Garmin Image (IMG) Subfiles

File Format

Herbert Oppmann

herby@memotech.franken.de

http://www.memotech.franken.de/FileFormats/

2022-05-17

Content

Garmin Image (IMG) Subfiles File Format 5
Basic data types 5
Common basic structure 6
DEM file format 8
Header 8
Section 1 8
GMP file format 10
Header 10
IDX file format 11
Header 11
LBL file format 12
Header 12
MAR file format 16
MD2 file format 17
Header 17
Section 1 18
Section 2 18
Section 3 18
Section 4 18
Section 5 18
Section 6 18
MDR file format 19
MET file format 20
Header 20
Section 1 21
Section 2 21
Section 3 21
Section 4 21
Section 5 (overlaps with Section 9 of TRE in GMP file) 21
MPS file format 22
General record structure 22
Record 'D': 22
Record 'F': Product 23
Record 'L': Map segment 23
Record 'M': used in BluChart Marine charts 23
Record 'P': 23
Record 'R': Route – not seen 23
Record 'T': Track – not seen 23
Record 'U': Unlock 23
Record 'V': Mapset name 24

Record 'W': Waypoint – not seen 24
NET file format 25
NOD file format 26
Header 26
Section 1 27
Section 2 27
Section 3 27
Section 4 27
Section 5 27
Section 6 27
QSI file format 28
RGN file format 29
Header 29
Section 1 29
Section 2 30
Section 3 30
Section 4 30
Section 5 30
S16 file format 31
SNR file format 32
SRT file format 33
TRE file format 34
Header 34
Section 1 37
Section 2 37
Section 3 37
Section 4 37
TRF file format 38
Header 38
Section 1 38
Section 2 38
Section 3 38
Section 4 38
TYP file format 40
Section 1 42
Section 2 45
Section 3 46
Section 4 47
Section 5 47
Section 6 48
Section 7 48
Section 8 48

Section 9 49
Section 10 49
Section 11 49
Section 12 50
Section 13 50
Section 14 50
References 51
Used sources of information 51
Standards and specifications 51
Sources of sample files 51

Garmin Image (IMG) Subfiles File Format

This document contains info about all files contained within IMG files. For the IMG/ADM/TTS Container File Format, there is a separate document available.

All subfiles which belong together have the same name and only differ in the file extension.

File types (determined by file name extension):

DEM Digital Elevation Model

GMP Subtile of a larger area

IDX ?

LBL Labels for map elements, city names, localities, etc.

MAR ? marine? (In bluechart 2 nautical charts) See GMP file format below.

MD2 ?

MDR Binding individual map files together.

MET ?

MPS Similar to the TDB file in that it is a list of maps along with other information describing the set.

NET Road network information (intersections, etc.)

NOD Routing information.

QSI ?

RGN Map elements such as polylines, polygons and points.

S16 ?

SNR ?

SRT Lookup table for characters

TBD See separate document for this file format.

TRE Map structure information that organizes the map elements into a data tree.

TRF ?

TYP Custom rendering styles. How a map element will be rendered on the GPS which allows you to display things in a different way. Appears only once in an IMG file.

This documentation is based on own research and the sources listed in the references section.

Text highlighted like this is where further investigation is needed.

Basic data types

All values are serialized in little-endian byte order (least significant byte first).

Type	Length	Description
byte	1	8 bit unsigned integer (range 0 .. 255)

Type	Length	Description
ushort	2	16 bit unsigned integer (range 0 .. 65535)
uint	4	32 bit unsigned integer (range 0 .. 4294967295)
ulong	8	64 bit unsigned integer (range 0 .. 18446744073709551615)
short	2	16 bit signed integer (range -32768 .. 32767)
int	4	32 bit signed integer (range -2147483648 .. 2147483647)
long	8	64 bit signed integer (range -9223372036854775808 .. 9223372036854775807)
double	8	64 bit floating point value according to [15].
char	1	A byte, which is a character code according to [16].
string	1..n	0-terminated string (may be empty).

Color

Structure, consisting of three byte: R (for red), G (for green) and B (for blue).

VarInt

An unsigned integer in a variable-sized encoding

Using 1 byte: (probably, but not seen yet)

	7	6	5	4	3	2	1	0
0	6	5	4	3	2	1	0	=1

Contains 7 bit. Used for numbers in the range 0 .. 0x7F.

Using 2 bytes:

	7	6	5	4	3	2	1	0
0	5	4	3	2	1	0	=1	=0

	7	6	5	4	3	2	1	0
1	13	12	11	10	9	8	7	6

Contains 14 bit. Used for numbers in the range 0x0080 .. 0x3FFF.

Common basic structure

Most of the files contained within an IMG file start with a header that has some fields in common.

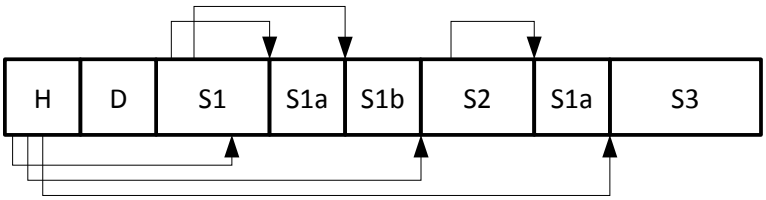
Offset	Type	Content
0x00	ushort	Header length (21 + file type specific header data)
0x02	char[10]	"GARMIN " + file name extension, e.g. "TRE"
0x0C	byte	? Values seen: 1, 2, 3
0x0D	byte	Locked, 0=no, 0x80=yes
Creation date time just like in IMG file:		
0x0E	ushort	Creation year Sometimes, you encounter values < 100, then add 2000. Else if value < 200 add 1900. Else it should be the correct 4-digit year.
0x10	byte	Creation month, 1..12 (some tools use 0..11)
0x11	byte	Creation day, 1-31

Offset	Type	Content
0x12	byte	Creation hour, 0-23
0x13	byte	Creation minute, 0-59
0x14	byte	Creation second, 0-59
0x15	file type specific data follow	

The file type specific header data have been extended several times over time. This has been done in a backward compatible way, i.e. new fields have been added at the end, and all previously defined fields are left unchanged.

Within the files that have this common header, the content is organized in sections. Info about the sections is contained in the file type specific header data. The section info contains a reference to the section and the length. For all references, the offset from the beginning of the file is used. If the section consists of records of equal size, then the section info in addition contains the size of each record.

