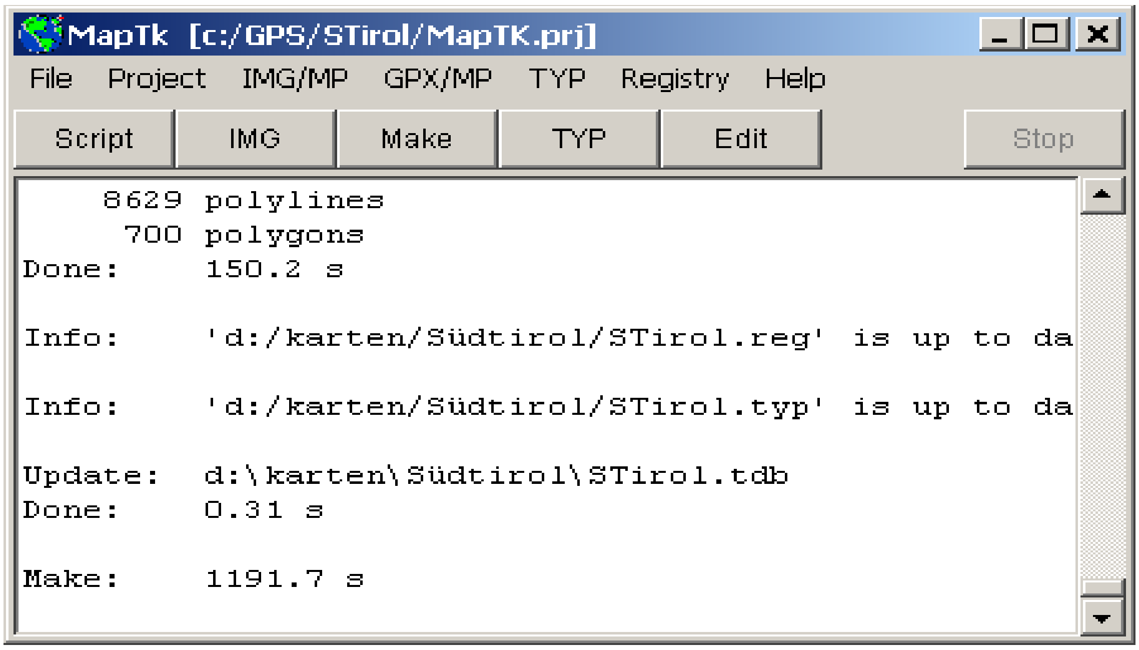
MapTk

Map Toolkit

Version 4.0


MapTk [c:/GPS/STirol/MapTK.prj]
File Project IMG/MP GPX/MP TYP Registry Help
Script IMG Make TYP Edit Stop

```
    8629 polylines
     700 polygons
Done:    150.2 s

Info:    'd:/karten/Südtirol/STirol.reg' is up to da

Info:    'd:/karten/Südtirol/STirol.typ' is up to da

Update:  d:\karten\Südtirol\STirol.tdb
Done:    0.31 s

Make:    1191.7 s
```

2016-06-21

Contents

1 Overview 4
1.1 Installation and start 4
1.2 Start parameter 5
1.3 Functions 5
1.4 Compatibility of maps 6
2 Menu and buttons 7
2.1 Menu 7
2.2 Buttons 15
3 Functions 16
3.1 'Script' 16
3.2 'IMG' 18
3.3 'Make' 18
3.4 'TYP' 19
3.5 Overview map 20
3.6 Export in GPSMapEdit 21
4 Automatic routing 22
4.1 Polylines for routing 22
4.2 Edit the map 23
4.3 Solving routing problems 24
5 Editor 25
5.1 New PRJ file 25
5.2 Header 26
5.3 Maps 27
5.4 Select window 28
5.5 POI 29
5.6 Polyline 31
5.7 Polygon 33
5.8 Script 34
6 Files 37
6.1 MapTk.dat 37
6.2 PRJ file 37
6.3 PRJ file and the functions 41
6.4 MP file 42
6.5 IMG file 45
6.6 TYP file 45
6.7 TDB file 46
6.8 Index 46
6.9 LOG file 47
7 Python-Script 48
7.1 Overview 48
7.2 Data of the Header 49
7.3 Data of Objects 49
7.4 Some Python 50
8 Tipps & Tricks 53
8.1 Relation TYP and IMG 53
8.2 Using the Levels 53
8.3 Very Large Maps 54
8.4 Complex lines and polygons 55
8.6 Folders 56
8.7 Working with many files 56
8.8 All files in one folder 56

8.9 Script....57
9 Appendix....61
9.1 Things to know....61
9.2 Map Example....62
9.3 'Standard' POI types / symbols....62
9.4 User defined Types....64
9.5 Color-palette (256 colors)....68
9.6 Language codes....69

1 Overview

This document describes the functions of the program MapTk (Map Toolkit). This description is no introduction into producing and processing sets of maps. It is presupposed that the user of the program is familiar with all steps of making maps. Sound PC-knowledge are necessary.

The author adopts no liability for damages, that could originate through the use of the program. That is applied also to possible injuries of the copyright through the application of the program.

The language of the program is English. The English is very simply held - as long as news of the operating-system or python doesn't come through – and should make unpracticed no problems.

Download at http://maptk.dnsalias.com. For processing the maps GPSMapEdit is strongly recommended. The functions of the registered version are useful.

1.1 Installation and start

Since version 3.3 only Windows is supported.

No external programs are necessary for the implementation of MapTk. Starting 'MapTk <version> Windows.exe' will install the Program using an Inno-Setup-Script. Only installing in the folder 'program files' needs administrator privileges (using Vista the program should start be started as an administrator). There are no registry entries for the program. The program is developed with python 2.7 under Windows 10 32-Bit. The application of python should guarantee no problems with other Windows versions. If missing the icon library 'MapTk.dat' an empty file with default preferences is written. A message is display at first start of MapTk.

At least two modes to start:

1. The extension '.prj' should be linked to MapTk.exe. Double click on a PRJ file will start MapTk with this file as the current project. The PRJ file must be located in the folder of the source MP files.
2. MapTk may also be called via link in the directory of the source files of the maps. The project file becomes MapTk.prj. 'Executes in' of the link either contains the current path, '.\' or remains empty.

The project-file, central for a set of maps 'MapTk.prj' controls the program. This file **must** be located in the directory of the source files of the set of maps. The data files of the project are located in this directory, the usually from MapSource used files in another.

If PDF files are not assigned to a PDF viewer the path to a viewer is to specify in the preference data.

1.2 Start parameter

Command line parameter:

- no parameter: MapTk is starting with 'MapTk.prj'. This file is created at the lastest before saving the project.
- '-make <file>': MapTk is starting with <file> as the actual project file and executes 'Make' immediately. A reduced protocol is written as 'MapTk.log'. MapTk terminates automatically.
- '-script <file>': MapTk is starting with <file> as the actual project file and executes 'Script' immediately. A reduced protocol is written as 'MapTk.log'. MapTk terminates automatically.
- '-findnotconnected'. Syntax:
  '-findnotconnected -distance=<distance in meter>[,auto] <PRJ file>' or shorter:
  ' -f -d=<distance in meter>[,auto] <PRJ file>'.
  Searching not connected near nodes in all tiles of the project.
- '<project>.prj': MapTk is started with this file as the current project file.
- '<map>.mp': The MP file is compiled a am IMG file.
- '<file1> <file2>': The MP file file1 is compiled into an IMG file file2.
- 4 parameter, used by GPSMapEdit to export into IMG files.

1.3 Functions

The most important functions of the program in the overview:

1. With the function 'Script' global alterations of maps and set of maps can be performed. I.e. renaming or removing POIs according to certain criterion, altering of the zoom-level, ...
   'Script' is a pre-processor for the 'IMG'-function.
2. Compiling of individual or several source files to Garmin image files ('IMG'), including 3-byte-types. This is the traditional style. Maps in this style can be converted to GMAP format using Garmins MapConverter.
3. From all IMG files optional an index file for POIs is produced (MDR file *_mdr.img).
4. If any of the roads has an address an optional address index is produced in the MDR file (page 55).
5. Tools to find errors in the routing network (not connected nodes) and in parameters of POIs (cities). The address part of the index is checked while building the index. Problems are found in the log.
6. Adding information for automatic calculation of routes is optionally. MapTk is based on MP file routing graphs prepared by GPSMapEdit. See page 22.
7. Generating of a project, i.e. compile all changed sources, if necessary generating of the TDB-, REG-, MDX-, MDR- and TYP files ('Make').
8. Work on TYP files with a graphical oriented editor ('Edit'). See page 25. List of used types from MP files or installed set of maps.
9. Compiling TYP file from the project-file ('TYP').

10. Create overview maps from one or more detailed maps. See page 20.
11. Making registry entries in the REG file to declare the set of maps in MapSource ('Make'). Backup of registry entries or remove set of maps from registry.
12. TYP files, also for existing sets of maps like 'City Select' or 'Topo Germany V2' ('TYPE').
13. Analyze of IMG-, TDB- and TYP files. No analyze of NT style files.
14. XOR with 0x96 of an IMG file and back ('XOR'),
15. Transparent flag in IMG files switch on or off.
16. Read and write the draw priority byte of IMG files.
17. Functions to work with GPX files. See page 10.
18. Compatible with GPSMapEdit. Export from GPSMapEdit, as substitute for cgpsmapper.exe and generation of a skin file. Using bookmarks.

Name conventions: Names for detailed maps are numerical, for example '01234567.img'; for the overview map with letters incipient up to 8 characters, i.e. 'abc2.img'. Recommended: The overview map should be called 'basemap'. The names of the MP files is any. The name of the overview map should not contain spaces. All other files are automatically named.

It is recommended to aim an individual folder for each set of maps for the source MP files and to start MapTk from these folder. A link should made. The IMG files generated for MapSource or MapInstall should also receive an individual directory per set of maps or must be located in the folder of the source files.

1.4 Compatibility of maps

The compiled IMG files are compatible with Garmin GPS receivers. Incompatible receivers are not known. 3-byte-types are only shown with newer receivers (about end of 2005).

Basecamp and MapSource already showing products with TYP file on the PC right. TYP files for for example 'CitySelect' can be generated. That with MapTk generated set of maps and TYP file is transferred quite normally with MapSource or MapInstall to the GPS receiver, also mixed for example 'CitySelect' with a topographical map.

The TYP files are suitable for 'CitySelect', 'MetroGuide', 'CityNavigator' and topographical maps.

Marine maps are not supported.

