bit 7 to bit 0

Next, reverse the bits and we see that it starts with 1010 0101 bit 0 to bit 7

This implies that longitude is always positive and latitude is positive as well (1111 0101 would have signified all values being negative)

How is this going to help us to determine the bit lengths of our latitude and longitude?

If it is always positive or negative the shape of the line tends to be a curve. If you want to check your code look for power cables or motor ways, they should general start with a 4 bit sign determinator.

### The 'official' algorithm:

Mechalas gives the following formula:

Longitude = 2 + base value + longitude sign + extra bits set in LBL (*but see further!!!*) Latitude = 2 + base value + latitude sign + extra bits set in LBL

I have found this to be incorrect; in my opinion it is:

Longitude = 2 + base value + longitude sign Latitude = 2 + base value + latitude sign

If the extra LBL bits are set then add 1 bit - this bit seems indicated the beginning of a line if set (ie 1).

When the Extra bit is set additional information about the highway is held in the NET and NOD subfile regarding speed and road type. It also means that the highway is routable.

If the sign of latitude is always the same then the latitude sign value = 0 else it is 1. Same applies to longitude.

In our example all longitude and latitude values remain the same ,ie always positive or always negative.

Let's return to the base byte (yellow above, ie ... FF 11 23 A5...): ie 23 :

1<sup>st</sup> 'digit' refers to latitude, second to longitude (MSB to LSB)

Longitude = 2 + 3 (base value) + 0 = 5

Latitude =  $2 + \frac{2}{2}$  (base value) + 0 = 4

If the base value is higher than 9, ie B4, 6A, or AC etc) then see Mechalas.

### Starting to parse bitstreams

Next, we start parsing the bit stream and begin with the fifth bit, because we've already used the first 4 to determine the sign for our coordinates.

Remember, our starting point is not always the  $5^{th}$  bit; it could be  $3^{rd}$  or  $4^{th}$ ; see following examples.

In our previous examples the purple bits always were 4 bits . Now, some examples when this is not the case.

#### Example 2



The first two bits are as above, ie 10, but then latitude is set to  $0 (3^{rd} bit)$  meaning its sign is variable and could be positive or negative. There is now no need for a 4<sup>th</sup> bit. We know that our longitude is always positive, and our latitude is variable, both positive and negative – total number of bits = 3.

Longitude = 2 + (base value) + 0 + 0Latitude = 2 + (base value) + 1 + 0 (1) indicates value being variable)

We start parsing after the 3<sup>rd</sup> bit, ie 1001...

#### Example 3

#### <mark>010</mark> 1011 ...

In this case the longitude bit starts with zero, meaning its sign is variable

Longitude = 2 + (base value) + 1Latitude = 2 + (base value) + 0

Again we start parsing after the 3<sup>rd</sup> bit , ie 1011

#### **Example 4**

#### 00 10010

Here both longitude and latitude are 0 meaning both signs are variable ; no more bits needed to mark our signs.

Longitude = 2+ (base value) + **1** Latitude = 2 + (base value) + **1** 

Parsing starts after the 2<sup>nd</sup> bit.

Exploring IMG Format

If you are writing your own code it is recommended you look at the last subdivision containing polylines as defined in TRE, as we don't need to bother about left shifting the values – more later.

(There is a minor binary value error in Mechalas description, ie 0 x ab )

We now know the length for each longitude and latitude chunk and can start parsing; remember the values are added to the coordinates of the centre of your maplevel block – see plotting POIs

Returning to our example

06 E0 01 00 5A FF 76 FF 11 23 A5 C2 52 93 4A 77 EA 5C F0 F0 8C 21 2A 5A 4B 14 01

- 1) start with 5<sup>th</sup> bit of your total stream
- 2) longitude = 3 bits and latitude = 3 bits: so group the rest of your bitstream into sets of 3's
- x of point 1 : 5A FF in degrees:  $(FF5A + centre longitude) * 360/(2^24)$
- y of point 1 : 76 FF in degrees: (FF76+ centre latitude) \* 360/(2^24)

x of point 2 : x=x + decimal(first 3 bits) in degrees :  $x * 360/(2^24)$ 

- y of point 2 : y=y + decimal(second 3 bits): in degrees :  $y * 360/(2^24)$
- x of point 2 :  $x=x + decimal(third 3 bits) : x * 360/(2^24)$
- y of point 2 :  $y=y + decimal(fourth 3 bits) : y * 360/(2^24)$

## Plotting Routable polylines

If bit 6 of the  $3^{rd}$  byte of the lbl pointer is set then the extra bit =1

In its simplest form 1 extra bit is added to the Latitude only and an extra bit is added to the start of the bitstream. Interestingly, not all highways of the same type in a subdivision are marked as routable, perhaps they are at the end of the map boundary, or not connected.

example: bitstreams starting with 1010; longitude in yellow, latitude in green

| 100 | <mark>110</mark> 1111 | 101 1001 | without routing (latitude sign is variable) | ) |
|-----|-----------------------|----------|---|---|
|-----|-----------------------|----------|---|---|

100 0 110 1111 1 101 1001 with routing (latitude sign is variable)

*Notice only 1 extra bit per coordinates.* We have not come across examples of extra bits being added to longitude as indicated by Mechalas.

Importantly and not mentioned by Mechalas, *the extra bit is added to the beginning of each longitude*.

It is not clear why the extrabit is sometimes 1 and sometimes 0, as it is not a simple case of marking a junction or not.

For more information see TRE7

## Left\_shifting Coordinates

The more you zoom out, the more sparse the map is going to be. You don't need to plot all POIs and your highways require fewer nodes, so the length of each bitstream tends to be short.

Also, crucially, you can reduce the accuracy of your coordinates and save bits.

#### Bits\_per\_coord

| Map   | Bits  | subdivision | subdivision |
|-------|-------|-------------|-------------|
| Level | per   |             |             |
|       | coord |             |             |
| 85    | &F    | 1           | 0           |
| 4     | &10   | 1           | 0           |
| 3     | &12   | 1           | 0           |
| 2     | &14   | 1           | 0           |
| 1     | &16   | 2           | 0           |
| 0     | &18   | 5           | 0           |

When we convert our data to degrees we use the formula:

Long= x \* 360/2^24 Lat: y \* 360/2^24

The third column, bottom row contains this number, &18 ie 24 dec

Each time we zoom out we use a different factor: 16,14,12,10,F

So, in our example, when we zoom out, we need to multiply our coordinates by: \* 360/2^22

## Left Shifting

However, each time we zoom out, before we obtain our degrees, we need to left shift our coordinates using the following formula: 24 – bits\_per\_coord.

Left shift means 'increase', in the sense that positive becomes more positive and negative more negative.

POI Example :

64 15 03 C0 1F 00 1C 00 0F

Supposing this POI was found in a maplevel 2 : we read off and find our bits\_per\_coord : &14 (dec 20)

First change 1F00 into bits. 1111000, starting from bit 0 to 7

We left\_shift by (24-20) ie 4 to get 00001111000: 576 or &h240

Take care when numbers to be left\_shifted are negative, ie &FFF7.

Note: you need to left\_shift all two byte values *before* adding them to the current coordinates if resolution is < 24 . Three byte values are NOT left\_shifted.

# **Plotting Polygons**

Polygons are parsed in the same way as polylines.

You will notice that most polygons are not drawn at the highest zoom level/ sublevel and that lower zoom levels generally contain a bounding box of &4B or 4A.

Interestingly and perhaps not surprisingly ,shapes are not closed.

## Special cases in a bitstream

Mechalas has given us some valid pointers to how we need to parse a bitstream chunk with ONLY its last bit set, ie 001 or 00001 etc (LSB to MSB).

However, unfortunately his description is somewhat incomplete.

Only cgpsmapper and topo maps seem to use this feature, so it is worth experimenting with cgpsmapper to unravel its obvious complexity.

Use gpsmapedit to create various zigzag lines and save as a mp. Then export using cgpsmapper.

Regard 0001 etc as a flag to indicate special cases to create larger numbers, either negative or positive.

The sign value is determined by what follows!

Examples: (from LSB to MSB)

a) 00001 001001 special case followed by negative number .

This has the effect of increasing its negative value - documented by Mechalas.

b) 00001 001010 special case followed by positive number

This has the effect of increasing its *positive* value – not documented.

Value =  $2^{4} + 20$ 

c) There is an additional case, also undocumented , when , say, 0001 is followed , often several times, by another 0001 , until a lower bit is set. - this only works if the extrabit is not set ( information given by Attila)

For example:

0001 0001 0001 1100

This creates a value of  $2^{3}(0001) + 2^{3} + 2^{3} + 3(1100)$ 

A visual example:

The following bitstream of a polygon contains somewhere in the middle some consecutive segments with last bit set only.