Example layout of common basic structure:



H = Header, including the common header

D = Description, optionally between the header and the first (sub)section

S1, S2, S3 = Sections referenced from header

S1a, S1b, S2a = Subsections, referenced from section

Often, the sections are in order, and the subsections fill the space between the sections. But there are counter-examples. This means, whether a description is present and how long it is can only be determined as the last step, after all references have be followed.

DEM file format

Digital Elevation Model. See [7], which has much more info on that!

This file follows the common basic structure.

Header

Header length: 37

Offset	Type	Content
0x15	uint	Flags Bit 0: Unit, 0=meter, 1=feet Seen bits: 3,4,5,6,7,8,9,10,11,12,13,15,16,17,18,19,20,21,22,24,26
0x19	ushort	Number of zoom levels (can be different from the number of map levels)
0x1B	uint	=0
0x1F	ushort	Record size in section 1 (=60)
0x21	uint	Offset of section 1

Header length: 41

Offset	Type	Content
0x25	uint	=0

Section 1

Offset	Type	Content
0x00	ushort	index of the record (starting with 0)
0x02	uint	number of pixel/tile (x-axis)
0x06	uint	number of pixel/tile (y-axis)
0x0A	uint	
0x0E	uint	
0x12	ushort	? =0
0x14	uint	Number of tiles in x direction – 1 (columns)
0x18	uint	Number of tiles in y direction – 1 (rows)
0x1C	ushort	Describe the structure of records in data block 1. The lowest two bits are the "offset size": 00 = 1 byte; 01 = 2 byte; 10 = 3 byte; 11 = 4 byte (last was not seen so far) The third bit defines size of "base height" field. If set this field is 2 bytes long, otherwise just 1. The forth bit defines size of "height difference" field. If set this field is 2 bytes long, otherwise just 1. If the fifth bit is set there will be an extra byte in the records.
0x1E	ushort	Record size in subsection 1 (up to 8)
0x20	uint	Offset of subsection 1
0x24	uint	Offset of subsection 2
0x28	uint	Western boundary
0x2C	uint	Northern boundary
0x30	uint	Distance between pixel (n-s direction)?

Offset	Type	Content
0x34	uint	Distance between pixel (w-e direction)?
0x38	ushort	minimum height. This is the minimum of all base heigth given in records of data block 1.
0x3A	ushort	maximum height. There is at least one record where base height + height difference = maximum height.

GMP file format

This file follows the common basic structure. It combines the contents of TRE, RGN, LBL, NET, NOD, DEM and MAR files into a single file. This helps keeping together all information which belong together.

Important: This file is not just a bundling of otherwise separate files. It is a single file that just contains all the content, but the different headers and sections may be placed anywhere, in any order, within the GMP file. All offsets used within the GMP file are based on the beginning of the GMP file. So if you want to extract the TRE info into a separate file, you would have to follow all the offsets, extract all sections and subsections, and recalculate all offsets.

Header

Header length: 49

Offset	Type	Content
0x15	uint	Unknown offset, only 0 seen
0x19	uint	Offset to TRE header
0x1D	uint	Offset to RGN header
0x21	uint	Offset to LBL header
0x25	uint	Offset to NET header
0x29	uint	Offset to NOD header
0x2D	uint	Offset to DEM header. Is 0 if not present.

Header length: 53

Offset	Type	Content
0x31	uint	Offset to MAR header. Is 0 if not present.

Header length: 57

Offset	Type	Content
0x35	uint	Unknown offset, only 0 seen

Header length: 61

Offset	Type	Content
0x39	uint	Offset to MET header. Is 0 if not present.

IDX file format

This file does **not** follow the common basic structure.

Header

Offset	Type	Content
0x00	uint	Length of header (=52)
0x04	char[22]	File name, e.g. "ISM.IDX", terminated with a 0 byte. Unused bytes at the end filled with 0.
0x1A	ushort	Year
0x1C	byte	Month, 1..12
0x1D	byte	Day, 1..31
0x1E	byte	Hour, 0..23
0x1F	byte	Minute, 0..59
0x20	byte	Second, 0..59
0x21	ushort	=0
0x23	uint	=2
0x27	uint	=0
0x2B	uint	=20
0x2F	uint	=24
0x33	byte	=0

LBL file format

This file follows the common basic structure.

Header

Header length: 170

Offset	Type	Content
0x15	uint	Offset of section 1
0x19	uint	Length of section 1
0x1D	byte	Data label offset multiplier. Values seen: 0, 1 and 3.
0x1E	byte	Label coding. 6 = 6 bit, 9 = 8 bit(!), 10 = 10 bit, 11 = ???
0x1F	uint	Offset of section 2 – Countries
0x23	uint	Length of section 2
0x27	ushort	Record size in section 2. Values seen: 3, 4, 5 (0 is also OK when the length of section is 0.)
0x29	uint	Flags for section 2. Values seen: 0, 1, 3
0x2D	uint	Offset of section 3 – Regions
0x31	uint	Length of section 3
0x35	ushort	Record size in section 3. Values seen: 5, 6
0x37	uint	Flags for section 3. Values seen: 0, 1
0x3B	uint	Offset of section 4 – Cities
0x3F	uint	Length of section 4
0x43	ushort	Record size in section 4. Values seen: 5, 7, 8 (0 is also OK when the length of section is 0.)
0x45	uint	Flags for section 4. Values seen: 0, 1, 3, 0x80000000, 0x80000001
0x49	uint	Offset of section 5 – POI index
0x4D	uint	Length of section 5
0x51	ushort	Record size in section 5. Values seen: 4 (0 is also OK when the length of section is 0.)
0x53	uint	Flags for section 5. Values seen: 0
0x57	uint	Offset of section 6 – POI properties
0x5B	uint	Length of section 6
0x5F	byte	Offset multiplier. Values seen: 0, 1 and 2
0x60	byte	Global mask. Values seen: 0, 0x17
0x61	ushort	Values seen: 0, 8, 0x0B08, 0x0BC8, 0x0BC9, 0x2BC8, 0x3B88, 0x3BC8, 0x4B88, 0x8B09, ..
0x63	byte	Value seen: 0, 1, 4, 5
0x64	uint	Offset of section 7 – POI types
0x68	uint	Length of section 7
0x6C	ushort	Record size in section 7. Values seen: 4, 5 (0 is also OK when the length of section is 0.)
0x6E	uint	Flags for section 7. Values seen: 0
0x72	uint	Offset of section 8 – ZIPs
0x76	uint	Length of section 8