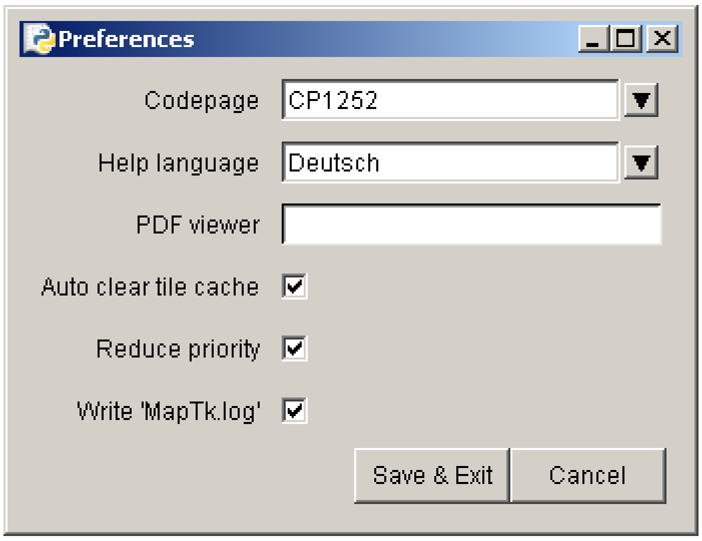
Maps with copy protection ('locked'), are not supported.

2 Menu and buttons

2.1 Menu

2.1.1 'File'

- 'Edit text file' processes a selectable text file. It has no accentuated key words.
- 'Clear cache': Clears the cache of MapSource, BaseCamp und MapInstall.
- 'Rename GPX to time stamp': GPX file from different sources have different names containing often date and time. Sorting by name is not possible. This function will unify the file name using the format 'YYYY-MM-DD hh:mm:ss' to allow sorting. Files with following formats are renamed:
  ```
  YYYY-MM-DD hh.mm.ss Day
  Day DD-MMM-YY hh.mm.ss
  Track_DD-MMM-YY hh.mm.ss
  Track_DD-MMM-YY hhmmss
  Track_YYYMMDD hhmmss
  YYYY-MM-DD <digits>
  MMM-DD-YY hhmmss
  DD-MMM-YY hhmmss
  DD-MMM-YY hh mm ss
  YYYYMMDD hhmmss
  YYYYMMDD-<digits>
  ```
  The year 'YYYY' must be >= 2000. 'MMM' is a abbreviation of a month (English, German). For file following this pattern the file is scanned for the oldest time stamp <time>. The time in GPX files is based on UTC (former GMT). The time is corrected to the timezone of the PC and used as the new name. Files without a time stamp are renamed based on the old file name. The contents of a file is not changed.
- 'Preferences':



Codepage: CP1252
Help language: Deutsch
PDF viewer:
Auto clear tile cache: ✓
Reduce priority: ✓
Write 'MapTk.log': ✓

  - Define the code page (CP1251, CP1252).
  - Select the language for the help file (English, German).

  - Set the path to the PDF viewer. If a PDF viewer is not defined for Windows the standard viewer is used.
  - After saving a new TDB file the tile cache of MapSource (since version 6.15), BaseCamp and Mapinstall can be automatically cleared to show the changes of in new TYP file.
  - For more time consuming processes the priority can be reduced automatically.
  - A compact logging is switched on checking “Write 'MapTk.log'” (see page 47).
- 'Exit' finishes the program. If there are still windows open or processes active confirming is required. Alterations of the file in the editor are discarded if MapTk is aborted.

2.1.2 'Project'

- 'Change working directory' changes the project directory. The current project directory is shown in the title bar of the program [...].
- 'New project file' produces a presentation of the project-file 'MapTk.prj'. Before the editor is started, a dialog-box appears to enter the data for section [Project] and to decide whether suggestions should be copied for TYP files and / or templates of scripts should be copied into the PRJ file. Details see page 37.
- 'Edit project file' processes a selectable PRJ file. It has colored syntax highlighting for the script part.
- 'Build project' (= button 'Make' without retrieval of the project-file) is producing a set of maps controlled by a PRJ file. See page 37.

2.1.3 'IMG/MP'

- 'Reorganize' is reorganizing MP files. Polylines and polygons with more than 256 points are automatically splitted. Exception: Roads must be splitted manually (error message while compiling). Multiple data statements after splitting and trimming maps are dissolved. Data in levels > 0 is removed. Additionally conversions of old syntax is performed.
- 'Script' (= button 'Script' without retrieval of the project-file) reorganizes MP files like function 'Reorganize'. Changes of the map can be made by 4 Python scripts, that are stored in one PRJ file. See button 'Script' page 16.
- 'Compile map' (= button 'IMG') is producing IMG files from MP files. See page 37.
- 'IMG analysis': generates a MP file from an IMG file. This file contains the objects of all levels. The name of the IMG file is taken as ID into the MP file. The name of the generated MP file is derived from the description of the map or from the eventually available 'tdb.dict' (in the MP file: 'Name = ...'). An IMG file also can be converted with GPSMapEdit into the MP-Format very well. Not all GMA[1]-Style maps may be transferred to the MP format. 'locked' maps cannot become analyzed. Copy protection !

1 The GMA section contains the subfiles RGN, TRE, LBL, NOD and NET.

- 'TDB analysis': writes information from a TDB file into a text file. That is copyrights, names of the detailed maps, ID, position, sizes for example. Additional a file 'tdb.dict' is written. This file contains a list of all tiles in format <ID> : <Name>. While decompiling an IMG file he list is used name the MP file – if available.
- 'XOR IMG file': codes an IMG file with 0x96 or makes a coding declining.
- 'IMG file set transparent': sets the transparency-bit in the TRE-Subfile, also for files coded with XOR. The maps generated with IMG can be made here opaque, alternative in the PRJ file. 'IMG file reset transparent' removes the transparent-bit.
- 'IMG get 'DrawPriority'' displays the draw priority for all selected files. 'IMG set 'DrawPriority'' sets the draw priority for all selected IMG files to 0 ... 31. This can be applied to all IMG files, even for locked files.
- 'Create overview map': See page 20.

2.1.4 'GMAP'

- 'Convert to GMAP format'
  Maps based on Windows registry are converted to the GMAP format. This format does not use the registry. The file 'info.xml', located in folder '<name of map>.gma', contains the same information.
  The folder '<name of map>.gma' must be contained in one of the map folders of Garmin software to be used in BaseCamp or MapSource. Instead of the folder itself a link to the location of the GMAP folder is preferred. Instead of the REG file 'info.xml' created.
  The folder for GMAPs variies with the operationg system and version. Typical folders for Windows are '%APPDATA%\GARMIN\Maps' or '%PROGRAMDATA%\Garmin\Maps\'.
  Selecting the map:
  'Convert': the selected (installed) set of maps is chosen and converted.
  'Find REG': the **REG file** of the maps must be selected in an other window to be converted. All information is take from the REG file.
  'Cancel': the process is stopped.
  The destination folder, where the '<name of map>.gma' is stored must be chosen in a next window.
  The source files will remain unchanged.
- 'GMAP reverse converter'
  This is a reverse function. A set of maps is converted from GMAP format into the registry based format. All converted files will share a single folder. A REG file with absolute path to the files is created. Base of the path is the folder of the converted map. Double clicking the REG file will write all registry data (perhaps administrator rights are required).
  Selecting the map:
  'Convert': the selected (installed) set of maps is chosen and converted.
  'Find GMAP: The **folder** <name of map>.gmap has to be selected in an other window and is converted. The file 'info.xml' must exits in that folder.
  'Cancel': the process is stopped.
  The destination folder (<name of map>\) for the converted map is selected in a next window.
  The source files will remain unchanged.

- 'Split GMAPSUPP to GMAP'
  MapInstaller or MapSource are producing the file 'gmapsupp.img'. This file may be renamed and can contain tile of different sets of map. This file contains all information of the selected sets of maps and is split into one or more GMAP folders. The index is specially handled. The index is optimized to be used for devices and is not to be used in BaseCamp or MapSource. So the index is not included, but sometimes partly to be reconstructed using 'Index MDR from GMAP'.
  The splitted maps (<name of map>.gmap) are to be used with BaseCamp or MapSource if they are written into one of the folders for Garmin maps or placing a link there instead.
- 'Index MDR from GMAP'
  Creating an index is adapted to maps produced by MapTk. If e.g. the format of tiles is different or the index becomes to large the process is stopped with an error message. To start indexing the folder (<name of map>.gmap) is chosen. On success 'info.xml' is updated.

2.1.5 'GPX/MP'

This menu offers functions to work with GPX files. Some of the function need a lot of computing time. To combine 4 MBytes tracks in one ore more GPX file may need one hour.

Result file: The result of searching in some in this menu is a MP file named '#<source>.mp'. Bookmarks for all found positions used as a marker. Load the MP file of the tile into GPSMapEdit and add the result file to find the result in the map. The marker may stay in the corrected tile. Script will remove the markers and the compiler ignores markers. Bookmarks are comment in the MP file.

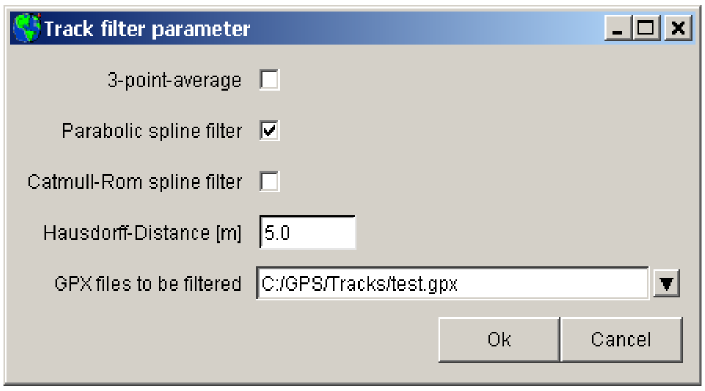
- 'MP to GPX': All polylines of specific types are converted to tracks in a GPX file. The type must be specified as a list. Example: '1-12,0x16' stands for all streets and trails. The types and groups of types are comma separated. The notation is decimal or hexadecimal. The label in the MP file becomes the name of the track[2]. The tag 'cmt' (comment) is use to store the type of the polyline in format 'Type=0x1234'The name of the GPX file[3] is derived from the name of the MP file.
  For POIs a symbol is added, derived from the type of POI (see page 62). Routes are transferred to the output file.
  The name(s) of the MP files are entered in the field 'MP files to convert'. The GPX file is names according with the MP file.
- 'Split GPX': A GPX file containing tracks, way points and routes is split in GPX files only containing the tracks, way points and routes. This function also makes most in BaseCamp or MapSource not readable GPX file compatible.
- 'Combine GPX files': This function combines one or more GPX files ('GPX files to combine') and the contents to a new file ('Combined output file').
  Tracks inside a mask with the width 'Width of track mask [m]' are combines to a new line using the average. That works fine for tracks of good quality. Crude errors like spikes or 'balls' logged during stops should be removed beforehand. The mask

2 If no label is given, the track is named Track_n, where n is an automatically generated number.

3 Attention: MapSource does not display and does not write back the comment field. Tracks without the a type are taken as trails 0x16 used further in MapTk.

should be in range 10 to 30 m. A too narrow mask will not combine belonging tracks. A too width mask will combine tracks which do not belong together. Better quality of the tracks allows narrow masks and will result in better quality of the result. Very good tracks should not be combined with poor tracks and will result in undistinguished lines. Only track of the same type (in tag 'cmt') are combined.
POIs may be are also combined inside a radius of 'Max. distance POI [m]'. Additional the name symbols must match and the name must be similar. Name are enough similar after compare phonetic (Soundex) and a Levenshtein-Distance of 0.
Routes are transferred to the output file.