#### 0000001 0000001 0000001 0000001

 3F
 44
 94
 D9
 54
 B3
 C1
 CF
 10
 20
 FC
 52
 B4
 9F
 EE
 55
 53
 EC
 62
 34
 C8
 0F

 FC
 43
 ED
 02
 81
 43
 AE
 30
 81
 DC
 0C
 92
 45
 4F
 80
 1B
 30
 18
 8B
 43
 8B
 20

 3B
 DC
 00
 24
 5B
 CB
 06
 00
 81
 40
 60
 00
 08
 04
 02
 4B
 00
 00
 10
 58
 04

(length=&3F ,base value=&44 etc)

And looks like this with polygon closed:



## TRE from 0x4a

| 0 x 4aPolylines Resolution Block4 $0 x 4e$ Length of block4 $0 x 52$ Length of record block (ie L= 2)4 $0 x 55$ other length ?2 $0 x 56$ other length ?2 $0 x 55$ Dolygon Resolution Block4 $0 x 5c$ Length of record block (ie L= 2)4 $0 x 60$ Length of record block (ie L= 2)4 $0 x 60$ Length of record block (ie L= 2)4 $0 x 64$ other length2 $0 x 66$ POI Resolution Block4 $0 x 66$ POI Resolution Block4 $0 x 66$ POI Resolution Block4 $0 x 70$ other length ?2 $0 x 77$ other length ?2 $0 x 77$ other length ?2 $0 x 77$ other length ?2 $0 x 78$ 4 $0 x 78$ 2 $0 x 84$ Size of record ie 0 x 0d2 $0 x 86$ 22 $0 x 84$ Size of record ie 0 x 0d2 $0 x 98$ Extended types and draw order pointer TRE84 $0 x 92$ Size of record $0 x 5$ 2 $0 x 98$ Extended types and draw order pointer TRE84 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 04$ <   | Offset      | description   | bytes |
|---|-------------|---|-------|
| 0 x 4eLength of block4 $0 x 52$ Length of record block (ie L= 2)4 $0 x 54$ Number of L+1 bytes ? other length2 $0 x 56$ other length ?2 $0 x 55$ Polygon Resolution Block4 $0 x 5c$ Length of block4 $0 x 60$ Length of record block (ie L= 2)4 $0 x 62$ Number of L+1 bytes ? / other length2 $0 x 64$ other length2 $0 x 66$ POI Resolution Block4 $0 x 6a$ Length of block4 $0 x 6a$ Length of block (ie L= 3)4 $0 x 72$ other length ?2 $0 x 72$ other length ?2 $0 x 74$ MAP ID also used in mp section of gmapsupp4 $0 x 7c$ Pointers for subdivision for extended elements TRE74 $0 x 80$ Length of this block4 $0 x 84$ Size of record ie $0 x 0d$ 2 $0 x 84$ Size of record ie $0 x 0d$ 2 $0 x 84$ Extended types and draw order pointer TRE84 $0 x 92$ Size of record2 $0 x 94$ - ADencrypted key last 4 bytes get set to zero after decryption !19TRE1 gets decrypted - firts line of mapsets2 $0 x b6$ Size of record (0 x 5)2 $0 x b6$ Size of record (0 x 5)2 $0 x 9a$ - ADencrypted key last 4 bytes get set to zero after decryption !19TRE1 gets decrypted - firts line of mapsets4 $0 x b6$ Size of record (0 x 5)2   | 0 x 4a      | Polylines Resolution Block                                    | 4     |
| $0 \times 52$ Length of record block (ie L= 2)4 $0 \times 54$ Number of L+1 bytes? other length2 $0 \times 56$ other length?2 $0 \times 5c$ Length of block4 $0 \times 60$ Length of block (ie L= 2)4 $0 \times 60$ Length of record block (ie L= 2)4 $0 \times 64$ other length2 $0 \times 64$ other length2 $0 \times 66$ POI Resolution Block4 $0 \times 6a$ Length of record block (ie L= 3)4 $0 \times 6a$ Length of record block (ie L= 3)4 $0 \times 70$ other length?2 $0 \times 72$ other length?2 $0 \times 74$ MAP ID also used in mp section of gmapsupp4 $0 \times 7c$ Pointers for subdivision for extended elements TRE74 $0 \times 80$ Length of this block2 $0 \times 86$ 22 $0 \times 86$ 22 $0 \times 94$ Size of record ie 0 x 0d2 $0 \times 94$ Size of record2 $0 \times 88$ 22 $0 \times 94$ Size of record2 $0 \times 86$ 22 $0 \times 94$ Dencypted 4 draw order pointer TRE84 $0 \times 94$ Size of record2 $0 \times 94$ Dencypted 4 bytes get set to zero after decryption !19 $0 \times 94$ Dencrypted key last 4 bytes get set to zero after decryption !19 $0 \times 94$ Dencrypted key last 4 bytes get set to zero after decryption !2 $0 \times 94$ Dencrypted key last 4 bytes get set to zero after decryption !2   | 0 x 4e      | Length of block   | 4     |
| 0 x 54Number of L+1 bytes ? other length2 $0 x 56$ other length ?2 $0 x 55$ Polygon Resolution Block4 $0 x 5c$ Length of block (ie L= 2)4 $0 x 60$ Length of record block (ie L= 2)4 $0 x 60$ Number of L+1 bytes ? / other length2 $0 x 66$ POI Resolution Block4 $0 x 66$ POI Resolution Block (ie L= 3)4 $0 x 66$ POI Resolution Block (ie L= 3)4 $0 x 60$ Length of block (ie L= 3)4 $0 x 70$ other length ?2 $0 x 72$ other length ?2 $0 x 72$ other length ?4 $0 x 7c$ Pointers for subdivision for extended elements TRE74 $0 x 80$ Length of this block4 $0 x 80$ Length of this block2 $0 x 86$ 22 $0 x 86$ 22 $0 x 88$ Length of this block4 $0 x 92$ Size of record ie 0 x 0d2 $0 x 92$ Size of record2 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 92$ Size of record of mapsets2 $0 x 94$ Pathers based to mapsets2 $0 x 94$ Apple Length of TRE104 $0 x 66$ Size of record (0 x 5)2 $0 x 94$ Nother length of this block4 $0 x 92$ Size of record 102 $0 x 94$ Deck decrypted - firts line of mapsets2 $0 x 94$ Decrypted key last 4 bytes   | 0 x 52      | Length of record block (ie $L=2$ )                            | 4     |
| 0 x 56other length ?2 $0x 58$ Polygon Resolution Block4 $0 x 5c$ Length of block4 $0 x 60$ Length of record block (ie L= 2)4 $0 x 62$ Number of L+1 bytes ? / other length2 $0 x 64$ other length2 $0 x 66$ POI Resolution Block4 $0 x 6a$ Length of block (ie L= 3)4 $0 x 6a$ Length of fecord block (ie L= 3)4 $0 x 70$ other length ? (ie L=5)2 $0 x 74$ MAP ID also used in mp section of gmapsupp4 $0 x 78$ 04 $0 x 78$ 04 $0 x 84$ Size of record ie 0 x 0d2 $0 x 86$ 22 $0 x 88$ Extended types and draw order pointer TRE84 $0 x 92$ Size of record2 $0 x 98$ 22 $0 x 98$ 22 $0 x 98$ 22 $0 x 94$ Nize of record (0 x 5)2 $0 x 94$ Nize of record (0 x 5)2 $0 x 84$ Size of record (0 x 5)2 $0 x 94$ Dencrypted - firts line of mapsets4 $0 x 94$ Dencrypted - firts line of mapsets4 $0 x 94$ Nize of record (0 x 5)2 $0 x 64$ $0 x 01$ 2? $0 x 64$ $0$   | 0 x 54      | Number of L+1 bytes ? other length                            | 2     |
| 0x 58Polygon Resolution Block4 $0x 5c$ Length of block4 $0x 60$ Length of record block (ie L= 2)4 $0x 64$ other length2 $0x 64$ other length2 $0x 66$ POI Resolution Block4 $0x 6a$ Length of record block (ie L= 3)4 $0x 70$ other length?2 $0x 72$ other length?2 $0x 772$ other length?2 $0x 772$ other length?2 $0x 772$ other length?2 $0x 772$ other length?4 $0x 772$ other length?4 $0x 772$ other length?4 $0x 78$ 04 $0x 76$ Pointers for subdivision for extended elements TRE74 $0x 80$ Length of this block4 $0x 84$ Size of record ie 0 x 0d2 $0x 84$ Size of record ie 0 x 0d2 $0x 84$ Extended types and draw order pointer TRE84 $0x 92$ Size of record2 $0x 96+$ Values used to encrypt data based on map-id2 $0x 98+$ 22 $0x 98+$ 22 $0x 84$ Size of record (0 x 5)2 $0x 84$ Size of record (0 x 5)2 $0x 65$ TRE 10n block4 $0x 64$ $0x 01$ 2 $0x$   | 0 x 56      | other length ?  | 2     |
| $0 \times 5c$ Length of block4 $0 \times 60$ Length of record block (ie L= 2)4 $0 \times 62$ Number of L+1 bytes ? / other length2 $0 \times 64$ other length2 $0 \times 66$ POI Resolution Block4 $0 \times 6a$ Length of block (ie L= 3)4 $0 \times 70$ other length ? (ie L=5)2 $0 \times 72$ other length ? (ie L=5)2 $0 \times 72$ other length ?2 $0 \times 72$ other length ?4 $0 \times 73$ MAP ID also used in mp section of gmapsupp4 $0 \times 72$ other length of this block4 $0 \times 80$ Length of this block4 $0 \times 84$ Size of record ie $0 \times 0d$ 2 $0 \times 84$ Size of record2 $0 \times 88$ Extended types and draw order pointer TRE84 $0 \times 82$ Size of record2 $0 \times 98$ 22 $0 \times 98$ 22 $0 \times 98$ 22 $0 \times b2$ Length of block4 $0 \times b2$ Length of block4 $0 \times b4$ Size of record $(0 \times 5)$ 2 $0 \times b6$ TRE194 $0 \times b2$ Length of block4 $0 \times b2$ Length of block4 $0 \times b4$ Olock4 $0 \times b4$ TRE1004 $0 \times b4$ Olock4   | 0x 58       | Polygon Resolution Block                                      | 4     |