Offset	Type	Content
0x7A	ushort	Record size in section 8. Values seen: 3
0x7C	uint	Flags for section 8. Values seen: 0
0x80	uint	Offset of section 9 – Highways
0x84	uint	Length of section 9
0x88	ushort	Record size in section 9. Values seen: 6
0x8A	uint	Flags for section 9. Values seen: 0
0x8E	uint	Offset of section 10 – Exits
0x92	uint	Length of section 10
0x96	ushort	Record size in section 10. Values seen: 4, 5
0x98	uint	Flags for section 10. Values seen: 0, 1, 2, 5
0x9C	uint	Offset of section 11 – Highway data
0xA0	uint	Length of section 11
0xA4	ushort	Record size in section 11. Values seen: 3
0xA6	uint	Flags for section 11. Values seen: 0

Header length: 196

Offset	Type	Content
0xAA	ushort	Codepage. Values seen: 0x04E4 Western Europe (Windows), see [5] and [6].
0xAC	ushort	Product ID / Family ID? Values seen: 0, 7 (sometimes, Codepage =0 and there is a code page at this offset, but this is probably a bug in the generating program)
0xAE	ushort	Product ID / Family ID? Values seen: 0, 0x8002
0xB0	uint	Offset of section 12 – Sort descriptor
0xB4	uint	Length of section 12
0xB8	uint	Offset of section 13 – ??
0xBC	uint	Length of section 13
0xC0	ushort	Record size in section 13
0xC2	ushort	=0

Header length: 208

Offset	Type	Content
0xC4	uint	Offset of section 14 – Tide prediction
0xC8	uint	Length of section 14
0xCC	ushort	Record size in section 14?
0xCE	ushort	=0

Header length: 236

Offset	Type	Content
0xD0	uint	Offset of section 15 – ??
0xD4	uint	Length of section 15
0xD8	ushort	Record size in section 15
0xDA	uint	=0
0xDE	uint	Offset of section 16 – ??
0xE2	uint	Length of section 16

Offset	Type	Content
0xE6	ushort	??
0xE8	uint	=0

Header length: 410

Offset	Type	Content
0xEC	uint	Offset of section 17 – ??
0xF0	uint	Length of section 17
0xF4	ushort	Record size in section 17
0xF6	...	

Header length: 422

Offset	Type	Content

Header length: 434

Offset	Type	Content

Header length: 458

Offset	Type	Content

Header length: 498

Offset	Type	Content

Header length: 528

Offset	Type	Content

Header length: 596

Offset	Type	Content

Header length: 618

Offset	Type	Content

Header length: 632

Offset	Type	Content

Header length: 660

Offset	Type	Content

Header length: 661

Offset	Type	Content

Header length: 681

Offset	Type	Content

MAR file format

Not seen yet.

This file most probably also follows the common basic structure, because it can be part of a GMP file.

MD2 file format

This file does **not** follow the common basic structure, but it is similar. The differences are:

- It has 32-bit header length
- Within the section info, the record size is 32-bit
- The record size is always present. If not applicable, the value is 0.

Header

Offset	Type	Content
0x00	uint	Length of header. Known value 236
0x04	ushort	Year
0x06	byte	Month, 1..12
0x07	byte	Day, 1..31
0x08	byte	Hour, 0..23
0x09	byte	Minute, 0..59
0x0A	byte	Second, 0..59
0x0B	uint	Offset section 1
0x0F	uint	Length section 1
0x13	uint	Record size in section 1. Known value 4.
0x17	uint	= 1
0x1B	uint	= 0
0x1F	uint	= 0
0x23	uint	= 0
0x27	uint	Offset section 2
0x2B	uint	Length section 2
0x2F	uint	= 0
0x33	uint	Offset section 3
0x37	uint	Length section 3
0x3B	uint	= 0
0x3F	uint	Offset section 4
0x43	uint	Length section 4 (may be 0 although offset != 0)
0x47	uint	= 0
0x4B	uint	Offset section 5
0x4F	uint	Length section 5
0x53	uint	Record size in section 5. Known value 8.
0x57	uint	= 3
0x5B	char[13]	"000-00000-00" + 0x00
0x68	uint	= 2
0x6C	uint	Offset section 6
0x70	uint	Length section 6
0x74	uint	Record size in section 6. Known value 0 (if length ==0), 7.
0x78	uint	? Values seen 0 (if length ==0), 3.
0x7C	byte[0x70]	=0

Section 1

Each record is 4 byte. uint. Values are in ascending order.

Section 2

?? A lot of it is 0xFF.

Section 3

?? A lot of it is 0xFF.

Section 4

?? Size is 0 in all examples seen so far.

Section 5

Each record is 8 byte. A lot of it is 0xFF.

Section 6

Each record is 7 byte.

Seems to be three ASCII chars (digit or upper case letter) plus an uint. The uints are increasing from one record to the next.

MDR file format

This file follows the common basic structure. TBD

Known header lengths: 286, 426, 568, 668, 708, 764, 772, and 784

MET file format

This file follows the common basic structure.

Header

Header length: 288

Offset	Type	Content
0x15	uint	Offset of section 1
0x19	uint	Length of section 1
0x1D	uint	? 0x00001C03, 0x00001C11, 0x00001C13, 0x00001C51
0x21	uint	? 0x001D0100, 0x001D0601, 0x001F0601
0x25	uint	? 0, 0x000F4240 = 1000000
0x29	uint	Offset of section 2
0x2D	uint	Length of section 2
0x31	uint	Offset of section 3
0x35	uint	Length of section 3
0x39	byte[95]	? E0 07 02 0B 15 25 37 01 00 02 80 09 07 00 1B DE 63 01 00 C8 31 00 AB DA 00 5A 1D 00 05 9B + 65 x 00
0x98	uint	Offset of section 4
0x9C	uint	Length of section 4
0xA0	ushort	Record size in section 4. Known value 4.
0xA2	byte[5]	? 00 00 00 00 + byte 0 or 1
0xA7	uint	Offset of section 5
0xAB	uint	Length of section 5
0xAF	ushort	Record size in section 5. Known value 5.
0xB1	byte[5]	? 01 00 00 06 00
0xB6	string	"LakeVü HD Ultra" or "BlueLink Basemap"+ terminated and filled up with 0x00 bytes
0x11B	byte[5]	01 E0 07 E4 04 or 00 E2 07 E4 04

Header length: 304

Offset	Type	Content
0x120	uint	Offset of section 6
0x124	uint	Length of section 6 (may be 0 although offset != 0)
0x128	uint	Offset of section 7
0x12C	uint	Length of section 7

Header length: 370

Offset	Type	Content
0x130	string	"Light Basemap"+ terminated and filled up with 0x00 bytes
0x162	ushort	0x0013
0x164	string	"006-D6655-00"+ terminated and filled up with 0x00 bytes

0x0172 = 370

Section 1

Offset	Type	Content
0x00	byte[]	? 00 or 01 or 00 4B or 00 41 00 00 26 00 ... - is this really a section?