- 'GPX track filter': All tracks in the file are filtered in order to beautify a track before integration into map. The start end the end point of a track is never changed. Points which are connected to other tracks in the file are also not changed. The name of the output file is derived from the input file, using a suffix '_filter'. There are 4 kinds of filter:

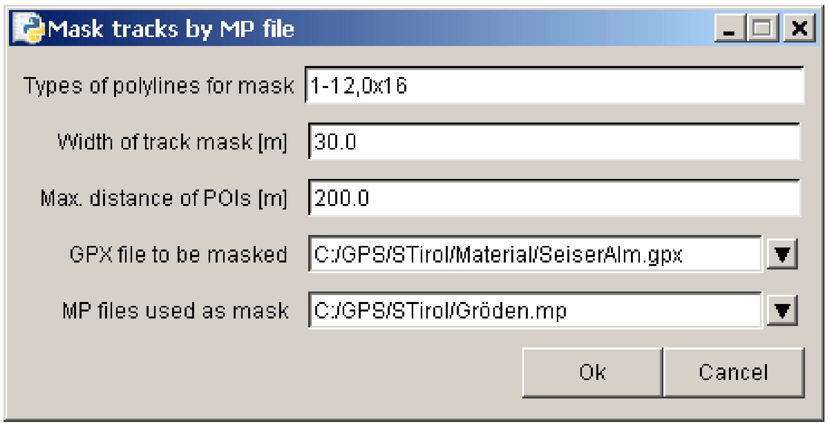

Track filter parameter

3-point-average	☐
Parabolic spline filter	☑
Catmull-Rom spline filter	☐
Hausdorff-Distance [m]	5.0
GPX files to be filtered	C:/GPS/Tracks/test.gpx

  - '3-point-average': The point and the two point adjacent points are taken to calculate the geometric average. This filter is recommended for more straight tracks with poor quality.
  - 'Parabolic spline filter': A polyline is calculated which is more smooth than the original track. This polyline must not go through the points of the track. The filter is suitable to beautify tracks, not changing the direction very sharp. The number of point is multiplied by 10.
  - 'Catmull-Rom spline filter': The calculated polyline will touch each point of the track. The filter should be used if the track contains sharp turns like serpentines. The number of point is multiplied by 10.
  - The parameter 'Hausdorf-Distance' is used for a Douglas-Peuker filter. This filter will reduce the number of points. All calculated points are inside a specified distance, the Hausdorff-Distance in meter. A typical Hausdorff-Distance is 3.0 m. A Hausdorff-Distance of 0.0 m will switch of the filter.

The spline filters will generate polylines with 10 times more points. So it is strongly recommended to use the spline filters only together the Douglas-Peuker filter. Only he Douglas-Peuker filter is to be combined with the other filters and is calculated last at all.
Which filter is best for a given track should be checked out.
All way points and routes are transferred to the output file.

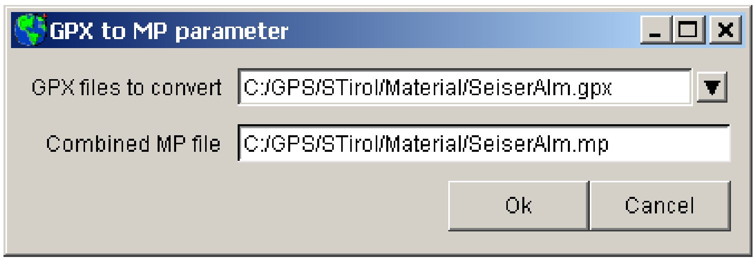
- 'Mask GPX tracks by MP':
  This function prepares Tracks to be taken into a map. Tracks of good quality are required to use these function successfully.

  
  Mask tracks by MP file

  Types of polylines for mask: 1-12,0x16
  Width of track mask [m]: 30.0
  Max. distance of POIs [m]: 200.0
  GPX file to be masked: C:/GPS/STirol/Material/SeiserAlm.gpx
  MP files used as mask: C:/GPS/STirol/Gröden.mp

  A mask with the width 'Width of track mask [m]' is generated from the polyline types in 'Types of polylines for mask'. The polylines are taken from the files in 'MP files used as mask'. All tracks points inside this mask are deleted. Meets a track one of the polylines the end point is moved to the polyline. The result is stored in GPX files in 'GPX file to be masked'. The name of the file is extended by '_mask'.
  POIs and routes are transferred to the output files.

- 'GPX track to MP':

  
  GPX to MP parameter

  GPX files to convert: C:/GPS/STirol/Material/SeiserAlm.gpx
  Combined MP file: C:/GPS/STirol/Material/SeiserAlm.mp

  All tracks are converted to polylines of the given type. The label is taken from the track. But if the name of the track was generated automatically by one of the above function, no label is written to the MP file. The level definition in the header of the MP fill is empty. So the the MP file can be integrated into any other MP file without problems.

- 'Find cities without region/country:
  All cities without defined region or country or invalid reference are listed. One or more files can be selected and are treated as one file. If the number of found cities is displayed a result file was written.

- 'Find multiple defined cities':
  Multiple defined cities are listed. One or more MP files may be selected. One or more files can be selected and are treated as one file. If the number of found cities is displayed a result file was written.

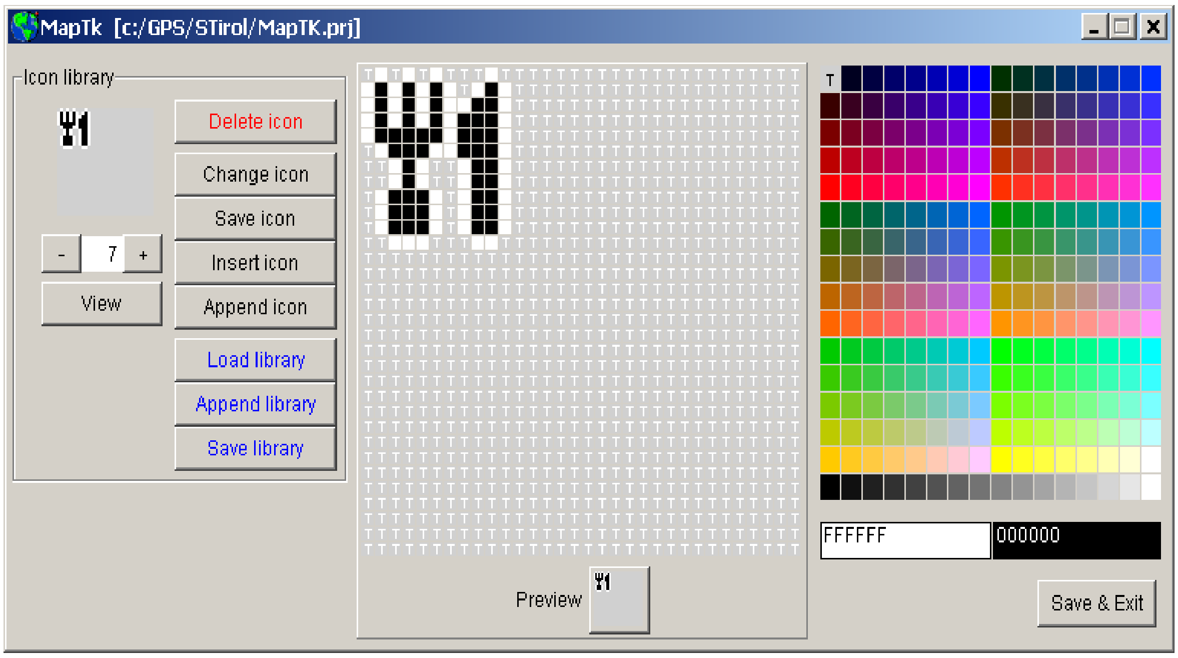
- 'Find POIs with invalid cities':
  The address of all POIs is checked to contain a valid city reference. One or more files can be selected and are treated as one file. If the number of found POIs is displayed a result file was written.

- 'Find not connected near roads':
  All nodes near to an other road are searched in selected MP files. The distance for searching is entered as 'Search for roads around nodes about [m]'. The search radius is entered in meters. If the number of found nodes is displayed a result file was written.
  Optional: Ending points can be connected automatically to routing nodes inside the searching radius .
- "Find not matching external nodes"
  This checks the whole project for misplaced or missing junctions between the tiles of the project. All IMG files of the project are dead. If the number of found nodes is displayed a result file was written. The name of this file is '#<FID>.mp' is written.

2.1.6 'Typ'

- 'Icon library': The editor for the icon library is similar to the POI editor (see page 29).

  The concept of usage is either a single library loaded and saved together with



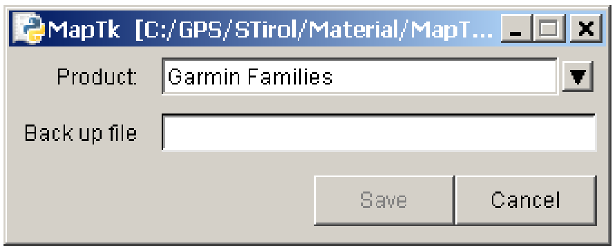
  MapTk or a project specific library using 'Load library' and 'Save library'. The advised name for specific libraries is 'MapTk.dat'.

  The appended file may contain an icon library or a single raster image of one of the following formats: BMP, GIF, ICO, JPG, PNG or TIF. 'Append library' appends all not yet existing icons to the current library. Multiple files are to select. The current library in memory is stored together with the preference data while terminating MapTk.
- 'TYP analysis': generates a PRJ file, that may be changed and converted into a TYP file again. <u>Restriction</u>: files with 2-character-color-definitions and night-colors are not supported.
- 'Compile TYP' (= Button 'TYP' without retrieval of the project file) compiles a TYP file from a PRJ file. See page 19.

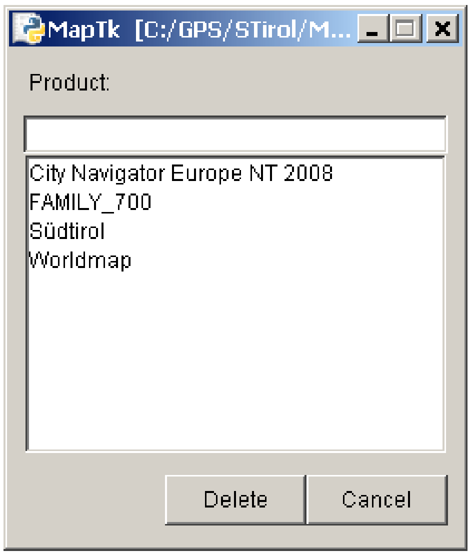
- 'List of used types' writes a list of all used POI, polyline or polygon type to a file for a set of maps. The concerned MP files are defined in block [IMG] in the PRJ file.
- PRJ file to GPSMapEdit skin' generates a text file from a PRJ file to be used to adapt the display in GPSMapEdit according to a TYP file. This works with GPSMapEdit 1.0.48.0[4] or above. The text file must be declared in 'View → Manage Map Skins...' and has to be activated in 'View → MapSkin'. The file is independent of ProductID and FamilyID. Only the day version should be used. Since version 2.1.78.5 also the TYP file may be used.