| 0 x 60Length of record block (ie L= 2)4 $0 x 62$ Number of L+1 bytes ? / other length2 $0 x 64$ other length2 $0 x 66$ POI Resolution Block4 $0 x 6a$ Length of block (ie L= 3)4 $0 x 70$ other length ? (ie L=5)2 $0 x 72$ other length ? (ie L=5)2 $0 x 74$ MAP ID also used in mp section of gmapsupp4 $0 x 78$ 4 $0 x 80$ Length of this block4 $0 x 80$ Length of this block4 $0 x 84$ Size of record ie 0 x 0d2 $0 x 88$ 22 $0 x 88$ 22 $0 x 84$ Size of record2 $0 x 84$ Size of record2 $0 x 84$ Length of this block4 $0 x 84$ Size of record2 $0 x 92$ Size of record2 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 94$ ADencrypted key last 4 bytes get set to zero after decryption !<br>TRE1 gets decrypted - firts line of mapsets19 $0 x b2$ Length of block4 $0 x b4$ Size of record ( $0 x 5$ )2 $0 x b4$ TRE 10n block4 $0 x c4$ $0 x 01$ 2? $0 x c4$ $0 x 01$ 2? $0 x c5$ Ant the length of the le   | 0 x 5c      | Length of block   | 4     |
| 0 x 62Number of L+1 bytes ? / other length2 $0 x 64$ other length2 $0 x 66$ POI Resolution Block4 $0 x 6a$ Length of block (ie L= 3)4 $0 x 6a$ Length of record block (ie L= 3)4 $0 x 70$ other length ? (ie L=5)2 $0 x 72$ other length ? (ie L=5)2 $0 x 74$ MAP ID also used in mp section of gmapsupp4 $0 x 78$ 44 $0 x 80$ Length of this block4 $0 x 80$ Length of this block4 $0 x 84$ Size of record ie 0 x 0d2 $0 x 84$ Size of record ie 0 x 0d2 $0 x 84$ Size of record2 $0 x 84$ Extended types and draw order pointer TRE84 $0 x 92$ Size of record2 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 98$ 22 $0 x 98$ 22 $0 x b6$ Size of record (0 x 5)2 $0 x b6$ Size of record (0 x 5)2 $0 x b6$ Size of record (0 x 5)2 $0 x b6$ Size of record (0 x 5)2 $0 x b6$ TRE 10n block4 $0 x c4$ $0 x 01$ 2? $0 x c6$ 44 $0 x c4$ $0 x 01$ 2? $0 x c5$ Auge as CF as a new header2 $0 x d3$ Parameters 03 002 $0 x d3$ Parameters 03 002 $0 x d3$ Parameters 03 002  | 0 x 60      | Length of record block (ie $L=2$ )                            | 4     |
| $0 \times 64$ other length2 $0 \times 66$ POI Resolution Block4 $0 \times 6a$ Length of block4 $0 \times 6a$ Length of record block (ie L= 3)4 $0 \times 70$ other length ? (ie L=5)2 $0 \times 72$ other length ?2 $0 \times 73$ MAP ID also used in mp section of gmapsupp4 $0 \times 78$ 44 $0 \times 78$ 4 $0 \times 78$ 4 $0 \times 78$ 4 $0 \times 80$ Length of this block4 $0 \times 84$ Size of record ie $0 \times 0d$ 2 $0 \times 84$ Size of record ie $0 \times 0d$ 2 $0 \times 86$ 22 $0 \times 8a$ Extended types and draw order pointer TRE84 $0 \times 92$ Size of record2 $0 \times 94$ Values used to encrypt data based on map-id2 $0 \times 94$ Values used to encrypt data based on map-id2 $0 \times 94$ Values used to encrypt data based on map-id2 $0 \times 95$ 222 $0 \times 94$ ADencrypted key last 4 bytes get set to zero after decryption !<br>TRE1 gets decrypted - firts line of mapsets4 $0 \times b5$ Size of record $(0 \times 5)$ 22 $0 \times b6$ Size of record $(0 \times 5)$ 22 $0 \times b6$ Size of record $(0 \times 5)$ 2 $0 \times b6$ Size of record $(0 \times 5)$ 2 $0 \times b6$ Size of record $(0 \times 5)$ 2 $0 \times b6$ Size of record $(0 \times 5)$ 2 $0 \times c6$ 44 $0 \times c6$ Must be 2 as CF as a  | 0 x 62      | Number of L+1 bytes ? / other length                          | 2     |
| $0 \times 66$ POI Resolution Block4 $0 \times 6a$ Length of block (ie L= 3)4 $0 \times 76$ Length of record block (ie L= 3)4 $0 \times 70$ other length ? (ie L=5)2 $0 \times 74$ MAP ID also used in mp section of gmapsupp4 $0 \times 78$ 4 $0 \times 78$ 4 $0 \times 76$ Pointers for subdivision for extended elements TRE74 $0 \times 80$ Length of this block4 $0 \times 84$ Size of record ie 0 x 0d2 $0 \times 86$ 2 $0 \times 88$ 2 $0 \times 88$ 2 $0 \times 88$ 2 $0 \times 88$ 2 $0 \times 92$ Size of record $0 \times 92$ Size of record $0 \times 94$ 2 $0 \times 94$ Values used to encrypt data based on map-id $0 \times 98$ 2 $0 \times 98$ 2 $0 \times 94$ Values used to encrypt data based on map-id $0 \times 92$ Length of block $0 \times 94$ Values used to encrypt data based on map-id $0 \times 98$ 2 $0 \times 98$ 2 $0 \times 84$ Size of record ( $0 \times 5$ ) $0 \times 84$ Size of record ( $0 \times 5$ ) $0 \times 84$ 3 $0 \times 64$ $0 \times 01$ $0 \times 75$ 2 $0 \times 78$ 4 $0 \times 98$ 2 $0 \times 84$ 4 $0 \times 78$ 4 $0 \times 78$ 4 </td <td>0 x 64</td> <td>other length</td> <td>2</td>  | 0 x 64      | other length  | 2     |
| $0 \times 6a$ Length of block4 $0 \times 6e$ Length of record block (ie L= 3)4 $0 \times 70$ other length ?2 $0 \times 72$ other length ?2 $0 \times 74$ MAP ID also used in mp section of gmapsupp4 $0 \times 78$ 4 $0 \times 78$ 4 $0 \times 78$ 4 $0 \times 80$ Length of this block4 $0 \times 80$ Length of this block4 $0 \times 86$ 2 $0 \times 86$ 2 $0 \times 88$ 2 $0 \times 88$ 2 $0 \times 88$ 2 $0 \times 88$ 2 $0 \times 98$ 2 $0 \times 96$ 2 $0 \times 98$ 2 $0 \times 98$ 2 $0 \times 80$ encrypted key last 4 bytes get set to zero after decryption !<br>TRE1 gets decrypted - firts line of mapsets $0 \times 80$ Length of block4 $0 \times b2$ Length of block4 $0 \times b3$ 2 $0 \times 60$ TRE 10 block4 $0 \times b4$ 22 $0 \times 60$ Length of TRE104 $0 \times c4$ $0 \times 01$ 2? $0 \times c4$ $0 \times 01$ 2? $0 \times c5$ Must be 2 as CF as a new header2 $0 \times d3$ Parameters 03 002 $0 \times d3$ Parameters 03 002 $0 \times d3$ Parameters 03 002 $0 \times d3$ Parameters4  | 0 x 66      | POI Resolution Block  | 4     |
| 0 x 6eLength of record block (ie L= 3)40 x 70other length ? (ie L=5)20 x 72other length ?20 x 74MAP ID also used in mp section of gmapsupp40 x 75440 x 7cPointers for subdivision for extended elements TRE740 x 80Length of this block40 x 84Size of record ie 0 x 0d20 x 8520 x 8620 x 8820 x 84Extended types and draw order pointer TRE840 x 82Length of this block40 x 84Extended types and draw order pointer TRE840 x 86220 x 88220 x 84Extended types and draw order pointer TRE840 x 92Size of record20 x 94Values used to encrypt data based on map-id20 x 98220 x 9820 x 94TRE1 gets decrypted - firts line of mapsets19TRE1 gets decrypted - firts line of mapsets20 x bcTRE 10 block40 x bcTRE 10 block40 x c40 x 012?0 x c64?4?0 x c644?0 x c644?0 x c6440 x c640 x c4 <t< td=""><td>0 x 6a</td><td>Length of block</td><td>4</td></t<>   | 0 x 6a      | Length of block   | 4     |