Section 2

Offset	Type	Content
0x00	string	"GARMIN LTD. AND ITS SUBSIDIARIES" + 0x1F + "2016"
	optional string	"© 2010 Intermap Technologies Inc. All rights reserved." + 0x1F + "2016" or "©1987-2005 HERE. All rights reserved." + 0x1F + "2016" or "U.S. Dept. of Transportation" + 0x1F + "2016" or "©2013 DigitalGlobe" + 0x1F + "2016" or "© 2015 Boat Owners Association of the United States" + 0x1F + "2016"
	optoinal string	"©1987-2005 HERE. All rights reserved." + 0x1F + "2016" or "© Her Majesty the Queen in right of Canada, as represented by the Minister of Fisheries and Oceans, on behalf of the Canadian Hydrographic Service" + 0x1F + "201" + 0x00
	optional string	"©1987-2005 HERE. All rights reserved." + 0x1F + "2016"

Section 3

Offset	Type	Content
0x00	uint	? 0x00000022 or 0x00000012
0x04	optional uint	? 0x00000048 or 0x00000025 (present when second string in Section 2 is present)
0x08	optional uint	? 0x00000036 or 0x00000042 (present when third string in Section 2 is present)
0x0C	optional uint	? 0x0000008E (present when fourth string in Section 2 is present)

Section 4

Offset	Type	Content
0x00	4, 5, 6 or 8 x uint	? ...

Section 5 (overlaps with Section 9 of TRE in GMP file)

Offset	Type	Content
0x00	byte[5]	? 20 C9 58 10 81

MPS file format

This file does **not** follow the common basic structure. Typical names: MAPSOURC.MPS, MAKEGMAP.MPS or BLUCHART.MPS.

The file name extension *.mps stands for "MapSource".

Note: Garmin uses this file format with the program MapSource V2.xx, and within *.img files. Later versions of MapSource, and all versions of BaseCamp and HomePort, use a different file format with the same file name extension. See my other document Garmin_MPS_GDB_and_GFI_Format.pdf.

The file consists of a sequence of variable sized records.

General record structure

All records have this general structure:

Offset	Type	Content
0x00	char	Record type. Identifies the record content. 'D' 'F' Product 'L' Map segment 'M' 'P' 'U' Unlock 'V' Mapset name
0x01	ushort	Record length, excluding the record type and length
0x03	byte[Record length]	Record content

Usual record sequence: L+ P+ V U F+

seen: L+ F+ V, L+ V F, L+ P V F, M+ D L+

The amount of P and F records is almost same and much smaller than the number of L records.

There are a lot of records with the same Product ID and Family ID, but it is not the same throughout the file.

Record 'D':

Offset	Type	Content
0x03		8 byte

Record 'F': Product

Offset	Type	Content
0x03	ushort	Product ID
0x05	ushort	Family ID
0x07	string	Name

Record 'L': Map segment

Offset	Type	Content
0x03	ushort	Product ID
0x05	ushort	Family ID
0x07	uint	Segment ID (usually, but not always, in ascending order)
0x0B	string	Name
	string	Segment name
	string	Area name
len-8	uint	Often, this is the same as the Segment ID, but sometimes different (but not 0)
len-4	uint	=0

Record 'M': used in BluChart Marine charts

Offset	Type	Content
0x03		10 byte

Record 'P':

Offset	Type	Content
0x03	ushort	Product ID
0x05	ushort	Family ID
0x07	ushort	? Known value: 0, 1, 40, 50, 58, 74, 83 and 154. The TRE file UnknownCA seems to be the same value.
0x09	uint	? Known values: 0 and 1

Record 'R': Route – not seen

Record 'T': Track – not seen

Record 'U': Unlock

Offset	Type	Content
0x03	string	Length is 25 characters. Characters are upper case letters or digits.

Record 'V': Mapset name

Offset	Type	Content
0x03	string	Mapset name
len-1	byte	Autoname flag. 0=no, 1=yes

Record 'W': Waypoint – not seen

NET file format

This file follows the common basic structure. TBD

Known header lengths: 55, 67 and 100

NOD file format

This file follows the common basic structure.

Header

Header length: 63

Offset	Type	Content
0x15	uint	Offset of section 1 – Nodes
0x19	uint	Length of section 1
0x1D	ushort	Flags. Bit 0: unknown, but seen. Bit 1: Enable turn restrictions Bit 2-4: ? Bit 5-7: Distance multiplier Bit 8: Drive on left Bit 9: unknown, but seen. Bit 11: unknown, but seen. Bit 13: unknown, but seen. Bit 15: unknown, but seen.
0x1F	ushort	? Values seen: 1, 16, 17, 20, 22, 23, 32, 36, 528, 791, 805, 4901 and many others
0x21	byte	Align. Values seen: 0, 1, 4, 6
0x22	byte	Mult1
0x23	ushort	Table A record length. Values seen: 4, 5
0x25	uint	Offset of section 2 - Road data
0x29	uint	Length of section 2
0x2D	ushort	=0
0x2F	ushort	? Values seen: 0, 14, 28, 30, 34, 38, 40, 76
0x31	uint	Offset of section 3 - Boundary nodes
0x35	uint	Length of section 3
0x39	ushort	Record size in section 3. Values seen: 9, 10
0x3B	uint	?

Header length: 127

Offset	Type	Content
0x3F	uint	Offset of section 4 – High class boundary
0x43	uint	Length of section 4
0x47	uint	Some offset within section 1?
0x4B	uint	Some length?
0x4F	uint	Some offset within section 1?
0x53	uint	Some length?
0x57	uint	?
0x5B	byte[12]	Seen: all 0; 0x8a 0x19 following 0
0x67	uint	Offset of section 5 - ?
0x6B	uint	Length of section 5

Offset	Type	Content
0x6F	ushort	Record size in section 5.
0x71	uint	Offset of section 6 - ?
0x75	uint	Length of section 6
0x79	ushort	Record size in section 6.
0x7B	uint	?