2.1.7 'Registry'

- 'Backup registry':
  The whole entry 'Families' or single sets of maps are stord in a REG file.



- 'Delete from registry':
  Single registry entries may be deleted. The map files are not deleted.



4 Polylines containing a pattern are displayed incorrect.

2.1.8 'Help'

- 'PDF-Help' shows the help-file (MapTk_*.pdf) in the PDF-Format in an individual process. The files must be located in the directory of MapTk.exe. English or German as defined in the preferences.
- 'Homepage' opens the homepage of MapTk in the standard browser.
- 'Info' shows the version of MapTk.exe.

2.2 Buttons

2.2.1 Script

Start a script based on the current PRJ file in the working directory. See page 16.

2.2.2 IMG

Compile one or more MP files to the IMG format. See page 18.

2.2.3 Make

Do all what is needed to build a complete set of maps, including TYP, index and REG files. Based on the PRJ file in the working directory. See Page 19.

2.2.4 TYP

Generate a TYP file from Data in the PRJ file. See page 19

2.2.5 Edit

Create or change a PRJ file with special editors for the different parts of the file. See page 25.

2.2.6 Stop

If activated the button stops the operation if multiple action are requested (e.g. In 'Make'). The number behind 'Stop' shows the number of pending processes:

0	no process is active
1	one process in progress
2 +	one or more precesses pending

3 Functions

The functions are started by click on the corresponding button or using the menu. With direct call of the function using the button, the project-file is taken from the current project directory. With start using the menu, the project-file can be selected in many cases.

3.1 'Script'

The maps for a set of maps often have different sources, mistakes and superfluous information is contained. It is recommended to reorganize new cards. With 'Script' gets the maps an uniform appearance, for example walkways (Type=0x16) only in the Level 0, residential streets (Type=0x06) until Level 2. The appearance of individual objects is adapted by Python scripts (in one PRJ file) to the personal wishes. See page 48. Because of the big amount of data a script is an essential work-relief.

Huge MP file may cause a memory overflow: `'MP file is too large !'`

The compiler ('IMG') appraises only data in the Level 0 and expects for zooming the statement 'EndLevel=' or 'Levels =' for all objects that are not only visible in Level 0. If MP files contain all objects separated in different levels a script mus be used (or with GPSMapEdit the function 'Join per-level elements'). Because the function 'IMG analysis' since version 1.3 generates 'IMG' compatible MP files, this step is no longer necessary if a MP file with 'IMG analysis' was generated. A PRJ file for four levels in the menu 'File' -> 'New project file' is generated (MapTk.prj).

The lines 'ID =' and 'Name =' must be stated in the MP file to be processed. Otherwise:
`Error:   'ID'` or
`Error:   'Name'`
and the function aborts.

If the ID itself is not declared for detailed maps '00000001' is inserted, with overview-maps the name of the MP file. 'Name' can remain empty but a warning is shown. The name of the overview map should not contain spaces.

In case of an error in a MP file a message is show and the script terminates. In case of a warning the script continues with message and a bookmark is written to the MP file.

See also example PRJ file for the functions on page 57.

By 'Script' made reorganization involves following points:

- The edition-file has in the block [IMG ID] always beside' ID =' and 'name = ':
  - 'Elevation=M'
  - 'LblCoding=9
  - 'CodePage=...' (if the codepage is not CP1252)
  - 'DrawPriority=...' (optional, default: 24)
  - 'Transparent=...' (optional)
  - 'Preview=' (optional)
  - 'Routing=' (optional)
  - 'Copyright=...' (optional)
  - 'Levels=...'
  - 'Levelx=...'
  - 'Zoomx=...' (optional)

  Exactly so many lines 'Levelx =' are necessary as 'Levels =..' are declared. 'x' stands for the Level 0... Levels-1. Except 'ID', 'Name', 'Transparent', 'Copyright', 'Codepage', 'Levels', 'Levelx' and 'Zoomx' all other statements as they are added by for example GPSMapEdit are without meaning for the later conversion into an IMG file and is not taken on. The entries 'Transparent', 'Copyright', 'Codepage', 'Levels' and all 'Levelx' / 'Zoomx' can be read and changed in the script part of the PRJ file [CUSTOM_HEADER]. The values for 'Zoom' are taken optionally from the MP file. Otherwise: 'Zoom0=0', 'Zoom1=1',.... - however only when compiling into an IMG file.

- The blocks [Countries], [Regions] and [Cities] are not supported since version 4.0. Configure GPSMapEdit to save the MP file using CityName, RegionName and CountryName instead of [City], [Region] and [Countries].
- The consistence of CityName, RegionName and CountryName is checked to avoid problems in the index. Messages like 'City has no region' marked as bookmark in the MP file (if index check is switched on for the project).
- Only coordinates in 'Origin0' or 'Data0' is taken on. Objects, with no 'Origin0' or 'Data0' are ignored. Lines with 'Data1' and coarser levels are ignored. The coordinates for level 1 and higher are generated later during the production of the IMG file from the Level 0.
- Some shields (for example highway) for streets are coded using control characters like in GPSMapEdit to be accepted from MapTk, for example '~[0x04]'.
- The names of the blocks are translated:
  [RGN10] or [RGN20] to [POI], [RGN40] to [POLYLINE] and [RGN80] to [POLYGON].
- Comments ('; ... ') are not taken over, appendixes ('; @... ') are taken over but not evaluated.
- Polylines and polygons with several lines 'Data0' as well as 'Origin0' become divided into several objects with otherwise the same data, one per 'Data0' / 'Origin0'.
- Polylines and polygons with more than 255 nodes become divided into several objects with otherwise the same data.
- During editing a map often routing nodes are left without any function. The routing information of those points is removed if the node does not connect different roads and does not carry turn restrictions.
- Heights- or depth-statements are converted from feet to meter if necessary.
- Objects with 'Levels' <0 are ignored, i.e. not into the output-file taken over (in the script 'Levels=-1' removes these objects).

Multiple treating a map with 'Script' doesn't alter the contents. The input-file is not overwritten but renamed to *:BAK.

The PRJ file may contain the blocks [...] only partly or may be empty. The script can be executed even without a PRJ file. In this case the following steps are performed:

- All data in levels >= 1 is removed.
- Cities in [RGN20] are moved to [POI].
- All cities in [RGN10] get the line 'City=Y'. If possible the lines CityName, RegionName and CountryName are added if CityIdx and [Country] are available.

3.2 'IMG'

This function generates an IMG file from a MP file directly. Extra codings are not necessary. Further files like the PRJ file are not necessary. The IMG file can be summarized into a set of maps or replace an existing tile. 'ID =' and 'Name =' must be stated in the MP file. The name of the generated IMG file is always the ID + '.img '.

Both detail-maps as well as overview maps can become generated. The differentiation takes place through the ID in the MP file. The number of the levels is 0 to 8. In overview-maps level 0 should start with the bit / coordinate of of highest level of the detail-maps. An additional Level clearly increases the IMG files.

Subfile RGN of a map may have up to 4 MByte. Larger maps issue an error message :
'`Abort: map is too complex (RGN) !`'.
The function is aborted. That yields IMG files is generated until something over 4 MByte from MP files of approximately 20 MByte ('topological' content). Larger maps must become divided using i.e. GPSMapEdit.

Standard object types and user defined types (> 0x10000) may be used.

Only the coordinates with the key 'Data0' or 'Origin0' are taken into the IMG file. Up to 8 levels are available. The highest Level is defined through 'Levels='. The coordinates of the Level 1 until Levels-1 are derived from 'Data0' through reduction of the bits per coordinate. Polylines and and polygons are simplified with the Douglas-Peuker-Function to reduce the file-size.

A background (Type=0x4b) is not necessary in any case, but is required for routeable maps ! With detailed-maps, a background should exist in order to show the borders of the tiles in MapSource. The background can be divided and also must not longer be a rectangle.

An already existing IMG file is headed.

The IMG files are at least with GPSMap60C(x), Colorado and Oregon compatible (tested). Reasons for possible incompatibility with other appliances are not known.

The size of generated IMG files is comparable to MapDekode for example. The speed is approximately 600 kByte/sec compiling a MP file (PC: 4 Gbyte memory, 3 GHz CPU, 4 cores).

3.3 'Make'

The updating of a set of maps is automated with this function. The control is performed by the PRJ file 'MapTk.prj'. 'Make' settles following tasks one after the other:

- If a MP file from the block [IMG] is older - or if the IMG file doesn't exist - the affiliated IMG file is compiled. In order to force a new compilation, a IMG file can simply be deleted in the folder 'IMGpath'.
- After successful compilation, the IMG file is moved into the folder 'IMGpath'. If moving goes wrong, because the tile is opened in MapSource for example, you can renewed later 'Make' moving this tile.
- After all tiles are updated the TYP- and REG files are generated. See page 19. Registry and REG file are compared and if inconsistent an information is displayed. If the 'Product name' is missing no REG file is produced.
- The TDB file is generated if the project file or one of the IMG files in the block [IMG] of the project-file is younger than the existing TDB file or no TDB file is found. Also IMG files may be use which are not generated by MapTk, even if XOR coded.
- At the end the optional index file (*_mdr.img) is generated. The detailed maps must be compiled with MapTK 4.0 or newer. Warning Messages in the log are optional. The map may be used with index-warnings in log but address will not be complete.

If the set of maps is already registered into the Registry, you immediately can use it in MapSource. Otherwise, the enrollment into the registry can take place with the straight generated REG file: Double-click on the REG file.

Apparently the registry entries are stored in two different paths using Windows-32-bit und Windows-64-bit. But this is only the view of 32- or 64-bit-programs. The user will not mention that.

A 32-bit-program (like MapTk) reads an writes

HKEY_LOCAL_MACHINE\SOFTWARE\Garmin\MapSource\Families\*

A 64-bit-program reads an writes

\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Garmin\MapSource\Families\*

3.4 'TYP'

A TYP file is generated on basis of a PRJ file, see page 37. Colors for a night-design and color-coding with 2 characters is not supported. If the TYP file is not registered the default appearance of the map is used. In this case some object may not be visible. To edit the file use the edit function (page 25) or use an ASCII-editor. The map display in GPSMapEdit may be changed according to the TYP file with a skin file (see page 14).