| $0 \times 70$ other length ? (ie L=5)2 $0 \times 72$ other length ?2 $0 \times 77$ MAP ID also used in mp section of gmapsupp4 $0 \times 78$ 4 $0 \times 77$ Pointers for subdivision for extended elements TRE74 $0 \times 70$ Pointers for subdivision for extended elements TRE74 $0 \times 80$ Length of this block4 $0 \times 84$ Size of record ie $0 \times 0d$ 2 $0 \times 84$ Size of record ie $0 \times 0d$ 2 $0 \times 86$ 2 $0 \times 88$ 2 $0 \times 8a$ Extended types and draw order pointer TRE84 $0 \times 8e$ Length of this block4 $0 \times 92$ Size of record2 $0 \times 94$ Values used to encrypt data based on map-id2 $0 \times 98$ 22 $0 \times 98$ 2 $0 \times 9a$ - ADencrypted key last 4 bytes get set to zero after decryption !<br>TRE1 gets decrypted - firts line of mapsets19 $0 \times b2$ Length of block4 $0 \times b4$ Size of record $(0 \times 5)$ 2 $0 \times b4$ Size of record $(0 \times 5)$ 2 $0 \times b4$ N to 12? $0 \times c4$ $0 \times 01$ 2? $0 \times c5$ Austor for an enclose of a size of Fas a new header2 $0 \times c5$ Austor for a size of Fas a new header2 $0 \times c4$ $0 \times 0300$ 2 $0 \times d3$ Parameters $0300$ 2<  | 0 x 6e      | Length of record block (ie $L=3$ )                            | 4     |
| $0 \ge 72$ other length ?2 $0 \ge 74$ MAP ID also used in mp section of gmapsupp4 $0 \ge 78$ 4 $0 \ge 78$ 4 $0 \ge 76$ Pointers for subdivision for extended elements TRE74 $0 \ge 80$ Length of this block4 $0 \ge 84$ Size of record ie $0 \ge 0d$ 2 $0 \ge 86$ 2 $0 \ge 88$ 2 $0 \ge 82$ Size of record $0 \ge 92$ Size of record $0 \ge 92$ Size of record $0 \ge 94$ Values used to encrypt data based on map-id $0 \ge 94$ 2 $0 \ge 94$ Values used to encrypt data based on map-id $0 \ge 94$ 2 $0 \ge 94$ 2 $0 \ge 94$ 2 $0 \ge 94$ A $0 \ge 94$ A $0 \ge 0 \ge 12$ A $0 \ge 22$ $0 \ge 22$ $0 \ge 94$ A $0 \ge 22$ $0 \ge 22$ $0 \ge 94$ $0 \ge 22$ $0 \ge 24$ $0 \ge 02$ <t< td=""><td>0 x 70</td><td>other length? (ie L=5)</td><td>2</td></t<>  | 0 x 70      | other length? (ie L=5)  | 2     |
| $0 \times 74$ MAP ID also used in mp section of gmapsupp4 $0 \times 78$ 4 $0 \times 78$ 4 $0 \times 7c$ Pointers for subdivision for extended elements TRE74 $0 \times 80$ Length of this block4 $0 \times 84$ Size of record ie $0 \times 0d$ 2 $0 \times 84$ Size of record ie $0 \times 0d$ 2 $0 \times 86$ 2 $0 \times 88$ 2 $0 \times 8a$ Extended types and draw order pointer TRE84 $0 \times 8e$ Length of this block4 $0 \times 92$ Size of record2 $0 \times 94$ Values used to encrypt data based on map-id2 $0 \times 96$ Values used to encrypt data based on map-id2 $0 \times 98$ 22 $0 \times 98$ 2 $0 \times 98$ 2 $0 \times ae$ TRE 94 $0 \times b2$ Length of block4 $0 \times b4$ Size of record $(0 \times 5)$ 2 $0 \times b5$ Size of record $(0 \times 5)$ 2 $0 \times b6$ Size of record $(0 \times 5)$ 2 $0 \times b6$ Size of record $(0 \times 5)$ 2 $0 \times b6$ NT TRE10 block4 $0 \times c6$ 44 $0 \times c6$ 4 <tr< td=""><td>0 x 72</td><td>other length ?</td><td>2</td></tr<>  | 0 x 72      | other length ?  | 2     |
| $0 \ge 78$ 4 $0 \ge 776$ Pointers for subdivision for extended elements TRE74 $0 \ge 80$ Length of this block4 $0 \ge 80$ Size of record ie $0 \ge 0$ d2 $0 \ge 84$ Size of record ie $0 \ge 0$ d2 $0 \ge 86$ 2 $0 \ge 86$ 2 $0 \ge 86$ 2 $0 \ge 88$ 2 $0 \ge 92$ Size of record2 $0 \ge 96+$ Values used to encrypt data based on map-id2 $0 \ge 96+$ Values used to encrypt data based on map-id2 $0 \ge 98$ 22 $0 \ge 94-$ Values used to encrypt data based on map-id19TRE1 gets decrypted - firts line of mapsets19 $0 \ge 92-$ Length of block4 $0 \ge b2-$ Length of block4 $0 \ge b2-$ Length of block4 $0 \ge b2-$ Length of TRE104 $0 \ge c6-$ 4?2? $0 \ge c6-$ 4?2? $0 \ge c6-$ 4?4? $0 \ge c6-$ 4?4? $0 \ge c6-$ 4?4? $0 \ge c6-$ 4?4? $0 \ge c6-$ 4?4 $0 \ge c6-$ 4.4 $0 \ge c6-$ 4.4 $0 \ge c6-$ 4.4 </td <td>0 x 74</td> <td>MAP ID also used in mp section of gmapsupp</td> <td>4</td>  | 0 x 74      | MAP ID also used in mp section of gmapsupp                    | 4     |
| $0 \ge 7c$ Pointers for subdivision for extended elements TRE74 $0 \ge 80$ Length of this block4 $0 \ge 84$ Size of record ie $0 \ge 0d$ 2 $0 \ge 86$ 2 $0 \ge 88$ 2 $0 \ge 88$ 2 $0 \ge 88$ 2 $0 \ge 88$ 4 $0 \ge 88$ 2 $0 \ge 88$ 4 $0 \ge 92$ Size of record $0 \ge 92$ Size of record $0 \ge 92$ Size of record $0 \ge 96+$ Values used to encrypt data based on map-id $0 \ge 96+$ Values used to encrypt data based on map-id $0 \ge 98-$ 2 $0 \ge 98-$ 2 $0 \ge 98-$ 2 $0 \ge 98-$ 4 $0 \ge 98-$ 2 $0 \ge 98-$ 4 $0 \ge 98-$ 2 </td <td>0 x 78</td> <td></td> <td>4</td>  | 0 x 78      |   | 4     |
| $0 \ge 80$ Length of this block4 $0 \ge 84$ Size of record ie $0 \ge 0d$ 2 $0 \ge 86$ 2 $0 \ge 86$ 2 $0 \ge 88$ 2 $0 \ge 88$ 2 $0 \ge 8$ 4 $0 \ge 92$ Size of record2 $0 \ge 96+$ Values used to encrypt data based on map-id2 $0 \ge 96+$ Values used to encrypt data based on map-id2 $0 \ge 96+$ Values used to encrypt data based on map-id2 $0 \ge 98-$ 22 $0 \ge 94-$ Patheter (a the state of the state o | 0 x 7c      | Pointers for subdivision for extended elements TRE7           | 4     |
| 0 x 84Size of record ie $0 x 0d$ $2$ $0 x 86$ $2$ $0 x 88$ $2$ $0 x 8a$ Extended types and draw order pointer TRE8 $4$ $0 x 8e$ Length of this block $4$ $0 x 92$ Size of record $2$ $0 x 96+$ Values used to encrypt data based on map-id $2$ $0 x 98$ $2$ $0 x 98$ $2$ $0 x 9a$ - ADencrypted key last 4 bytes get set to zero after decryption !<br>TRE1 gets decrypted - firts line of mapsets $19$ $0 x ae$ TRE 9 $4$ $0 x b2$ Length of block $4$ $0 x b4$ Size of record ( $0 x 5$ ) $2$ $0 x b6$ Size of record ( $0 x 5$ ) $2$ $0 x b6$ TRE 10n block $4$ $0 x c0$ Length of TRE10 $4$ $0 x c4$ $0 x 01$ $2?$ $0 x c6$ $4?$ $0 x c4$ $0 x 01$ $2?$ $0 x c6$ $4?$ $0 x c5$ A byte parameter $4$ $0 x d3$ Parameters 03 00 $2$ $0 x d5$ A block $4$  | 0 x 80      | Length of this block  | 4     |
| $0 \times 86$ 2 $0 \times 88$ 2 $0 \times 8a$ Extended types and draw order pointer TRE84 $0 \times 8e$ Length of this block4 $0 \times 8e$ Length of this block4 $0 \times 92$ Size of record2 $0 \times 96+$ Values used to encrypt data based on map-id2 $0 \times 98$ 2 $0 \times 9a$ ADencrypted key last 4 bytes get set to zero after decryption !<br>TRE1 gets decrypted - firts line of mapsets19 $0 \times ae$ TRE 94 $0 \times b2$ Length of block4 $0 \times b4$ Size of record ( $0 \times 5$ )2 $0 \times b6$ Size of record ( $0 \times 5$ )2 $0 \times b6$ TRE 10n block4 $0 \times c4$ $0 \times 01$ 2? $0 \times c6$ 4? $0 \times c4$ $0 \times 01$ 2? $0 \times c6$ 4? $0 \times cE$ Must be 2 as CF as a new header2 $0 \times d3$ Parameters 03 002 $0 \times d5$ A block4   | 0 x 84      | Size of record ie 0 x 0d                                      | 2     |
| 0 x 882 $0 x 8a$ Extended types and draw order pointer TRE84 $0 x 8e$ Length of this block4 $0 x 92$ Size of record2 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 98$ 2 $0 x 98$ 2 $0 x 9a$ - ADencrypted key last 4 bytes get set to zero after decryption !19TRE1 gets decrypted - firts line of mapsets19 $0 x ae$ TRE 94 $0 x b2$ Length of block4 $0 x b6$ Size of record $(0 x 5)$ 2 $0 x b8$ 22 $0 x bc$ TRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4?2? $0 x c6$ 44? $0 x c6$ Must be 2 as CF as a new header2 $0 x d3$ Parameters 03 002 $0 x d5$ A block4   | 0 x 86      |   | 2     |