Section 1

TBD

Section 2

TBD

Section 3

TBD

Section 4

TBD

Section 5

TBD

Section 6

TBD

QSI file format

This file does **not** follow the common basic structure. Usual name: QSIINDEX.QSI.

Offset	Type	Content
0x00	char[]	"QSI"
0x		
0x		

RGN file format

This file follows the common basic structure. Important: Garmin has another file format with the same file name extension RGN which is used for firmware update.

Header

Header length: 29

Offset	Type	Content
0x15	uint	Offset of section 1 – Data
0x19	uint	Length of section 1

Header length: 125

Offset	Type	Content
0x1D	uint	Offset of section 2
0x21	uint	Length of section 2
0x25	uint	? Known values: 0, 2
0x29	uint	= 0
0x2D	uint	? Known values: 0, 0x20001FFF
0x31	uint	? Known values: 0, 0x00FFFCFD
0x35	uint	= 0
0x39	uint	Offset of section 3
0x3D	uint	Length of section 3
0x41	uint	= 0
0x45	uint	= 0
0x49	uint	? Known values: 0, 0x200000FF
0x4D	uint	? Known values: 0, 0x0000FFFD
0x51	uint	= 0
0x55	uint	Offset of section 4
0x59	uint	Length of section 4
0x5D	uint	= 0
0x61	uint	= 0
0x65	uint	? Known values: 0, 0x20007FFF
0x69	uint	? Known values: 0, 0x3FFFF73F
0x6D	uint	= 0
0x71	uint	Offset of section 5
0x75	uint	Length of section 5
0x79	uint	? Known values: 0, 0x000000E3, 0x000000E7

Section 1

Seems TRE file content is needed to decode this.

Section 2

TBD

Section 3

TBD

Section 4

TBD

Section 5

TBD

S16 file format

This file does **not** follow the common basic structure.

SNR file format

This file does **not** follow the common basic structure.

SRT file format

This file follows the common basic structure. TBD

Known header lengths: 29, and 37

TRE file format

This file follows the common basic structure.

Header

Header length: 116

Offset	Type	Content
0x15	u3	North boundary
0x18	u3	East boundary
0x1B	u3	South boundary
0x1E	u3	West boundary
0x21	uint	Offset of section 1 - Map levels
0x25	uint	Length of section 1
0x29	uint	Offset of section 2 - Subdivisions
0x2D	uint	Length of section 2
0x31	uint	Offset of section 3 - Map copyright
0x35	uint	Length of section 3
0x39	ushort	Record size in section 3. Values seen: 3
0x3B	uint	=0
0x3F	byte	POI Display Flags Bit 0: Detailed map Bit 1: Transparent map Bit 2: Show street before street number Bit 3: Show ZIP before city Bit 4: ? Bit 5: Drive on left Bit 6: ?
0x40	u3	Display priority
0x43	byte	Distance for start route calculation. Values seen: 0, 1
0x44	byte	Draw priority. Values seen: 0, 1, 2, 3, 4, 6
0x45	ushort	? Values seen: 1, 4, 7, 10, 13, 15, 17, 20, 22, 23, 29, 31, 33, 43
0x47	ushort	? Values seen: 1, 0x00B6, 0x00C8, 0x00E6, 0x00EB, 0x00EE, 0x00F2, 0x00F7, 0x00F8, 0x0101, 0x010C, 0x0193, 0x0175, 0x01B5, ...
0x49	byte	=0
0x4A	uint	Offset of section 4 - Polyline overview
0x4E	uint	Length of section 4
0x52	ushort	Record size in section 4. Values seen: 2, 3
0x54	uint	? Values seen: 0, 1, 2, 3
0x58	uint	Offset of section 5 - Polygon overview
0x5C	uint	Length of section 5
0x60	ushort	Record size in section 5. Values seen: 2, 3
0x62	uint	? Values seen: 0 – 13
0x66	uint	Offset of section 6 - Point overview
0x6A	uint	Length of section 6

Offset	Type	Content
0x6E	ushort	Record size in section 6. Values seen: 3
0x70	uint	=0

Header length: 120

Offset	Type	Content
0x74	uint	Map ID

Header length: 154

Offset	Type	Content
0x78	uint	=? Values seen: 0 and 1000000
0x7C	uint	Offset of section 7 - Extended type offsets
0x80	uint	Length of section 7
0x84	ushort	Record size in section 7. Values seen: 1, 4, 5, 8, 9, 12, 13, 16, 17, 20, 28, 29, 33, 36 and 37 (0 is also OK when the length of section is 0.)
0x86	ushort	? Values seen: 0, 1, 3, 5, 7, 0x0480, 0x04C0, 0x0607, 0x0647, 0x0807, 0x1007, 0x1607, 0x6000, 0x6001, 0x6002, 0x6407, 0xE401, 0xE407, 0xE607 and more
0x88	ushort	? Values seen: 0, 1, 64, 65
0x8A	uint	Offset of section 8 - Extended type overviews
0x8E	uint	Length of section 8
0x92	ushort	Record size in section 8. Values seen: 3, 4, 5, 6 (0 is also OK when the length of section is 0.)
0x94	ushort	Number of ExtType Line Types
0x96	ushort	Number of ExtType Area Types
0x98	ushort	Number of ExtType Point Types

Header length: 188

Offset	Type	Content
0x9A	uint	Map value 1
0x9E	uint	Map value 2
0xA2	uint	Map value 3
0xA6	uint	Map value 4
0xAA	uint	Key. Should be 0 if not locked and <>0 if locked.
0xAE	uint	Offset of section 9 - ??
0xB2	uint	Length of section 9
0xB6	ushort	Record size in section 9.
0xB8	uint	=0

Header length: 202

Offset	Type	Content
0xBC	uint	Offset of section 10 - ??
0xC0	uint	Length of section 10
0xC4	ushort	Record size in section 10.
0xC6	uint	=0

Header length: 206

Offset	Type	Content
0xCA	byte	? Values seen: 0, 40, 50, 58, 83, 154 and 155
0xCB	3 x byte	=0