A registry-file with the data of the project is written into the folder of the TYP file. If the PRJ file is aimed only to the production of the TYP file, the REG file contains only the reference to the TYP file. The file can be called for example 'CS.prj'. It is absolutely necessary it that the 'productID' and 'familyID' matches with the affiliated set of maps. In the mistake-case, MapSource passes out cryptic error messages or says goodbye without commentary. BaseCamp ignores the set of maps.

PRJ file may be changed using an text editor. It is strongly recommended to the graphical editor of MapTk. That avoids learning the syntax an typing errors.

Peculiarities:

- The program supports colors that can be represented with one 24-bit-RGB-value. In accordance with the specification of some appliances, the representation is on a color-palette of maximum 256 colors however limited, see page 68. A not in the palette occurring color is rounded. Color-characters in patterns 'Color =...', that are not existing in the list of the colors makes the corresponding pixels transparent.
- Points (POI) are allowed to an any rectangular combination from 1 * 1 to 32 * 32 pixel-columns / lines. Larger pictures are trimmed. A good measurement is 10 * 10 or 12 * 12, but also small, for example 4 lines with 7 columns for i.e. a small triangle (mountain). Transparent pixels are allowed. A warning is displayed for POIs without a picture.
- Polylines always have 32 pixels in the TYP file. In the definition in the PRJ file, the size may be smaller or larger. All over 32 is trimmed. Under 32 pixels become repeated as long as 32 is reached. In order to receive clean patterns, the number of the pixels should therefore multiple of 2, 4, 8, 16 or be 32. The number of the lines should not be too big. More than 5 lines cover a lot of the map. An upper limit is not tested.
- Polylines can be defined with at most 2 colors. These polylines are two-colored then. With one color, the line becomes transparent at pixels of the lacking color, like with the symbol for points. Lines defined by 'LineWidth' and 'BorderWidth' cannot use the color 'transparent'.
- Polylines with 'LineWidth' and 'BorderWidth' set to 0 are invisible.
- Patterns in lines are always turned to the direction of the line (orientation bit is set).
- Polygons with pattern have at most 2 colors. With only one color, the surface becomes transparent at the positions of the 2nd color. Polygon without a pattern cannot be transparent.
- The size of text can be modified with 'TextSize=...'.
- The color of the text can be defined with 'TextColor=...'.

3.5 Overview map

Overview maps are used in MapSource only to select the detailed maps to be transferred to the instrument. The overview map is necessary for MapSource / BaseCamp ! An overview map can be build from one or more detailed maps. The map is build from the MP files listed in the current PRJ file in the working directory. The layout of the map is not critical. This map will not be transferred to the GPS receiver but use for overview in MapSource.

- The program assumes correct detailed maps with only the background type 0x4b in the highest level. That is level 4 if 'Levels=5' in header. All detailed maps must have the same level definition.

- All objects in the highest active level of the detailed maps are copied to the overview map. That is level 3 if 'Levels=5' in the detailed maps. The copies are placed in level 0. So the overview map has only 2 levels. 2 levels are sufficient for MapSource. The result may be improved using the script function of MapTk.
- Additional all cities with type < 0x0b are copied to the overview map.
- From ID, name and the extend of each detailed map the definition polygon 0x04a is produced in the overview map. The size of the definition polygon is derived from the max. extend of all map elements, except the background 0x4b.
- 'Preview=Y' is written to the header.
- The first detailed map in the list defines the code page and elevation parameter. The overview map is not transparent.
- The name and ID of the overview map are taken with following priority
  1. from PRJ file definition of overview map or
  2. from the overview map in the list of MP files or
  3. is set to 'Basemap'.

  The name of the map becomes also the name of the file. The name of the overview map should not contain spaces.
- An existing overview map with the same filename is overwritten.

3.6 Export in GPSMapEdit

In GPSMapEdit, the path will input for cgpsmapper before the first export. If one inputs instead the path to MapTK.exe, GPSMapEdit can export directly into an IMG file.

- The complete name of the IMG file is to be declared.
- Export pure POI files is not supported and is ignored.

4 Automatic routing

The automatic routing functions are designed for MP files edited and checked with GPSMapEdit. Using GPSMapEdit does not require knowledge of the syntax details of MP files. <u>A text editor should never be used to change automatic routing related parts of the MP file.</u> The script function of MapTk has some variables to for general setting of road parameter.

4.1 Polylines for routing

Only a few polylines are able to have routing information. The following table is a list of these polylines, the usage of standard roads and and an advice for the non standard.

Code	**Level**	**Usage**	**Speed**	**Class**	**Restrictions** 0 1 2 3 4 5 6 7 8
01	3	Major highway	6	4	0, 0, 0, 0, 0, 0, 1, 1, 0
02	3	Principle highway	5	3	0, 0, 0, 0, 0, 0, 0, 0, 0
03	3	Other highway	4	3	0, 0, 0, 0, 0, 0, 0, 0, 0
04	3	Arterial road	4	2	0, 0, 0, 0, 0, 0, 0, 0, 0
05	3	Collector road	4	2	0, 0, 0, 0, 0, 0, 0, 0, 0
06	2	Residential street	3	1	0, 0, 0, 0, 0, 0, 0, 0, 0
07	2	Paved road	2	0	0, 0, 1, 1, 1, 1, 0, 0, 1
08	2	Highway ramp	3	1	0, 0, 0, 0, 0, 0, 0, 0, 0
09	1	Highway ramp	3	1	0, 0, 0, 0, 0, 0, 0, 0, 0
0A	1	Unpaved road	2	0	0, 0, 1, 1, 1, 1, 0, 0, 1
0B	3	Major highway	3	2	0, 0, 0, 0, 0, 0, 1, 1, 0
0C	2	Roundabout	2	1	0, 0, 0, 0, 0, 0, 0, 0, 0
0D		Trail / track [5]	1	0	0, 1, 1, 1, 1, 1, 0, 0, 1
0E	1	Not visible (e.g. parallel cable car path)	1	0	0, 1, 1, 1, 1, 1, 0, 0, 1
0F	1	Marked trail	1	0	0, 1, 1, 1, 1, 1, 0, 0, 1
10	1	Cycle path	1	0	0, 1, 1, 1, 1, 1, 1, 0, 1
11	1	Bridle path	1	0	0, 1, 1, 1, 1, 1, 0, 0, 1
12	1	Difficult track	1	0	0, 1, 1, 1, 1, 1, 0, 1, 1
13	2	Pedestrian area	1	0	0, 1, 1, 1, 1, 1, 0, 1, 1
16	1	Trail / track	1	0	0, 1, 1, 1, 1, 1, 0, 0, 1
1A	2	Ferry	1	2	0, 0, 0, 0, 0, 0, 0, 0, 0
1B	2	Ferry	1	2	0, 0, 0, 0, 0, 0, 0, 0, 0

Yellow: Types of polylines for automatic routing (Script: list ROUTING).

Blue: Example for usage of the none standard roads.

Columns 'Level', 'Restrictions' and 'Class' are examples for a topographic map.

5 'Endlevel' ignored in GPSR

Speed class	avg. km/h
0	5
1	20
2	40
3	60
4	80
5	90
6	110
7	

Index	Restriction
0	is toll road
1	no emergency
2	no delivery
3	no car / motorcycle
4	no bus
5	no taxi
6	no pedestrian
7	no bicycle
8	no truck

4.2 Edit the map

The map must have a background polygon Type=0x4b !

For best results activate 'Snap to grid' and 'Stick to neighbors' in menu 'Tools → Options → Edit' in GPSMapEdit.

Routing starts with a complete map containing all roads and tracks. To add information for automatic routing the MP file must contain some extra information:

- 'Routing=Y' in the header.
- 'RoadID=...' for each road (polyline, → GPSMapEdit)
- 'Nodx=...' for each connection or ends of roads (routing nodes, → GPSMapEdit).

The MP file may contain optional information to control the routing behavior:

- 'RouteParam=...' for the behavior of each road (see tables above, → GPSMapEdit).
- Blocks [Restrict] … [End] to define turn restriction for routing nodes (→ GPSMapEdit). Here defined restrictions are not for pedestrians. Time dependent restrictions are not supported.

GPSMapEdit will write this information as correct MP statements while editing the map. Get the GPSMapEdit manuals and help and find out the details.

A complete map or set of maps requires following steps in GPSMapEdit to make it fit for automatic routing:

1. Generate a routing graph in GPSMapEdit ('Tools → Generate Routing Graph → Using Coinciding Points of Polylines'). Connections are only set as a routing node if the points of a polyline are at the same coordinates.
2. Run the check of the routing graph: 'Tools → Verify Map …' with all options 'Routing Graph' on. <u>This function should be called after any changing the routing graph.</u>
3. Remove all reported problems until 'No invalid objects found' is reported. Invalid object are e.g. self intersection roads or double routing nodes. Not connected ends of a road need a routing node. A road cannot end at the start coordinates. Missing junctions are not detected.

4. Mark all ends of a road along the border of the tile as 'external'. This are the points where the route will continue on next tile.

While creating a map or adding new roads connect the roads in context menu of polylines: 'Connect to Nearest Nodes'. Avoid near points on roads on bridges / tunnels and crossing roads.

4.3 Solving routing problems

Premised that all classification and restrictions of roads are as expected there are 2 groups of problems left to be solved:

- Many roads are not correct connected because the points do not match exactly.
- The external nodes do not match.

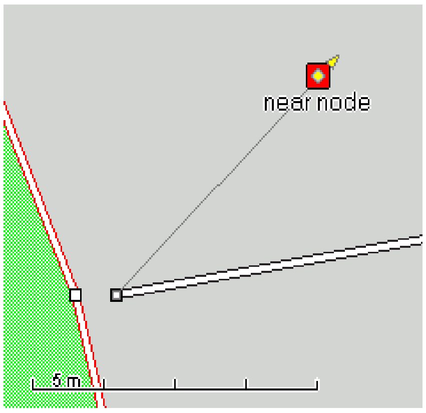
MapTk will help to find the missing connections.

Compile the tiles of the project. Normally the compilation will report no problems if the map was verified without reporting invalid objects. But testing the map in MapSource will show a lot, sometimes hundreds of problems.

MapTk has 2 functions in the menu 'GPX/MP' to find the not matching points:

- 'Find not connected near roads': All not connected nodes near to an other road but not connected are marked. 'Near' is a parameter in meters. Normally all nodes up to 3 m can be connected without checking the surrounds. Larger distances should be checked to avoid connecting not existing junctions. See page 13.
- 'Find not matching external nodes': This function works on all IMG files of the project and generates a MP file containing markers for each misplaced or not connected node over the whole project. To use this the project must be compiled before ('Make'). See page 13.

The result of both functions is a MP file with bookmarks for each not connected node. The text of a bookmark is 'near node' for not connected near nodes and the Ident of he tile for not connected external nodes.



Add the file with the markers to the tile and correct all problems manually (automatic connections seems too dangerous). Selecting the road first allows to move the node inside the marker-POI. Before saving the MP file the markers can be removed (no problem if not, bookmarks are ignored by the compiler).