| 0 x 8aExtended types and draw order pointer TRE84 $0 x 8e$ Length of this block4 $0 x 92$ Size of record2 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 98$ 2 $0 x 98$ 2 $0 x 9a - AD$ encrypted key last 4 bytes get set to zero after decryption !19TRE1 gets decrypted - firts line of mapsets4 $0 x b2$ Length of block4 $0 x b5$ Length of block4 $0 x b6$ Size of record ( $0 x 5$ )2 $0 x b6$ Size of record ( $0 x 5$ )2 $0 x b6$ TRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4? $0 x ca$ NT TRE11? & 1A4 $0 x cE$ Must be 2 as CF as a new header2 $0 x d3$ Parameters 03 002 $0 x d5$ A block4  | 0 x 88      |   | 2     |
| 0 x 8eLength of this block4 $0 x 92$ Size of record2 $0 x 94$ Values used to encrypt data based on map-id2 $0 x 98$ 2 $0 x 98$ 2 $0 x 98$ 19TRE1 gets decrypted + firts line of mapsets19 $0 x ae$ TRE 94 $0 x b2$ Length of block4 $0 x b6$ Size of record ( $0 x 5$ )2 $0 x b6$ Size of record ( $0 x 5$ )2 $0 x b7$ TRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4? $0 x c4$ $0 x 01$ 2? $0 x c4$ $0 x 01$ 2? $0 x c4$ $0 x 01$ 2? $0 x c4$ $0 x 01$ 4 $0 x c4$ $0 x 01$ 4 $0 x c4$ $0 x 01$ 4 $0 x c4$ $0 x 01$ 2 $0 x c4$ $0 x 0F$ $4$ $0 x c4$ $0 x 0F$ $4$ $0 x d3$ Parameters $03 00$ $0 x d5$ $A$ block $4$  | 0 x 8a      | Extended types and draw order pointer TRE8                    | 4     |
| $0 \ge 92$ Size of record2 $0 \ge 96+$ Values used to encrypt data based on map-id2 $0 \ge 98$ 2 $0 \ge 98$ 2 $0 \ge 98$ 2 $0 \ge 98$ 19TRE1 gets decrypted key last 4 bytes get set to zero after decryption !19TRE1 gets decrypted - firts line of mapsets4 $0 \ge b2$ Length of block4 $0 \ge b6$ Size of record $(0 \ge 5)$ 2 $0 \ge b6$ Size of record $(0 \ge 5)$ 2 $0 \ge b6$ TRE 10n block4 $0 \ge c0$ Length of TRE104 $0 \ge c2$ $0 \ge 0$ 2? $0 \ge c6$ 4? $0 \ge c6$ 4 $0$  | 0 x 8e      | Length of this block  | 4     |
| $0 \ge 96+$ Values used to encrypt data based on map-id2 $0 \ge 98$ 2 $0 \ge 98$ 2 $0 \ge 98$ 19TRE1 gets decrypted key last 4 bytes get set to zero after decryption !19TRE1 gets decrypted - firts line of mapsets4 $0 \ge ae$ TRE 9 $0 \ge ae$ Length of block4 $0 \ge b6$ Size of record ( $0 \ge 5$ )2 $0 \ge b6$ Size of record ( $0 \ge 5$ )2 $0 \ge b8$ 2 $0 \ge bc$ TRE 10n block4 $0 \ge c0$ Length of TRE104 $0 \ge c0$ Length of TRE104 $0 \ge c6$ 4? $0 \ge c6$ 4 $0 \ge c7$ 4 $0 \ge c6$  | 0 x 92      | Size of record  | 2     |
| $0 \ge 98$ 2 $0 \ge 98$ encrypted key last 4 bytes get set to zero after decryption !19TRE1 gets decrypted - firts line of mapsets4 $0 \ge ae$ TRE 94 $0 \ge b2$ Length of block4 $0 \ge b6$ Size of record $(0 \ge 5)$ 2 $0 \ge b8$ 2 $0 \ge bc$ TRE 10n block4 $0 \ge c0$ Length of TRE104 $0 \ge c0$ Length of TRE104 $0 \ge c4$ $0 \ge 01$ 2? $0 \ge c6$ 4?2? $0 \ge c6$ 4? $0 \ge c6$ 4? $0 \ge c6$ 4? $0 \ge c6$ 4? $0 \ge c6$ 4 $0 \ge c6$ 4 $0 \ge c7$ 4 byte parameter $0 \ge c7$ 4 byte parameter $0 \ge c7$ 4 byte parameter $0 \ge d3$ Parameters 03 002 $0 \ge d5$ A block4  | 0 x 96+     | Values used to encrypt data based on map-id                   | 2     |
| $0 \ge 9a - AD$ encrypted key last 4 bytes get set to zero after decryption !19 $TRE1$ gets decrypted - firts line of mapsets4 $0 \ge ae$ $TRE 9$ 4 $0 \ge b2$ Length of block4 $0 \ge b6$ Size of record $(0 \ge 5)$ 2 $0 \ge b8$ 2 $0 \ge b8$ 2 $0 \ge bc$ $TRE 10n \ block$ 4 $0 \ge c0$ Length of TRE104 $0 \ge c0$ Length of TRE104 $0 \ge c4$ $0 \ge 01$ 2? $0 \ge c6$ 4? $0 \ge c6$ 4? $0 \ge cE$ Must be 2 as CF as a new header2 $0 \ge cF$ 4 byte parameter4 $0 \ge cd3$ Parameters 03 002 $0 \ge d5$ A block4  | 0 x 98      |   | 2     |
| TRE1 gets decrypted - firts line of mapsets4 $0 x ae$ TRE 94 $0 x b2$ Length of block4 $0 x b6$ Size of record ( $0 x 5$ )2 $0 x b8$ 2 $0 x bc$ TRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4? $0 x c6$ 4? $0 x c2$ NT TRE11? &1A4 $0 x cE$ Must be 2 as CF as a new header2 $0 x CF$ 4 byte parameter4 $0 x d3$ Parameters 03 002 $0 x d5$ A block4   | 0 x 9a - AD | encrypted key last 4 bytes get set to zero after decryption ! | 19    |
| 0 x aeTRE 94 $0 x b2$ Length of block4 $0 x b6$ Size of record $(0 x 5)$ 2 $0 x b6$ Size of record $(0 x 5)$ 2 $0 x b8$ 2 $0 x bc$ TRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4? $0 x c6$ 4? $0 x ca$ NT TRE11? &1A4 $0 x cE$ Must be 2 as CF as a new header2 $0 x CF$ 4 byte parameter4 $0 x d3$ Parameters 03 002 $0 x d5$ A block4  |             | TRE1 gets decrypted - firts line of mapsets                   |       |
| 0 x b2Length of block4 $0 x b6$ Size of record $(0 x 5)$ 2 $0 x b8$ 2 $0 x bc$ TRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4? $0 x c6$ 4? $0 x c2$ Must be 2 as CF as a new header2 $0 x CF$ 4 byte parameter4 $0 x d3$ Parameters 03 002 $0 x d5$ A block4  | 0 x ae      | TRE 9   | 4     |
| 0 x b6Size of record $(0 x 5)$ 2 $0 x b8$ 2 $0 x bc$ TRE 10n block $0 x c0$ Length of TRE10 $0 x c4$ $0 x 01$ $0 x c4$ $0 x 01$ $0 x c6$ 4? $0 x ca$ NT TRE11? &1A $0 x cE$ Must be 2 as CF as a new header $0 x CF$ 4 byte parameter $0 x d3$ Parameters 03 00 $0 x d5$ A block  | 0 x b2      | Length of block   | 4     |
| 0 x b82 $0 x bc$ TRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4? $0 x c6$ 4? $0 x ca$ NT TRE11? &1A $0 x cE$ Must be 2 as CF as a new header2 $0 x CF$ 4 byte parameter4 $0 x d3$ Parameters 03 002 $0 x d5$ A block4   | 0 x b6      | Size of record (0 x 5)  | 2     |
| 0 x bcTRE 10n block4 $0 x c0$ Length of TRE104 $0 x c4$ $0 x 01$ 2? $0 x c6$ 4? $0 x ca$ NT TRE11? &1A4 $0 x cE$ Must be 2 as CF as a new header2 $0 x CF$ 4 byte parameter4 $0 x d3$ Parameters 03 002 $0 x d5$ A block4   | 0 x b8      |   | 2     |
| $0 \ge c0$ Length of TRE104 $0 \ge c4$ $0 \ge 01$ 2? $0 \ge c6$ 4? $0 \ge c6$ 4? $0 \ge ca$ NT TRE11? & 1A $0 \ge ca$ Must be 2 as CF as a new header $0 \ge cF$ 4 byte parameter $0 \ge CF$ 4 byte parameter $0 \ge d3$ Parameters 03 00 $0 \ge d5$ A block $4$  | 0 x bc      | TRE 10n block   | 4     |
| $0 \ge c4$ $0 \ge 01$ $2?$ $0 \ge c6$ 4? $0 \ge ca$ NT TRE11? & 1A $0 \ge ca$ Must be 2 as CF as a new header $0 \ge cE$ Must be 2 as CF as a new header $0 \ge cF$ 4 byte parameter $0 \ge cF$ 4 byte parameter $0 \ge d3$ Parameters 03 00 $0 \ge d5$ A block $4$   | 0 x c0      | Length of TRE10   | 4     |
| 0 x c6       4?         0 x ca       NT TRE11? &1A       4         0 x cE       Must be 2 as CF as a new header       2         0 x CF       4 byte parameter       4         0 x d3       Parameters 03 00       2         0 x d5       A block       4  | 0 x c4      | 0 x 01  | 2?    |
| 0 x caNT TRE11? &1A40 x cEMust be 2 as CF as a new header20 x CF4 byte parameter40 x d3Parameters 03 0020 x d5A block4  | 0 x c6      |   | 4?    |
| 0 x cEMust be 2 as CF as a new header20 x CF4 byte parameter40 x d3Parameters 03 0020 x d5A block4  | 0 x ca      | NT TRE11? &1A   | 4     |
| 0 x CF         4 byte parameter         4           0 x d3         Parameters 03 00         2           0 x d5         A block         4  | 0 x cE      | Must be 2 as CF as a new header                               | 2     |
| $\begin{array}{c c} 0 \times d3 \\ \hline 0 \times d5 \\ \hline 0 \times d5 \\ \hline A \ block \\ \hline 4 \\ \hline \end{array}$  | 0 x CF      | 4 byte parameter  | 4     |
| 0 x d3         Parameters 03 00         2           0 x d5         A block         4  |             |   |       |
| 0 x d5 A block 4  | 0 x d3      | Parameters 03 00  | 2     |
| T CIOVA   | 0 x d5      | A block   | 4     |