Header length: 207

Offset	Type	Content
0xCE	byte	? Values seen: 0, 65, 66, 69, 75, 82, 88

Header length: 211

Offset	Type	Content
0xCF	uint	Map ID 2. Should be same as Map ID

Header length: 213

Offset	Type	Content
0xD3	ushort	? Values seen: 0, 1, 5, 21

Header length: 273

Offset	Type	Content
0xD5	uint	Offset of section 11 - ??
0xD9	uint	Length of section 11
0xDD	ushort	Record size in section 11.
0xDF	uint	? Values seen: 0, 10001, 10006, 10262, 18326, 26390
0xE3	uint	Offset of section 12 - ??
0xE7	uint	Length of section 12
0xEB	ushort	Record size in section 12.
0xED	uint	? Values seen: 0 and many others
0xF1	uint	Offset of section 13 - ??
0xF5	uint	Length of section 13
0xF9	ushort	=0
0xFB	uint	Offset of section 14 - ??
0xFF	uint	Length of section 14
0x103	ushort	Record size in section 14.
0x105	byte[12]	=0

Header length: 289

Offset	Type	Content
0x111	uint	? Values seen: 0, 1
0x115	uint	? Values seen: 0, 17, 20, 885, 2202, 2214, 2279, 2317, 2365, 2488, 2595 and many others
0x119	uint	? Values seen: 0, 0x000DCA11
0x11D	uint	? Values seen: 0, 1, 2

Header length: 295

Offset	Type	Content
0x121	ushort	=0
0x123	uint	? Values seen: 17, 20, and many others

Header length: 309

Offset	Type	Content
0x127	uint	Offset of section 15 - ??
0x12B	uint	Length of section 15
0x12F	ushort	=0
0x131	uint	=0

Section 1

TBD

Section 2

TBD

Section 3

TBD

Section 4

TBD

TRF file format

This file follows the common basic structure.

Special: The record size is between the offset and the length!

Header

Header length: 73

Offset	Type	Content
0x15	byte	? Value seen: 0x18
0x16	byte	? Value seen: 0x16 (Could be number of records in section 5)
0x17	uint	Offset of section 1 - ?
0x1B	uint	Length of section 1
0x1F	uint	Offset of section 2 - ?
0x23	uint	Length of section 2
0x27	ushort	Code page for section 2. Value seen: 65001 (UTF-8), see [5] and [6].
0x29	uint	Offset of section 3 - ?
0x2D	uint	Length of section 3
0x31	uint	Offset of section 4 - ?
0x35	ushort	Record size in section 4. Values seen: 40
0x37	uint	Length of section 4
0x3B	uint	Offset of section 5 - ?
0x3F	ushort	Record size in section 5. Values seen: 7
0x41	uint	Length of section 5
0x45	uint	? Value seen: 7

Section 1

TBD

Section 2

A whole lot of 0-terminated UTF-8 strings (see [17]). Sometimes there are two 0x00 bytes in sequence. This could mean that a number of strings belong together, but some can be empty.

Section 3

TBD

Section 4

TBD

...

TYP file format

This file follows the common basic structure.

Known header lengths: 91, 110, 156, 164, and 174

Header length: 91 (0x5B)

Offset	Type	Content	
0x15	ushort	Codepage	
0x17	uint	Offset of section 1 - Points	
0x1B	uint	Length of section 1	
0x1F	uint	Offset of section 2 - Lines	
0x23	uint	Length of section 2	
0x27	uint	Offset of section 3 - Polygons	
0x2B	uint	Length of section 3	
0x2F	ushort	FID = Family ID (a.k.a. MapId)	For the known values of PID and FID see my other document Garmin_MPS_GDB_and_GFI_Format.pdf
0x31	ushort	PID = Product ID	
0x33	uint	Offset of section 4 – Points Table	
0x37	ushort	Record size in section 4 Value depends on the length of section 1: 3: length < 0xFF 4: length < 0xFFFF 5: length < 0xFFFFFF 6: length >= 0xFFFFFF (0 is also OK when the length of section is 0.)	
0x39	uint	Length of section 4.	
0x3D	uint	Offset of section 5 – Lines Table	
0x41	ushort	Record size in section 5. Value depends on the length of section 2: 3: length < 0xFF 4: length < 0xFFFF 5: length < 0xFFFFFF 6: length >= 0xFFFFFF (0 is also OK when the length of section is 0.)	
0x43	uint	Length of section 5	
0x47	uint	Offset of section 6 – Polygons Table	
0x4B	ushort	Record size in section 6. Value depends on the length of section 3: 3: length < 0xFF 4: length < 0xFFFF 5: length < 0xFFFFFF 6: length >= 0xFFFFFF (0 is also OK when the length of section is 0.)	
0x4D	uint	Length of section 6	
0x51	uint	Offset of section 7 – Polygons Order Table	

Offset	Type	Content
0x55	ushort	Record size in section 7. Seen value: 5 (0 is also OK when the length of section is 0.)
0x57	uint	Length of section 7

Header length: 110 (0x6E) – Extra Points

Offset	Type	Content
0x5B	uint	Offset of section 8 – Extra points Table
0x5F	ushort	Record size in section 8. Value depends on the length of section 9: 3: length < 0xFF 4: length < 0xFFFF 5: length < 0xFFFFFF 6: length >= 0xFFFFFF (0 is also OK when the length of section is 0.)
0x61	uint	Length of section 8
0x65	byte	Seen values: 0, 7, 11, 14, 15, 19, 31 (probably 0 when length of section 8 is 0) Flags?
0x66	uint	Offset of section 9 – Extra points
0x6A	uint	Length of section 9

Header length: 156 (0x9C) – extra Point labels

Offset	Type	Content
0x6E	uint	Seen value 0
0x72	uint	Offset of section 10 – Strings
0x76	uint	Length of section 10
0x7A	uint	Record size in section 11. Value is length of offset + length of type. Length of offset depends on the length of section 10: 3: length < 0xFF 4: length < 0xFFFF 5: length < 0xFFFFFF 6: length >= 0xFFFFFF Seen values: 0, 4 and 6 (0 is OK when the length of the section is 0.)
0x7E	uint	Seen value 0, 18, 27 Flags?
0x82	uint	Offset of section 11 – Strings Table sorted by offset
0x86	uint	Length of section 11
0x8A	uint	Record size in section 12. Value is length of offset + length of type. Length of offset depends on the length of section 10: 3: length < 0xFF 4: length < 0xFFFF 5: length < 0xFFFFFF 6: length >= 0xFFFFFF Seen values: 0, 4 and 6 (0 is OK when the length of the section is 0.)
0x8E	uint	Seen value 0, 18, 27 Flags?
0x92	uint	Offset of section 12 – Strings Table sorted by type