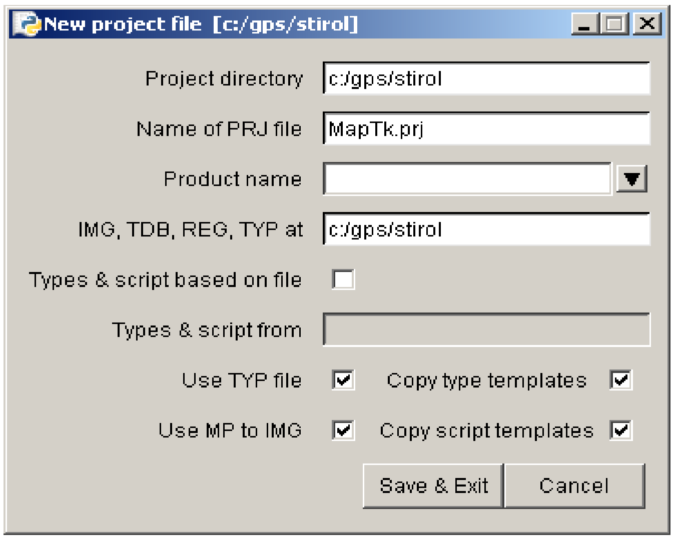
Modification of project header or Windows registry are not necessary. If no problem is left (invalid object in GPSMapEdit, not connected routing nodes and all external nodes are matching) the automatic routing should work perfect.

5 Editor

The editor is splitted into 6 specialized parts. Each part is select by tabs in the editor window. 'Exit' saves the changes in the PRJ file and terminates editing. POIs, polylines and polygons are written sorted to the file.

5.1 New PRJ file

Calling the editor before a PRJ file was taken to the working directory (or file menu: 'New project file') MapTk asks what kind of project file has to be created.



New project file [c:/gps/stirol]

Project directory	c:/gps/stirol		
Name of PRJ file	MapTk.prj		
Product name			
IMG, TDB, REG, TYP at	c:/gps/stirol		
Types & script based on file	☐		
Types & script from			
Use TYP file	☑	Copy type templates	☑
Use MP to IMG	☑	Copy script templates	☑

A new product must have a name.

The combo box 'Product name' contains all sets of maps registered for MapSource. In this case the project data are taken from selected set of maps. No more data must be entered here.

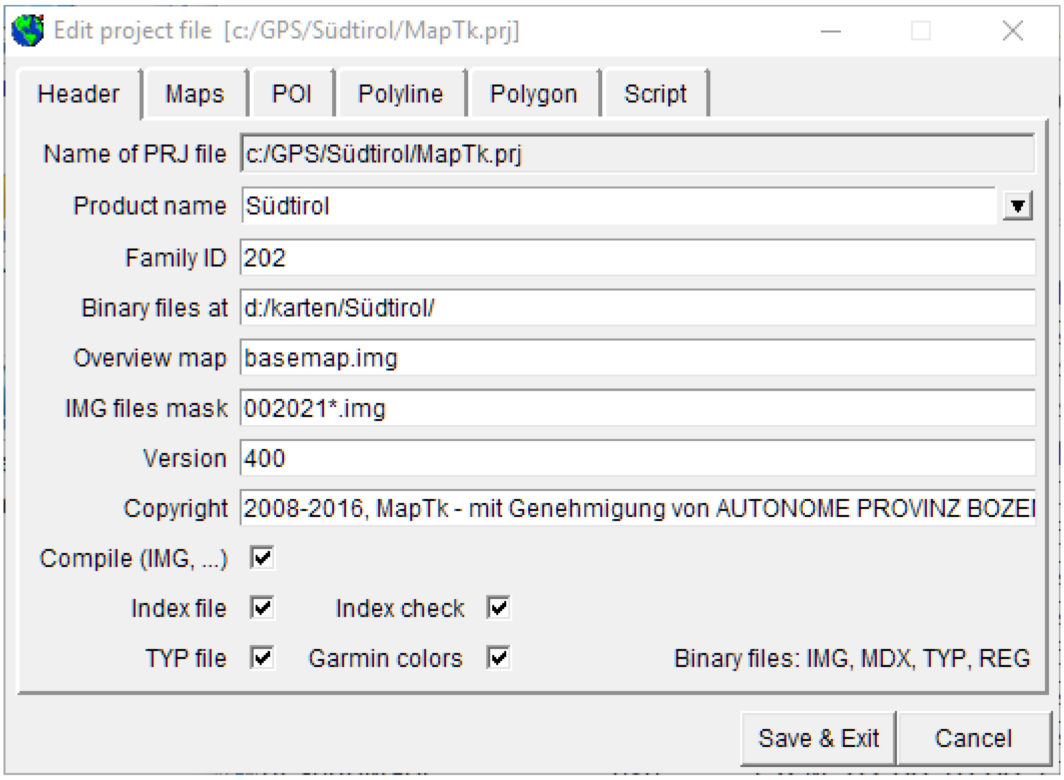
The field 'IMG, TDB, REG, TYP in' defines the folder for those files (black triangle right). If the folder is not changed all files will be located in a single folder, the working folder.

Checking 'Types & script based on file' allows to use the data for the TYP file and scripting from an existing PRJ file - entered at 'Types and scripts from'. In this case the following check boxes are disabled. 'Use TYP file' builds a path and the name for the TYP file. To copy layout examples to the PRJ file check 'Copy TYP templates'. 'Use MP to IMG' tries to build the name for the overview map and the location of the IMG files as well as the mask for the IMG files from the contents of the folders. Check 'Use scripting templates' if you want examples for the scripting part.

Except the name of the PRJ file all data entered here can be changed in the project editor (see next chapter).

5.2 Header

The header tab defines the project parameters. The data is taken from the Windows registry if 'Product name' was chosen from the list.



Edit project file [c:/GPS/Südtirol/MapTk.prj]

Name of PRJ file	c:/GPS/Südtirol/MapTk.prj
Product name	Südtirol
Family ID	202
Binary files at	d:/karten/Südtirol/
Overview map	basemap.img
IMG files mask	002021*.img
Version	400
Copyright	2008-2016, MapTk - mit Genehmigung von AUTONOME PROVINZ BOZEI

Compile (IMG, ...) ✓

Index file ✓ Index check ✓

TYP file ✓ Garmin colors ✓ Binary files: IMG, MDX, TYP, REG

The name of the product may be changed or the data of an existing product may be used (button on the right),

The family ID must be entered (1 … 65535) and must be different from all other maps used in MapSource.

'Binary files at' defines the folder for IMG, MDX, MDR, TYP and REG files. The field may be empty or contain './' to use the working folder ('all in one folder'). The name of MDX, MDR, TYP and REG files is derived from the family ID. E.g for FID = 202 we have

```
M00202_mdr.img
M00202.TDB
M00202.MDX
M00202.TYP
M00202.REG
```

'Overview map' defines the name of the overview map. The folder for this file like the binaries.

'Version' and 'Copyright' are use in the binary files.

'Compile (IMG, …)' must be activated to produce a complete set of maps.

'Index file' is activated to have an index (*_mdr.img). If this function is not activated but an index is already stored an empty index is produced instead. This guaranties the consistency without changing the registry.

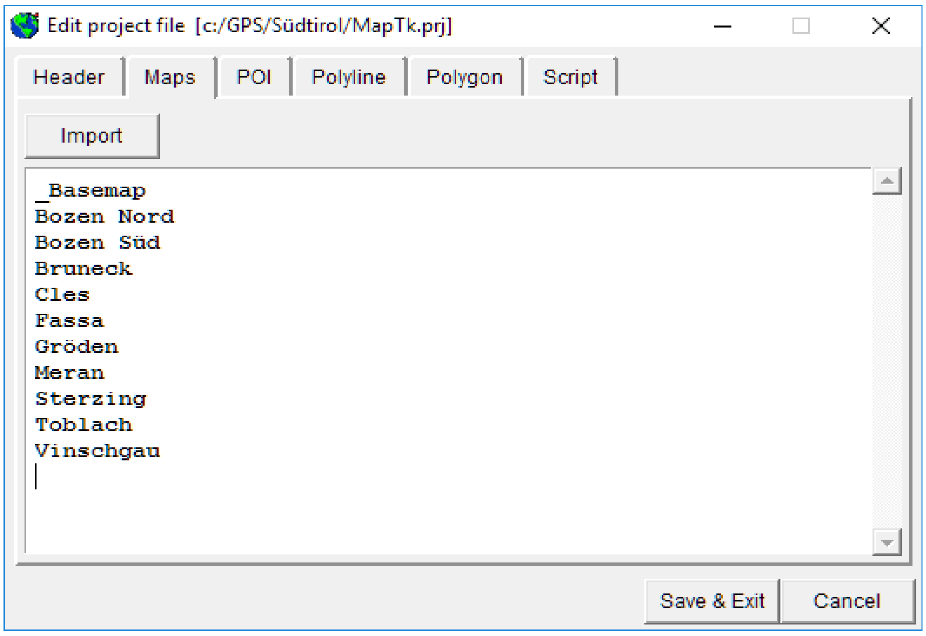
'Index check' activates checking the consistence and completeness of the data for the index. The flag is use for scripting and building the index file.

If index, TDB file and the tiles are not compatible problem showing the map with MapSource or the GPS receiver will happen.

To have a TYP file the check box is activated.

'Garmin colors' is selecting one of two palettes for the TYP file. Either the colors used until version 3.1.2 (checked, default from table page 68) or new and brighter colors. See page 68.

5.3 Maps

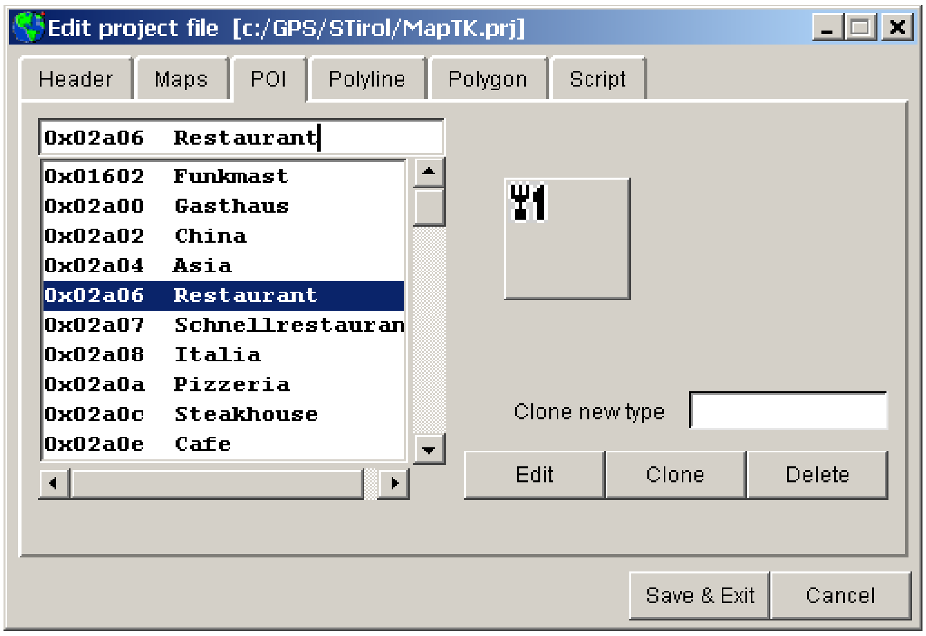


```
_Basemap
Bozen Nord
Bozen Süd
Bruneck
Cles
Fassa
Gröden
Meran
Sterzing
Toblach
Vinschgau
```

The button 'Import' writes the name of all MP files in the working directory into a sorted list. File name starting with '#' are not added to the list. Import overwrites any former contents of the list. The list is presented and altered with a simple text editor.