| 0 x e3  | block                                | 4 |
|---------|--------------------------------------|---|
| 0 x e7  | size                                 | 4 |
| 0 x eb  | Record length (6)                    | 2 |
| 0 x ed  | Block looks like a pointer or length | 4 |
|         |                                      |   |
|         |                                      |   |
| 0x f1   | block                                | 4 |
| 0 x f5  | length                               | 4 |
|         |                                      |   |
| 0 x fb  | block                                | 4 |
| 0 x ff  | length                               | 4 |
| 0 x 103 | Record length (?) 9                  | 2 |

See headers up to &110

Some non NT imgs have a TRE header with length 0 x ca but most of are 0 x bc long.

The latest 2012 TOPO imgs have headers up to &da

No idea what TRE10 or 11 represent.

A lot of the TRE is now clear and documented ; but there is still a lot to do!

To give you a taster ,the PIDs are kept in a single byte with say EF down to E6 incrementing the PID and E0 to E5 decreasing its value! Offset rule varies and is linked to values stored from 0x96. Similarly with the FID (2 bytes), proceeding the PID (1 byte). There is no problem obtaining both values in a gmapsupp as they are kept in the mps section.

## TRE7

Each block of extended elements, contains elements at various resolutions. So, if your IMG contains extended pois or polylines etc then they too are plotted at different resolutions.

Trouble is, that without any pointers, it is impossible to plot these extended elements correctly as they may need to be left\_shifted

Fortunately,TRE7 provides essential pointers for each subdivision. Interestingly, in locked files these are not scrambled!

Pointers for each subdivision are found at 0x7C offset from TRE, called TRE7, usually in blocks of 13. Length of this block at 0x80

The size of each block is defined in TRE + &84, ie &0D

So, the length of each group of extended pois is determined by the beginning of the next offset or then end of the block itself. Each offset is always calculated from the beginning of each block of extended elements.

| Polygons 4 bytes | Polylines 4 bytes | POIs 4 bytes | Refers to number of    |
|------------------|-------------------|--------------|------------------------|
|                  |                   |              | element types in nxt   |
|                  |                   |              | subdivision (0 - 3)    |
|                  |                   |              | 0 means nothing to     |
|                  |                   |              | follow                 |
|                  |                   |              | 3 means pois polylines |
|                  |                   |              | & polygons)            |
|                  |                   |              |                        |

example of tre7 (magenta number of elements in next subdivision)

However, it can be more complex particularly if size of records are >13. In fact it is not clear what additional information has been added. Again, then element indicator, doesn't seem to follow the above mentioned rules. As extended types are designed for marine maps, the additional information may include min max depths for each subdivision, as in the DEM subfile.

#### example 2

| 00000000000000000000000000000000000000                    |
|---|
| 00000000000000000000000000000000000000                    |
| 00000000000000000000000000000000000000                    |
| 00000000000000000000000000000000000000                    |
| 0000000050200000700000006032000024400000C2870100F425020   |
| 00000007C040000060000006032000024400000C2870100F425020    |
| 0000000DD0400000400000006032000024400000C2870100F425020   |
| 0000000E50500000700000006032000024400000C2870100F425020   |
| 0000000D80800000400000006032000024400000C2870100F425020   |
| 00000004F0900000500000006032000024400000C2870100F425020   |
| 0000000C40900000500000006032000024400000C2870100F425020   |
| 0000000060A00000400000006032000024400000C2870100F425020   |
| 00000003B0A00000400000006032000024400000C2870100F425020   |
| 0000000580A00000400000006032000024400000C2870100F425020   |
| 0000000580A0000050000006032000024400000C2870100F425020    |
| 0000000770A00000F00000006032000024400000C2870100F425020   |
| 0000000EB0C00000D0000006032000024400000C2870100F425020    |
| 0000000660D0000000000006032000024400000C2870100F425020    |
| 0000000660D000040000006032000024400000C2870100F425020     |
| 9A030000BA0D0000669020000A2320000CE4E0000358A0100F625020  |
| 24070000270E000007B6040000DE320000ED5400003D930100F625020 |
| 780A00004E10000004F6060000383400002A6900007B980100FA25020 |

## TRE8

This block located at TRE + &h8a contains all the extended types and their draworder found in a particular IMG file. Each element data can be 3 bytes or 4 bytes long.

Example: &10F08

| Typ mod &100 | draworder | subtype | ? |
|--------------|-----------|---------|---|
| F            | 2         | 8       | 0 |

An element with the lowest drawnumber seems to be at the highest level, ie the top,

The order in which they are plotted is:

Polylines ,followed by polygons,followed by POIs. Interesting that POIs can have a draworder!

an example of TRE8

## TRE9

Nothings is known about this section; if present, there only seems to be one record of 5 bytes

## **NET subfile**

Here we find additional data concerning routable highways ,such as its length , its direction (if one way),the maximum speed allowed, and its house address information if any.

The header looks like this:

| NET Offset | NET Header                                      |
|------------|---|
| 00         | Header Length                                   |
| 02         | GARMIN NET                                      |
| 15         | Pointer to beginning of NET1                    |
| 19         | Length of this block                            |
| 1D         | Road definitions offset multiplier (power of 2) |
| 1E         | NET2 Segmented Roads                            |
| 22         | Length of this block                            |
| 26         | offset multiplier (power of 2)                  |
| 27         | NET3 Sorted Roads                               |
| 2B         | Length of this block                            |
| <b>2</b> F | Sorted roads record size                        |
|            |   |

#### NET1

We've already discussed how highways can have up to 4 labels. The following table shows a 'typical' road definition entry in NET1; the length of each record varies depending on number of labels, number of subdivisions the highway is plotted and whether it has address information.

| Items                     | Bytes        | example                           |
|---------------------------|--------------|-----------------------------------|
| labels                    | 3 per label  | 12 00 00 , 45 00 00, 24 01 81     |
| Road Data                 | 1            | &44                               |
| Road Length               | 3            | 32 01                             |
| RGN_index_overview        | 1 per record | 1,81 ( 2 records)                 |
| Highway pointer           | 1            | 01 05 , 01 06                     |
| Subdivision number        | 2            |                                   |
| House_number blocks       | 1            | Set if bit 4 or Road Data is set  |
| Street Address info block | varies       |                                   |
| NOD length of pointer     | 1            | 1 or 2                            |
| NOD2 offset               | 2 or 3       | 2 or 3 of NOD length or pointer=2 |

#### Example:

| 37 | 00 | 47 | 41 | 52 | 4D | 49 | 4E | 20 | 4E | 45 | 54 | 01 | 00 | DB | 07 | 7-GARMIN NET -U.                       |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--|
| 09 | 1E | 07 | 15 | 06 | 37 | 00 | 00 | 00 | 24 | 00 | 00 | 00 | 00 | 5B | 00 | •[-                                    |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 5B | 00 | 00 | 00 | 06 | 00 | 00 | 00 | 03 | ·····[·····k                           |
| 00 | 00 | 00 | 00 | 00 | 01 | 00 | 60 | 00 | 80 | 44 | 06 | 00 | 00 | 01 | 81 | ····· •• • • • • • • • • • • • • • • • |
| 01 | 06 | 00 | 01 | 05 | 00 | 01 | 00 | 00 | 68 | 00 | 80 | 44 | OB | 00 | 00 | h - ED?                                |
| 01 | 81 | 02 | 06 | 00 | 02 | 05 | 00 | 01 | 07 | 00 | 12 | 00 | 00 | 00 | 00 | · · · · · · · · · · · · · · · · · · ·  |

#### Image shows 2 highways in NET1

|                    | Highway 1      | Highway 2     |
|--------------------|----------------|---------------|
| LBL                | 60 00 80       | 68 00 80      |
| Data               | 44             | 44            |
| Length of Road     | 06 00 00       | 0B 00 00      |
| RGN                | 01 81          | 01 81         |
| Record 1           | 01 at subdiv 6 | 2 at subdiv 6 |
| Record 2           | 01 at subdiv 5 | 2 at subdiv 5 |
| Length of pointers | 1 ie 2         | 1             |
| Pointer to NOD 2   | 00 00          | 07 00         |

Notice how 3 byte records 1 and 2 follow each other after 00 81

Data 44 means : has NOD info and has bit 2 set  $% \left( {{{\rm{A}}} \right)$  , see Mechalas Route definitions

In NOD 2 at offset 00 00 I find: a seven byte record

17 00 00 00 02 00 03 and

17 22 00 00 03 00 07

The first byte refers to the speed class and road type – see Mechalas

&17 is &10 + &7; the when translated gives me a max speed of 35 mph with a road type=1

Te next three bytes are offsets from NOD1 , ie 00 00 00 and 22 00 00  $\,$ 

This is followed by 3 bytes giving information about the number of routing nodes in a highway.

## Length of highways

Unlike stated by Mechalas ,you double the value to obtain length in metres. So the first hw was 06 00 00 which gives you a value of 12 metres and the second, 0B 00 00, produces a value of 22 metres. These values together with the max speed can be used to calculate ETA.

## NOD subfile

Mechalas offers some valuable information regarding the NOD subfile, but unfortunately a lot of its structure remains unclear. The NOD subfile is as the name implies about nodes and how they are linked ;it only exists if the IMG is routable. I am grateful to Robert Vollment for additional pointers regarding the NOD file structure although my findings differ in many respects.