Offset	Type	Content
0x96	uint	Length of section 12
0x9A	ushort	Seen value 0, 7, 18, 19, 100

Header length: 164 (0xA4) – indexing a selection of POIs

Offset	Type	Content
0x9C	uint	Offset of section 13 – Strings Table sorted by type Sometimes this section shares the file space with Section 12 (TYP12 == TYP13), and sometimes it is a separate section.
0xA0	uint	Length of section 13

Header length: 174 (0xAE) – active routing

Offset	Type	Content
0xA4	uint	Offset of section 14 – ?
0xA8	ushort	Record size in section 14. Seen value: 6
0xAA	uint	Length of section 14

Order of sections seen:

2, 5, 7

3, 6, 2, 5, 1, 4, 7

3, 6, 2, 5, 14, 1, 4, 7, 9, 8

3, 6, 2, 5, 14, 1, 4, 7, (32 byte unused =0), 8, 9, 10, 11, 12, 13

8, 9, 10, 11, 12

MapTk leaves some space before Section 7 and writes there:

00 00 4d 61 70 54 6b 20 33 2e 34 2e 35 00 “MapTk 3.4.5”

TYPWiz4 leaves some space between Section 5 and 14 and writes there:

54 59 50 57 69 7a 34 00 “TYPWiz4”

It also adds 32 unused bytes 00 at the end.

Section 1

Points Data

Contains a compact sequence of point entries, whose layout is explained below in the table. Section 4 (see below) is an array containing type and offset for each of these point entries.

Point entry:

Type	Content
byte	Flags bit 0: always set bit 1: 1=night bitmap present bit 2: 1= label present bit 3: 1 = font present
byte	Columns
byte	Lines
Bitmap (see below)	(Day) bitmap
If flags bit 1 is set:	
Bitmap	Night bitmap
If flags bit 2 is set:	
Label (see below)	
If flag bit 3 is set:	
Font (see below)	

Bitmap

Type	Content
byte	Number of colors (size of color table below)
byte	Color mode: 0 = no transparency available 16 = one transparent color available 32 = 16 levels of transparency available
Color table:	
If number of colors >0, contains a table of colors. Otherwise, if color mode is 16, contains a single color value for the transparent color, otherwise is empty:	
Number of colors x color - or -	The table of colors is stored in a compact way (no gaps). The last byte of the table may contain unused bits which should be 0. If color mode is 0 or 16: Each color value is three byte RGB Otherwise (32): Each color value is ARGB encoded as three x 8 bit RGB + 4 bit alpha. An alpha value of 0 means completely opaque and a value of 15 means completely transparent. Yes, each second color starts on an odd nibble within a byte! You have to mask each first color and shift each second color.
color	Color value is a three byte RGB value
Pixel array:	
Number of lines	Each line is stored in a compact way (no gaps). The last byte of each line may contain unused bits which should be 0. If a color table is present, the line consists of indices into this color table, otherwise it consists of color values directly:
	Indices: The size of the indices depends on the number of colors and the color mode. Let *n* be the number of colors. For color mode 16, 1 must be added for the

Type	Content	
		transparent color which is not in the color table and uses the highest index. Then the size of the index is: 1 bits if *n* < 2 2 bits if n >= 2 and < 4 4 bits if *n* >= 4 and < 16 8 bits otherweise
	- or - Colors	If color mode is 0 or 16: Each color value is three byte RGB Otherwise (color mode 32): Each color value is ARGB encoded as three x 8 bit RGB + 4 bit alpha, as described above.

Label

Type		Content
VarInt		Size of label in byte
Sequence of up to four(?) texts in different languages, each consisting of:		
	byte	Language code
	string	Text

Language code values:

Value	Meaning
0	unspecified
1	french
2	german
3	dutch
4	english
5	italian
6	finnish
7	swedish
8	spanish
9	basque
10	catalan
11	galician
12	gaelic
13	welsh
14	danish
15	norwegian
16	portuguese
17	slovak
18	czech
19	croatian
20	hungarian
21	polish
22	turkish
23	greek
24	slovenian

Value	Meaning
25	russian
26	estonian
27	latvian
28	romanian
29	albanian
30	bosnian
31	lithuanian
32	serbian
33	macedonian
34	bulgarian
35	brazilian
36	korean
37	japanese
38	chinese
39	traditional chinese
40	thai
41	arabic
42	belarusian
43	ukrainian
44	moldavian
45	montenegrin
46	irish
47	? English-derivate
51	? French-derivate

Font

Type	Content
byte	Font flags bit 0-2: extended font bit 3: 1=day font present bit 4: 1= night font present
If font flag bit 3 is set:	
Color	Day font color
If font flag bit 4 is set:	
Color	Night font color

Section 2

Lines Data

Contains a compact sequence of line entries, whose layout is explained below in the table. Section 5 (see below) is an array containing type and offset for each of these line entries.

Line entry:

Type		Content
ushort		Flags bit 0: 1=separate colors for day and night bit 1: 1=no background (if bit 0=0) / no night background (if bit 0=1) bit 2: 1=no border width (if bit 0=0) / no day background (if bit 0=1) bit 3-7: array lines bit 8: label present bit 9: wide bit 10: font present
If flags bit 0 is not set:		
	Color	Foreground color
	Color	Background color – present if flags bit 1 is not set
Else (flags bit 0 is set):		
	Color	Day foreground
	Color	Day background – present if flags bit 2 is not set
	Color	Night foreground
	Color	Night background – present if flags bit 1 is not set
If array lines ==0:		
	byte	Line width
	byte	Border width – present if flags bit 2 is not set
Else (array lines >0):		
	array lines x uint	Pixel
If flags bit 8 is set:		
	Label (see above)	
If flag bit 10 is set:		
	Font (see above)	

Section 3

Polygons data

Contains a compact sequence of polygon entries, whose layout is explained below in the table. Section 6 (see below) is an array containing type and offset for each of these line entries.