5.4 Select window

For POIs, polylines and polygons a common window layout is used. The list of objects is build from the type and the first string. The symbol for the selected object is shown.



Edit project file [c:/GPS/STirol/MapTK.prj]

Header | Maps | POI | Polyline | Polygon | Script

```
0x02a06   Restaurant

0x01602   Funkmast
0x02a00   Gasthaus
0x02a02   China
0x02a04   Asia
0x02a06   Restaurant
0x02a07   Schnellrestauran
0x02a08   Italia
0x02a0a   Pizzeria
0x02a0c   Steakhouse
0x02a0e   Cafe
```

Change an object: All object of the type in the PRJ file are listed. The editing start with selecting of an object and click on 'Enter' or a double click on the line in list.

Create a new object: Enter the new type in the entry field above the list (decimal or hexadecimal) press 'Return' key or click on 'Edit'.

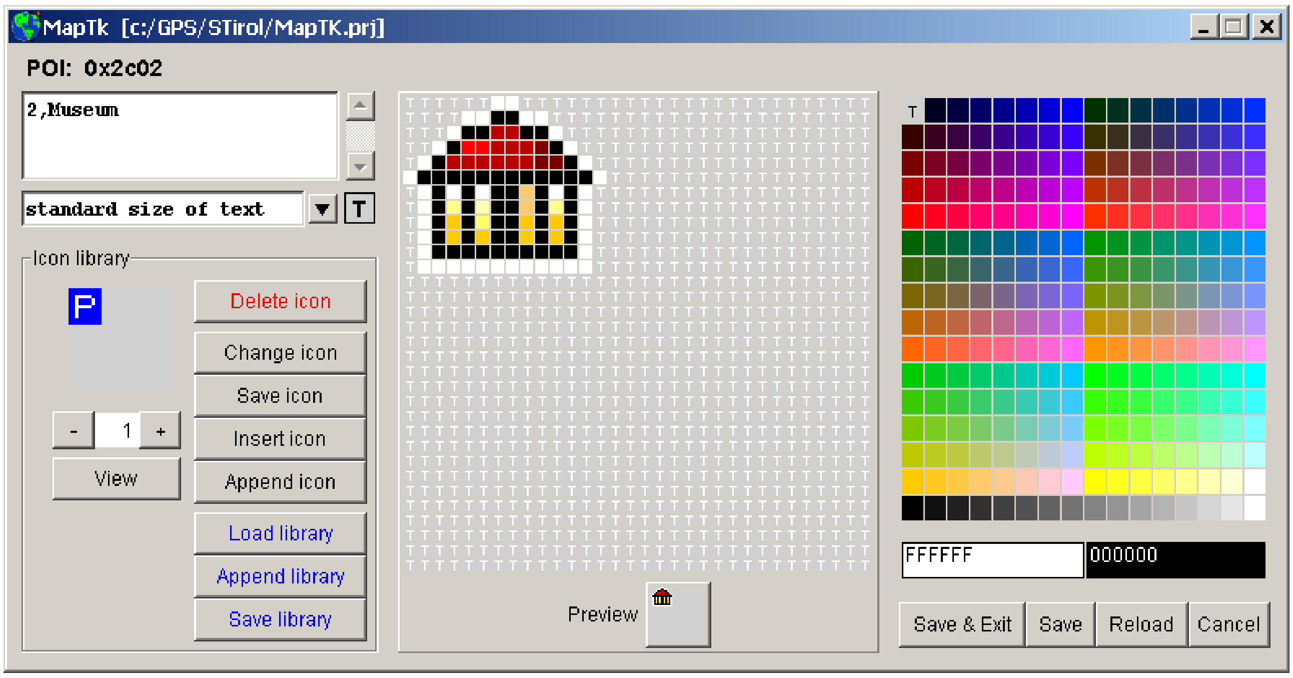
Copy an object: The data can be copied to a new object (type must be changed) by typing the new type in the entry field 'Clone new type' and click on 'Clone'.

Delete an object: An object is deleted clicking on 'Delete'.

Starting the editor opens a window, with 4 different areas. The general function is similar for POIs, polylines and polygons.

5.5 POI

See also page 13.


MapTk [c:/GPS/STirol/MapTK.prj]

POI: 0x2c02

```
2,Museum
```

standard size of text

FFFFFF 000000

POI editor

5.5.1 Upper left area

A simple text editor for the strings (functions like Ctrl-C, Ctrl-Z, ... are supported). One line per language containing the known country code (see page 69) separated by ',' from the describing text. The total text cannot exceed a maximum of 125 characters if 1 language, 123 if 2 languages, 121 if 3 languages ...

The size of the displayed text can be chosen in the combo box below. 'standard size of text' is the default and the size is defined by Garmin.

The field right defines the text color. Click right or left copies the color. Garmin standard colors are used if 'transparent' is selected, this is the default.

5.5.2 Lower left area

Access to the contents of the icon library:

- 'Change icon' copies the icon to the drawing area. The former picture is destroyed.
- 'Save icon' the picture on the drawing area replaces the icon in the icon library.
- 'Insert icon' inserts the icon on the drawing in the icon library.
- 'Append icon' appends the icon on the drawing to the icon library.
- 'Delete icon' deletes the shown icon from the library.
- 'View' generates a page with all icons in the icon library. This page may be exported as a postscript file for printing.
- '+' / '-' increments / decrements the icon number (shift: 10 times). Input of a number selects an icon directly.

The icon library is loaded together with MapTk and stored while terminating MapTk. The library can be processed:

- 'Load library' load a library from file. The current library in memory is overwritten.
- 'Append library' adds a library or a bitmap (BMP, GIF, ICO, JPG, PNG, TIF) to the icons in memory. Already loaded icons are not appended.
- 'Save library' writes the icons in memory into a file.

Deleting or changing an icon must be confirmed.

5.5.3 Middle area

Drawing the icon for the POI. Click on one of the 32 * 32 pixel will change the color of this pixel. All 255 colors can be used. The color is taken from the color palette. With 'T' marked grey pixels are transparent. A POI is completely transparent if all pixel are transparent.

- Left mouse button: color from the left color field (current color).
- Right mouse button: color from the right color field (current color).
- Holding the button while moving the mouse: draw a freehand line.
- Control and a mouse button: color from drawing area becomes the current color.
- Shift and a mouse button: color in the drawing is changed to the current color.
- Control + cursor keys moves the picture pixel by pixel

The preview field shows the icon in the original size.

5.5.4 Right area

Color picking area. Clicking a color using the left or right mouse button changes the current color. The two fields are showing the current colors together with the RGB value (hex) or the word 'transparent' if the color stands for transparent. The transparent (RGB = D0D0D0) color is located in the upper left corner of the palette.

5.5.5 4 buttons

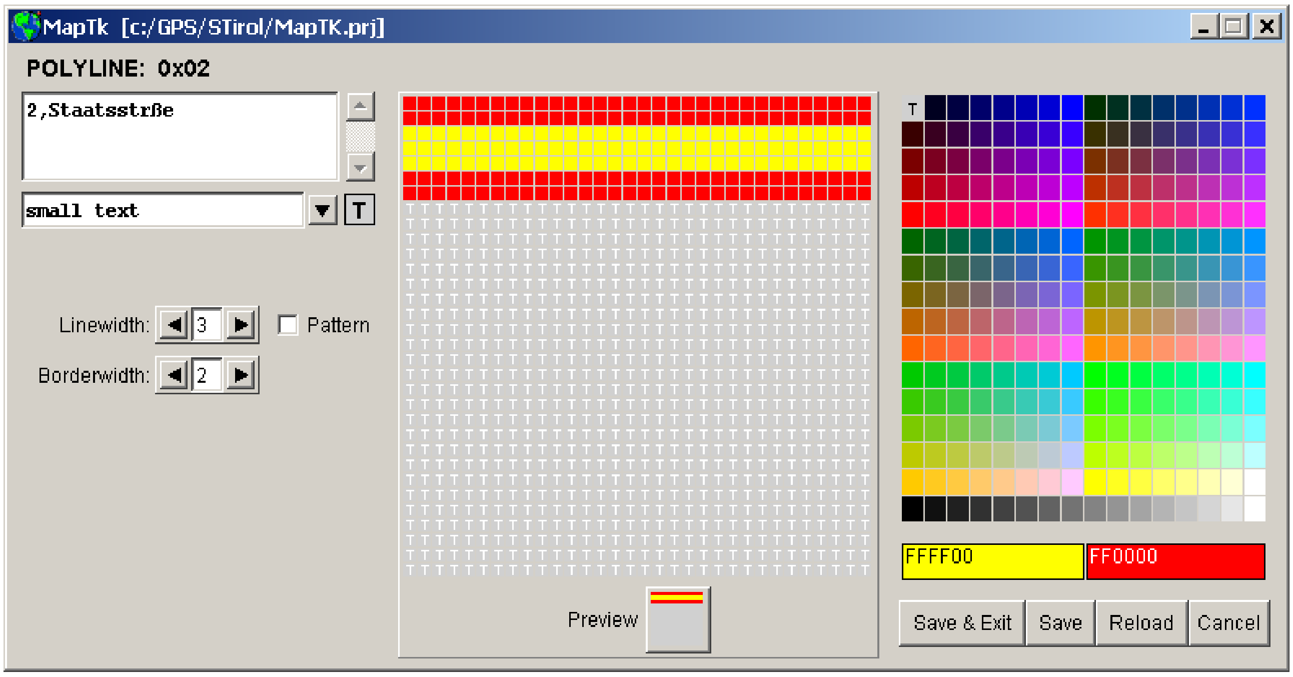
'Exit' saves all changes an terminate editing.

'Save' stores the changes and continues editing.

'Reload' restores the POI data,

'Cancel' leaves the editor, changes are discarded.

5.6 Polyline



Polyline editor

5.6.1 Upper left area

A simple text editor for the strings (functions like Ctrl-C, Ctrl-Z, ... are supported). One line per language containing the known country code (see page 69) separated by ',' from the describing text. The total text cannot exceed a maximum of 125 characters if 1 language, 123 if 2 languages, 121 if 3 languages ...