## NOD 1

NOD 1 contains information about nodes, linked directly or indirectly. Some of it still seems unclear .

| NOD 1 | Node records |                               |
|-------|--------------|-------------------------------|
|       | Block 1      |                               |
|       | Table Header | Details of size of each table |
|       | Table A      | Road segments                 |
|       | Table B      | Inter area links              |
|       | Table C      | restrictions                  |
|       | Block 2 ,3,4 |                               |
|       | etc          |                               |
| NODA  |              |                               |
| NOD 2 |              |                               |
| NOD 3 |              | Boundary nodes                |

The structure is quite complex; within NOD1 there are , depending on the number of nodes, various 'green' blocks, following each other .

Each records block is often terminated by so called boundary nodes – these are used as links to other IMGs.

Finding the length of each record can be quite challenging.

We can ascertain where records start from offsets found in NOD 2 and NOD 3.

The header of each record begins with a pointer to other tables, ie A, B or C.

| pointer | Flags | coordinates | coordinates | linked  | Current | Flags A | Flags B |  |
|---------|-------|-------------|-------------|---------|---------|---------|---------|--|
|         | -     |             |             | highway | highway | -       | -       |  |
|         |       |             |             | nodes   | nodes   |         |         |  |
| 02      | 44    | 2E 70 FE    | 48 12 17    | 05      | 07      | 6D      | 29      |  |

## Pointer

The first byte points to the Tables Header – see below.

### Flags at offset 1

There are several flags which are set to indicate special conditions

| Mask   | Purpose                                 |
|--------|---|
| 0 x 4  |   |
| 0 x 8  | Marking a boundary                      |
| 0 x 10 | Marking a restriction                   |
| 0 x 20 | 2 byte coordinates offsets instead of 3 |
| 0 x 40 | Direct links                            |
| 0 x 45 |   |
| 0 x 50 |   |

Any combination is possible but most frequently encountered are 44 or 4C

44=40+44C=40+4+8, ie boundary nodes as found in NOD3

## **Direction Coordinates**

These could be 2 or 3 bytes depending on Flag 0 x 10

## **Nodes Bytes**

The two bytes after the coordinates indicate number of nodes in a highway as a multiple of 2 (?) The first nodes byte contains the number of nodes ( as a multiple) of a linked highway ,ie (5+1)/2=3

The second contains number of nodes in current highway: (7+1)/2 = 4

## Flags A & B

Flag A is one byte and Flag B can be 2 bytes. Both also contain information showing bearings between nodes.

| Flag A |                   |
|--------|-------------------|
| 0 x 7  | Destination Class |
| 0 x 38 |                   |
| 0 x 40 | Going Forward     |
| 0 x 80 | New Direction     |

| Flag B |                 |
|--------|-----------------|
| 0 x 40 | Inter area link |
| 0 x 80 | Last link       |

### **Tables Header**

There can be several tables headers within NOD1.

An offset to a tables header is NOT found in the NOD header; instead it has to be calculated. Presumably, this is because of the overwhelming number of nodes an IMG may contain. Strangely each new 'green' block, except for the first one starts with a tables header.

To calculate the start of a tables header you need to add &40 to the end of a previous records block and then find the nearest multiple of &40

Example: end of node records block :1C76

1C76 + 40 = 1CB6

The next multiple of &40 is: 1CC0 so it starts at 1CC0

This is a 9 byte header:

| 00          | 01 | 02 | 03 | 04         | 05 | Table A | Table B | Table C |
|-------------|----|----|----|------------|----|---------|---------|---------|
| coordinates |    |    | C  | Coordinate | S  | number  | number  | number  |

Number indicates number of records found in each table. Because each table contains records of a fixed length we can calculate its total length and thus the beginning of the next node records block, if any

### Table A

| 0x00 | 3 | Bits 0-29: Pointer to NET; bit 30: no delivery; bit 31: no emergency   |
|------|---|--|
| 0x03 | 1 | Road class : bits 0-3: road speed; 4: oneway; 5-6: road class; 8: toll |
| 0x04 | 1 | RoadID   |

Each record has fixed length of 5 bytes:

| 00 | 00 | 13 | 00 | 12 | RoadID | = | 18 |
|----|----|----|----|----|--------|---|----|
| 00 | 00 | 13 | 00 | 20 | RoadID | = | 32 |
| 00 | 00 | 13 | 00 | 2C | RoadID | = | 44 |
| 01 | 00 | 13 | 00 | ЗE |        |   |    |
| 01 | 00 | 13 | 00 | FЗ |        |   |    |
| 00 | 00 | 03 | 00 | 32 |        |   |    |
| 00 | 00 | 11 | 00 | 44 |        |   |    |
|    |    |    |    |    |        |   |    |

## *NOD 2*

It starts at NOD + &25 with length:NOD + &29 It does not appear to be accessed from any subfile.

| 0                      | 1          | 2      | 3 | 4                 | 5     | 6+             |
|------------------------|------------|--------|---|-------------------|-------|----------------|
| Road<br>classification | Offset int | o NOD1 |   | <mark>mask</mark> | Mask? | Node<br>bitmap |

Its length is generally 7 bytes but depends on the first and 'blue' mask byte. If the mask's value is >8 then an extra byte is added for every multiple of 8. In addition, if bit 8 of the first byte is set, extra bytes highlighted in grey are added– see below.

Examples:

| 17               | 00                | 00         | 00 | 02              | 00 | 03 |    |    |    |    |    |    |    |    |
|------------------|-------------------|------------|----|-----------------|----|----|----|----|----|----|----|----|----|----|
| 03               | <mark>25</mark>   | 59         | DB | 01              | 00 | 04 |    |    |    |    |    |    |    |    |
| 25               | <mark>87</mark>   | 00         | 00 | 8 0             | 00 | FF |    |    |    |    |    |    |    |    |
| <mark>8</mark> 8 | 5C                | 01         | 00 | <mark>08</mark> | 00 | FF | 04 | 14 |    |    |    |    |    |    |
| 87               | 1D                | 2D         | 03 | <mark>0в</mark> | 00 | FF | 06 | 04 | 12 |    |    |    |    |    |
| BB               | <mark>в0</mark>   | 5B         | 00 | 14              | 00 | E3 | F7 | 0F | 0C | 09 | 40 | 00 | 0D | 0E |
| 25               | 82                | 61         | 00 | 09              | 00 | FF | 01 |    |    |    |    |    |    |    |
| 03               | <mark>.3</mark> ਜ | <b>1</b> A | 00 | 12              | 00 | FF | FF | 03 |    |    |    |    |    |    |

For road classification see Mechalas .

Notice how offsets into NOD1 can show masks in the  $3^{rd}$  byte, ie &DB - their significance is uncertain.

Byte 4 acts as a mask for byte 5 or 6, ie gives you the number of bits to consider when examining byte 5 (and?) or 6.

In our first example byte 6 contains 0x03 which in bits from LSB looks like 11000000.

The mask value (2) makes us consider only the first 2 bits, which could imply that this highway has at least 2 nodes connecting to other highways, both of them set.

05 00 1B : 11011000

This would tell us to count the first 5 bits; the idea is that a node will be ignored if a bit is not set, ie 0, so we skip node 3.

If this is true then the maximum number of nodes can only be 8; to overcome this an extra byte is added for each additional multiple of 8 – see last 2 examples

Exploring IMG Format

At present, the function of byte 5 is unknown but we surmise that the mask is 2 bytes long to allow for values >255

| 0                      | 1        | 2       | 3  | 4    | 5     | 6+             |              |  |
|------------------------|----------|---------|----|------|-------|----------------|--------------|--|
| Road<br>classification | Offset : | into NO | D1 | mask | Mask? | Node<br>bitmap | 04,08<br>,0C |  |

If  $8^{th}$  bit is set then the value after 0C signifies the length of extra bytes needed using a simple algorithm: extra bytes = (value-1)/2

Example 0C 13  $\rightarrow$  (13 -1)/2 = 9

| 9B 13 08 00 13 00 7F FB 07 | 0C <mark>09</mark> 42 00 | <mark>28 30</mark>   |
|----------------------------|--------------------------|----------------------|
| A5 82 BE 01 17 00 F5 FF 7B | 0C <mark>13</mark> 41 00 | 15 16 17 18 19 1D 1E |

The 'red' bytes are always ordered according to size. It is not clear what they mean.

## **DEM subfile**

Elevation data is found and plotted in the DEM subfile. For more information see: *Exploring\_DEM.pdf* 