Polygon entry:

Type	Content
byte	Flags bit 0: 1=separate colors for day and night bit 1: 1=no background (if bit 0=0) / no night background (if bit 0=1) bit 2: 1=unknown (if bit 0=0) / no day background (if bit 0=1)

Type		Content
		If bit 0 is not set, bits 1 and 2 are same most of the times bit 3: pixel array present bit 4: label present bit 5: font present
If flags bit 0 is not set:		
	Color	foreground color
	Color	background color – present if flags bit 1 is not set
Else (flags bit 0 is set):		
	Color	day foreground
	Color	day background – present if flags bit 2 is not set
	Color	night foreground
	Color	night background – present if flags bit 1 is not set
If flag bit 3 is set:		
	128 x byte	Pixel array 32 x 32 pixel @ two colors
If flags bit 4 is set:		
	Label (see above)	
If flag bit 5 is set:		
	Font (see above)	

Section 4

Table with types and offsets for points. Each entry has fixed size.

Entry layout:

Offset	Type	Content
0x00	ushort	Type (Type << 5 | Subtype)
0x02	byte, ushort, uint24 or uint32	Offset in Section 1

Section 5

Table with types and offsets for lines. Each entry has fixed size.

Entry layout:

Offset	Type	Content
0x00	ushort	Type (Type << 5 | Subtype)
0x02	byte, ushort, uint24 or uint32	Offset in Section 2

Section 6

Table with types and offsets for polygons. Each entry has fixed size.

Entry layout:

Offset	Type	Content
0x00	ushort	Type (Type << 5 | Subtype)
0x02	byte, ushort, uint24 or uint32	Offset in Section 3

Section 7

Polygon draw order table. Lists the types / subtypes in their draw order for each level.

Entry layout:

Offset	Type	Content
0x00	byte	Type
0x01	uint32	SubTypes

There are three types of entries:

- Type == 0 and Subtypes == 0
  Increases the level by one.
- Type != 0 and Subtypes == 0
  Means the type with value Type.
- Type != 0 and Subtypes != 0
  Means the type with value 0x100 + Type and with the subtypes indicated by the individual bits set in SubTypes. Bit 0 .. 31 stand for subtypes 0 .. 31.

Example:

0x01 0x02 0x00 0x00 0x00 Level 1, Type 257, Subtypes 1

0x04 0x01 0x02 0x00 0x00 Level 1, Type 260, Subtypes 0 and 9

0x01 0x00 0x00 0x00 0x00 Level 1, Type 1

0x00 0x00 0x00 0x00 0x00 next level

0x01 0x04 0x00 0x00 0x00 Level 2, Type 257, Subtypes 2

...

Section 8

Table with brand and offsets for extra points. Each entry has fixed size.

Entry layout:

Offset	Type	Content
0x00	ushort or uint24	Brand
0x02	byte, ushort, uint24 or uint32	Offset in Section 9

Section 9

Extra Points Data

Contains a compact sequence of extra point entries, whose layout is explained below in the table. Section 8 (see above) is an array containing brand and offset for each of these extra point entries.

Extra point entry:

Type	Content	
byte	Number of icons	
Number of icons times:		
	VarInt	Number of bits
	VarInt	Number of additional Bitmaps
	byte	Width
	byte	Height
	Bitmap (see above)	Bitmap
	Number of additional Bitmaps x Bitmap	Bitmap

Section 10

List of strings.

Sequence of 0-terminated strings. The first string is always empty, meaning “no string”.

Section 11

Table with offset and type for each string in table 10. Sorted in ascending order by offset. Each entry has fixed size.

Entry layout:

Type	Content
byte, ushort, uint24 or	Offset in Section 10, points to beginning of string

Type	Content
uint32 (calculated from size of section 10)	In ascending order, but multiple entries may point to the same string.
ushort or uint24 (remaining space)	Type (Type << 5 | Subtype)

Section 12

Table with type and offset for each string in table 10. Sorted in ascending order by type. Each entry has fixed size.

Entry layout:

Type	Content
ushort or uint24 (remaining space)	Type (Type << 5 | Subtype) In ascending order.
byte, ushort, uint24 or uint32 (calculated from size of section 10)	Offset in Section 10, points to beginning of string

Section 13

This table has same layout as Section 12. Sometimes this section shares the file space with Section 12 (TYP12 == TYP13), and sometimes it is a separate section.

Section 14

Active Routing

Each table entry has fixed size of 6.

Entries are for:

1) walking, 2) cycling, 3) hiking, 4) tour cycling, 5) mountaineering, 6) mountain-biking

References

Used sources of information

Also see the references in the IMG/ADM/TTS Container File Format document.

[1] http://svn.parabola.me.uk/display/trunk/doc/nod.txt

[2] cGPSmapper from Stanislaw Kozicki https://sourceforge.net/p/cgpsmapper/code/HEAD/tree/

[3] http://www.offroad-bulgaria.com/showthread.php?t=65281

[4] https://www.digital-eliteboard.com/threads/neue-garmin-mapids-und-garmin-mapid-finder.26439/

[5] Code Pages http://msdn.microsoft.com/en-us/goglobal/bb964653.aspx

[6] Code Pages http://en.wikipedia.org/wiki/Code_page

[7] https://github.com/FSofTlpz/Garmin-DEM-Build/raw/master/DEM-Daten.pdf

[8] https://www.pinns.co.uk/osm/typformat.html

[9] https://web.archive.org/web/20110813055156/http://pinns.co.uk/osm/docs/Type_Format.pdf

[10] TYPViewer https://sites.google.com/site/sherco40/

[11] genTYP https://sites.google.com/site/cypherman1/gentyp

[12] https://web.archive.org/web/20130312100018/http://ati.land.cz/gps/typdecomp/source.cgi

[13] https://web.archive.org/web/20130312100049/http://ati.land.cz/gps/typdecomp/source.cgi?v=1

[14] TYPwiz4 https://web.archive.org/web/20161021224405/http://pinns.co.uk/osm/typwiz4.html

Standards and specifications

[15] ISO/IEC/IEEE 60559:2011 Information technology – Microprocessor Systems – Floating-Point arithmetic (or IEEE Std 754-2019)

[16] ISO/IEC 8859-1:1998 Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1

[17] UTF-8, a transformation format of ISO 10646, https://tools.ietf.org/html/rfc3629

Sources of sample files

Also see the references in the IMG/ADM/TTS Container File Format document.

[18] os50 Style & TYP https://www.pinns.co.uk/osm/styles.html

[19] polygons.typ https://www.pinns.co.uk/osm/typwiz6_whatsnew.html

[20] Freizeitkarte Designs https://www.freizeitkarte-osm.de/garmin/de/design.html#designdownloads