**DEM Investigator** is a GUI showing all known values including base heights for each tile.

| Pite   |       |        |     |     |      |     |     |     |     |     |     |        |     |     |     |      |     |     |       |
|--|-------|--------|-----|-----|------|-----|-----|-----|-----|-----|-----|--------|-----|-----|-----|------|-----|-----|-------|
| DEM33.444160 DEM.3.377000  |       |        |     |     |      |     |     |     |     |     |     |        |     |     |     |      |     |     |       |
| 444160 index 0   | -     |        | 0   | 1   | 2    | 3   | 4   | 5   | 6   | 7   |     | 9      | A   |     | С   | p    | .8  | F   |       |
| 444167 pixels m omin 64  | 100   | 371000 | 25  | 0.0 | 47   | 41  | 8.2 | 40  | 49  | 48  | 20  | 44     | 45  | 40. | 61  | 60   | 05  | 87  | 8-015 |
| stelen pixels y exis se  |       | 377010 | 00  | 08  | 1.1  | 14  | 18  | 11  | 91  | 30  | -01 | 04     | 60  | 20  | 0.0 | 60   | 80  | 3.0 | 225+0 |
| 444148 7 83  |       | 377020 | 00  | 40  | 21   | 00  | 00  | 00  | 00  | 00  | 12  | 0.2    | TE  | 0.0 | 00  | 34   | 02  | -00 | - 81  |
| 444160 7 1   |       | 377630 | 10  | 01  | 35   | 00  | 0.0 | 14  | 65  | 00  | EA  | 61     | 65  | 20  | 0.0 | 31   | 0.9 | 00  | đ +   |
| 444160 % tilms 27  |       | 377040 | 00  | 02  | 4.5  | 00  |     | 2.8 | 80  | 00  | 18  | 62     | 8.2 | 00  | 0.0 | 18   | 10  | 00  | P.R   |
| 444160 size of record in dal 0   |       | 377058 | 70  | 0.2 | 46   | 00  | 88  | 18  | 10  | 00  | 3.5 | 62     | 60  | 100 | 0.0 | 0.00 | 12  | 0.0 | In F  |
| 444160 dal starts 377025   |       | 277040 | 24  | 0.2 | 4.0  | 0.0 | 00  | 35  | 15  | 00  | 90  | 0.0    | 14  | 00  | 00  | 3.8  | 17  | 00  | THI-  |
| 444163 wont:0<br>444180 month 407291296  |       | 377070 | 47  | 02  | EA.  | 00  | 00  | 18  | 18  | 00  | C1. | 02     | 41  | 00  | 00  | 5.7  | 10  | 00  | 018-  |
| #44190 distance between pixels NS 1648   |       | 317065 | 4.3 | 02  | 79   | 80  | 00  | 28  | 18  | 00  | 47  | 62     | 06  | 00  | 50  | 1.7  | 22  | 0.0 | 61.01 |
| 444194 distance between pixels EV:1648   |       | 377090 | C0  | 01  | 28   | 01  | 00  | 42  | 24  | 00  | 115 | 85     | 48  | 105 | 00  | C7   | 24  | 0.0 | A I   |
| 444178 min height 0  |       | 377080 | 4.0 | 01  | 5.0  | 01  | 00  | 85  | 20  | :00 | 1.0 | 65     | 1.0 | 01  | 00  | 82   | 31  | -00 |       |
| stars and mergins rate.  | -     | 377080 | 45  | 01  | 87   | 01  | 00  | 21  | 38. | 00  | 5.9 | 01     | 63  | .01 | 00  | 17   | 38  | -00 |       |
| 41 41 41   |       | 377903 | 58  | 01  | 30   | 81  | 00  | 14  | 10  | 60  | 29  | 01     | 32  | 111 | 60  | 0.5  | 41  | 00  | X B   |
|  | -     | 3770D0 | 18  | 01  | AA   | 01  | 00  | 16  | 45  | 0.0 | 16  | 01     | 11  | 01  | 00  | 11   | 48  | 0.0 | 1. *  |
| period of the second se | -     | 377050 | C9  | 00. | 44   | 01  | 00  | 0.5 | 40  | 00  | 80  | 00     | 12  | 00  | 00  | CC.  | 42  | 00  | 2.2   |
|  | 100   | 377989 | CS  | 00  | 45   | 01  | 00  | 90  | 52  | 80  | 2.9 | 00     | 34  | 01  | 00  | 10   | 55  | -00 | 3.1   |
| (1) · · · · · · · · · · · · · · · · · · ·  | 1000  | 377100 | 0.6 | 01  | 68   | 41  | 60  | 45  | 14  | 00  | 84  | 05     | 17  | :00 | -00 | 85   | 10  | 00  | - 0   |
| <b>놰</b> 껆内 <b>뇄뭱킕됕놣닅</b> 뭱붱랞녎뷤뼟둯相뉑셷벑뮫닅止뵹귿   | 1000  | 277110 | 00  | 01  | 95   | 90  | 00  | 48  | 61  | 00  | 00  | 02     | 67  | 20  | 50  | D2   | 63  | 00  | 0     |
|  | 1000  | 377120 | 1.0 | 02  | 85   | 00  | 00  | 68  | 64  | .00 | 10  | 0.2    | 42  | 00  | 00  | 45   | 69  | 00  | 41    |
| and the second se  |       | 377130 | 58  | 02  | 64   | 00  | 00  | 90  | 62  | 00  | 19  | 62     | 3.5 | 00  | 00  | 14   | 62  | 00  | nd-   |
|  | 1.11  | 377145 | 28  | 01  | 85   | 00  | 00  | 54  | 71  | 00  | -04 | 02     | 00  | 00  | 00  | 17   | 15  | 00  | 4 4   |
|  |       | 377150 | 50  | 02  | A2   | 80  | 00  | 58  | 78  | 00  | 90  | 02     |     | 88  | 00  | 18   | 18  | -00 | Pro   |
|  |       | 377160 | 34  | 02  | 27   | 00  | 00  | 75  | 78  | 00  | 20  | 02     | 82  | 00  | 00  | 18   | 85  | :00 | 412   |
|  |       | 377170 | 42  | 02  | 44   | 00  | 00  | 32  | 84  | 00  | 8.9 | 02     | 48  | 00  | 00  | 5.9  | 87  | 00  | 018-  |
|  |       | 377180 | FD  | 01  | 10   | 01  | 00  | 04  | #8  | 00  | 38  | 01     | 14  | 01  | 00  | 1.8  | 12  | 0.0 | 2     |
|  |       | 377190 | 47  | 01  | 58   | 01  | 88  | 51  | 92  | 00  | 16  | 01     | -04 | 01  | 00  | 82   | 14  | 00  | G X   |
|  |       | 3771A0 | 16  | 01  | 78   | 81  | 00  | 68  | 38  | 00  | TC  | 01     | 17  | 81  | 00  | 17   | 32  | 00  | 7 2   |
|  |       | 377180 | 38  | 02  | CD   | 80  | 00  | 77  | A1  | 00  | 6.9 | 20     | 39  | 01  | 00  | 78   | 3.5 | -00 | Bil-  |
|  |       | 377100 | 0.0 | 0.0 | 42   | 01  | 00  | 1.6 | 3.5 | 00  | 34  | 60     | 05  | 00  | 60  | 02.  | AC. | 60  | 0.0   |
|  |       | 377100 | 54  | 00  | 78   | 01  | 00  | BA. | AF  | 00  | 80  | 01     | 75  | 01  | 00  | 40   | 81  | .00 | 1.1   |
|  |       | 3771E0 | 10  | 02  | 2.9  | 00  | 00  | 29  | 37  | 00  | CB  | 01     | 87  | .00 | 00  | 11   | 24  | 00  | 14    |
|  |       | 377120 | 0¢  | 02  | 87   | 00  | 00  | 80  | 80  | 00  | 38  | 02     | 62  | 00  | 00  | 15   | 00  | 00  | F14-  |
|  |       | 377200 | 27  | 02  | . 81 | 30  | 00  | 67  | 02  | .00 | 10  | 02     | 83  | 00  | 00  | 68   | CS  | .00 | 110-1 |
|  |       | 377210 | 00  | 01. | CT.  | 00  | 00  | 43  | CE  | 00  | 35  | 01     | 82  | 00  | 00  | 18   | CB  | -00 | t c   |
|  |       | 377220 | BA  | 01  | 81   | 00  | 00  | 32  | CF  | 00  | 00  | 02     | 34  | 00  | 00  | 12   | 02  | 00  |       |
|  |       | 377230 | 0C  | 02  | AB   | 50  | 0.0 | 63  | 05  | 00  | EC. | 01     | D7  | 00  | 00  | CT   | Dt  | 00  | +1+-  |
|  | 11.12 | 377240 | BA  | 05  | 1.9  | 01  | 00  | CD  | DC. | 00  | 87  | 01     | 17  | 00  | 00  | 60   | 20  | 00  | * }   |
|  |       |        |     |     |      |     |     |     |     |     |     | 1. Lui |     |     |     |      |     |     |       |

Exploring IMG Format

## **Creating IMG files**

There are several ways of creating an IMG file, each with its own personality:

- 1) using cgpsmapper
- 2) using MapTK
- 3) using mkgmap.jar

It all started with cgpsmapper, still in many ways the King , making full use of all the bitstream parsing options, but unable to cope with extended elements and latest developments. It uses 8 bit LBL encoding.

MapTK is quite a remarkable piece of software ; it produces IMG files from a text file and handles extended elements, using 8 bit LBL encoding. It is struggling to keep up to date but its IMG files are almost text book, 'ganz gründlich' ie methodical.

By far the 'neatest' ,using fewer subdivisions and most up to date, is mkgmap created by a team of programmers ; its IMGs are a delight to parse and highly recommended. 6 bit LBL encoding is used.

None of them can parse DEM subfiles.

## **NT POIs**

The structure of NT pois is basically understood and similar to non NT pois. The only difference is the data streams following the LBL / lat/lon blocks.

Such data streams *do not have fixed lengths* and Garmin is employing several tricks to define the various lengths - more information on

www.pinns.co.uk/osm/garmin.html.

## Extra POI data stream

This consists of 3 or more bytes containing the ID as found in a TYP file - the maximum ID seems to be &FFFF.

Each of the 3 bytes include masks the purpose of which is not clear. This also makes it very hard to establish the extra poi ID.

IMG2TYP can display all IDs



bitstream, 28, 32, 33, 34, 36 cgpsmapper, 13, 36 Coordinates, 27 DEM, 1, 49 extended types, 17, 23 extra poi, 51 FID, 39 Garmin, 1 GMT, 5 header, 9 IMG, 1, 5, 7, 10 IMG Explorer, 5 IMG2TYP, 2, 6, 51 LBL, 18 left\_shift, 35 Locked, 17

MapTK, 50 Mechalas, 5, 16, 21, 22 mkgmap, 50 NET, 1, 7, 18, 20, 21, 24, 30, 42 NOD, 7, 21, 30, 42, 43, 44, 46, 47 NT pois, 50 PID, 39 POI, 17, 18, 20, 22 polygons, 7, 8, 10, 12, 16, 24, 36 Polygons, 10, 26 POLYLINES, 24 RGN, 1, 7, 10, 11, 12 Subdivisions, 9, 13 TRE, 1, 10 TRE8, 41 types 0 x 100+, 25, 26