The size of the displayed text can be chosen in the combo box below. 'standard size of text' is the default and the size is defined by Garmin.

The field right defines the text color. Click right or left copies the color. Garmin standard colors are used if 'transparent' is selected, this is the default.

5.6.2 Lower left area

'Pattern' not checked: line width (0 ... 12) and the border width (0 ... 10) is defined here. LineWidth and BorderWidth set to 0 makes a line invisible. The line style is displayed in the drawing area.

'Pattern' is checked: The width and the height of the drawing area is defined. The pixels in the drawing are are repeated until the line is 32 pixel long. Up to 2 colors (including transparent ' are possible.

5.6.3 Middle area

'Pattern' not checked: display of the line only. Clicking on the line or the border will change to color of the line directly.

- Left or right mouse key: the color of the left/right color field (actual color) is taken for the line or the border. Transparent is not possible.

- Control and a mouse key: the color of the pixel in the drawing area is taken as the actual color.

'Pattern' is checked: defining of the pattern in the drawing area (white borders). If the drawing area contains already the maximum of 2 colors, the color of the clicked pixel will be changed for the whole polyline to the current color. Similar to editing a polygon.

5.6.4 Right area

Color picking area. Clicking a color using the left or right mouse button changes the current color. The two fields are showing the current colors together with the RGB value (hex) or the word 'transparent' if the color stands for transparent. The transparent color is located in the upper left corner of the palette.

5.6.5 4 buttons

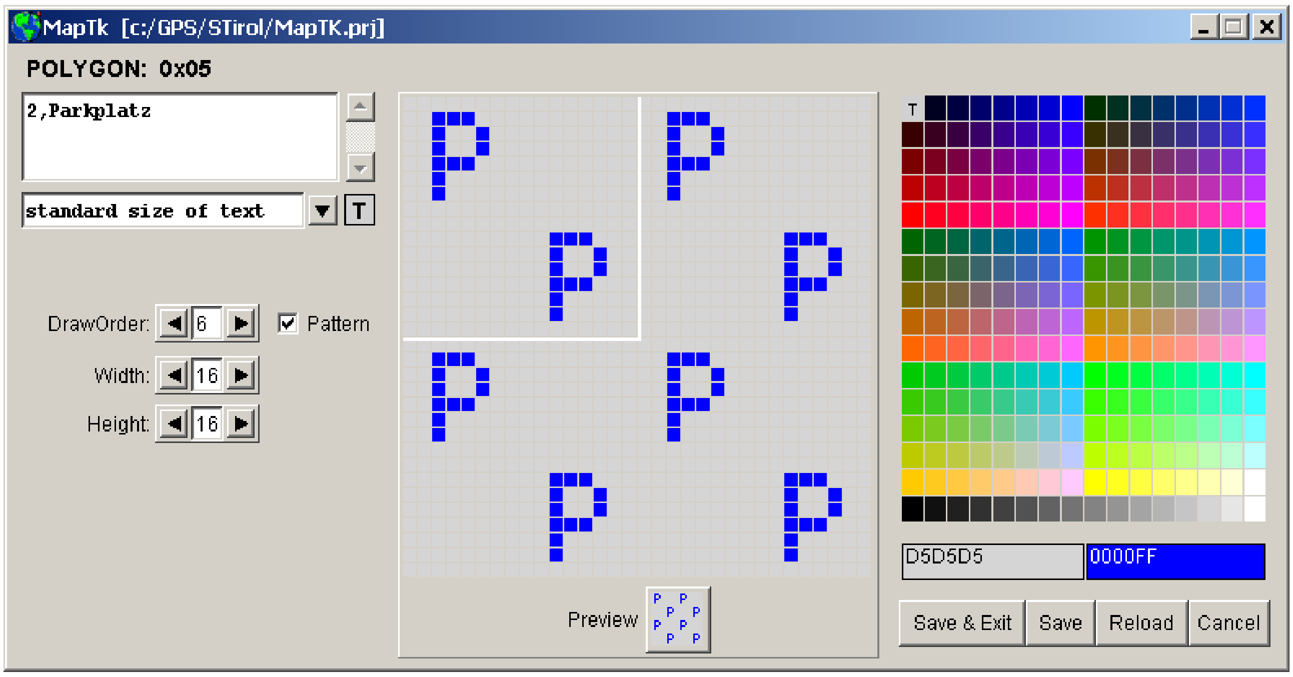
'Exit' saves all changes an terminate editing.

'Save' stores the changes and continues editing.

'Reload' restores the polyline data,

'Cancel' leaves the editor, changes are discarded.

5.7 Polygon



Polygon editor

5.7.1 Upper left area

A simple text editor for the strings (functions like Ctrl-C, Ctrl-Z, ... are supported). One line per language containing the known country code (see page 69) separated by ',' from the describing text. The total text cannot exceed a maximum of 125 characters if 1 language, 123 if 2 languages, 121 if 3 languages ...

The size of the displayed text can be chosen in the combo box below. 'standard size of text' is the default and the size is defined by Garmin.

The field right defines the text color. Click right or left copies the color. Garmin standard colors are used if 'transparent' is selected, this is the default.

5.7.2 Lower left area

The draw order of the polygon is defined here in the range 0 ... 7.

'Pattern' not checked: No further data to enter.

'Pattern' is checked: The width and the height of the drawing area is defined.

5.7.3 Middle area

'Pattern' not checked: display of the color only. No editing. Clicking with a mouse key will change the color of the whole are to the current color directly.

'Pattern' is checked: defining of the pattern in the drawing area (white borders). If the drawing area contains already the maximum of 2 colors, the color of the clicked pixel will be changed for the whole polygon to the current color. Similar to edition a polyline.

5.7.4 Right area

color picking area. Clicking a color using the left or right mouse button changes the current color. The two fields are showing the current colors together with the RGB value (hex) or the word 'transparent' if the color stands for transparent. The transparent color is located in the upper left corner of the palette.

5.7.5 4 buttons

'Exit' saves all changes an terminate editing.

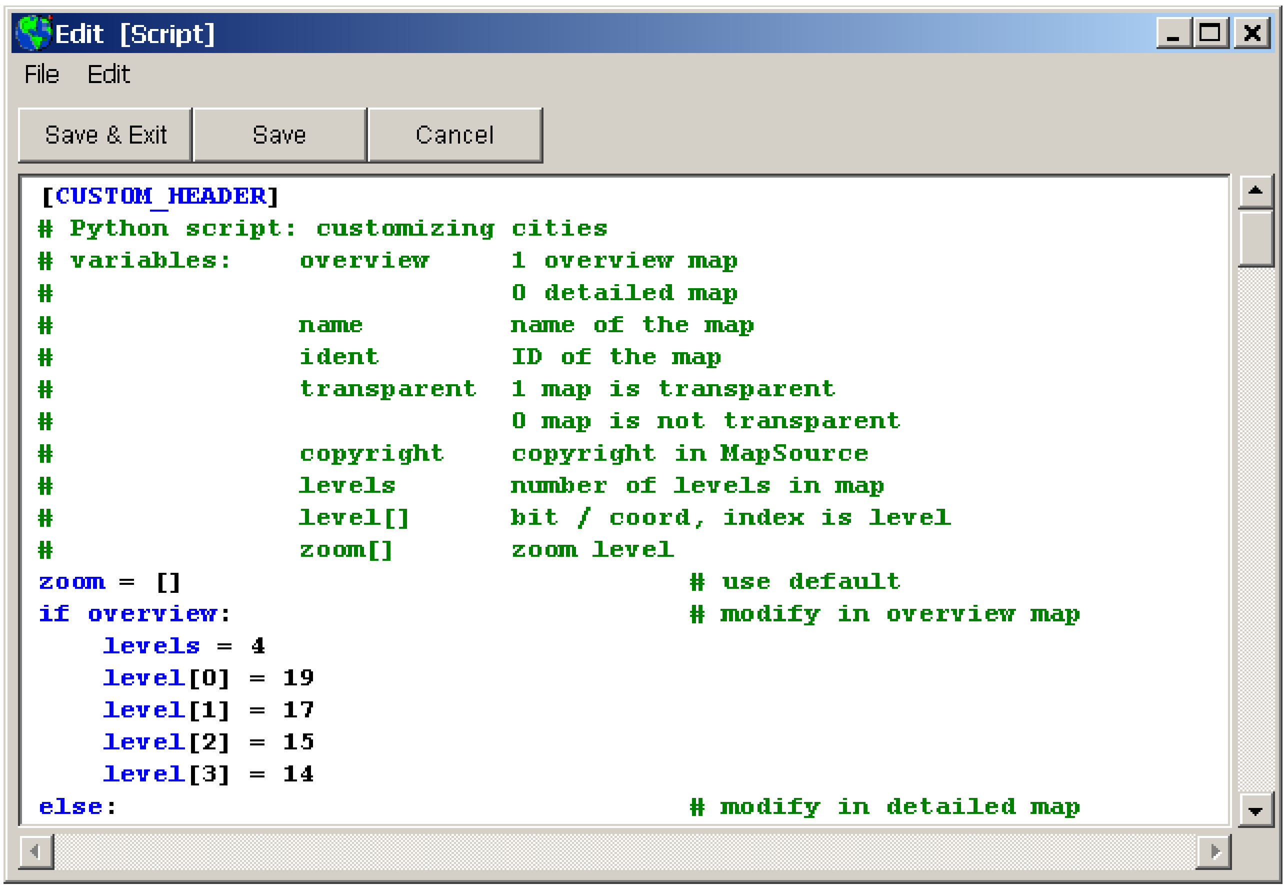
'Save' stores the changes and continues editing.

'Reload' restores the POI data,

'Cancel' leaves the editor, changes are discarded.

5.8 Script

The button 'Exit' saves the script in memory. The PRJ file is updated before termination the editor function. 'Save' writes the contents back and stays in the editor. 'Cancel' terminates, changes are discarded.



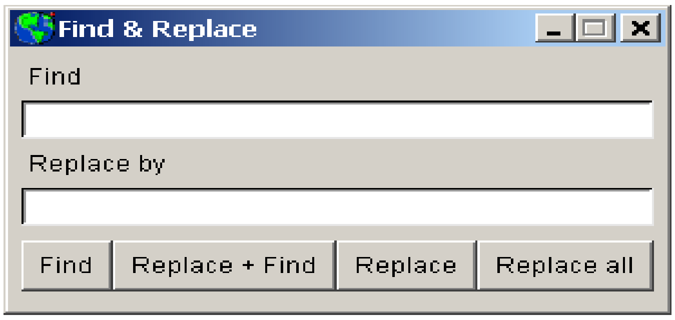
```
[CUSTOM_HEADER]
# Python script: customizing cities
# variables:     overview      1 overview map
#                              0 detailed map
#                name          name of the map
#                ident         ID of the map
#                transparent   1 map is transparent
#                              0 map is not transparent
#                copyright     copyright in MapSource
#                levels        number of levels in map
#                level[]       bit / coord, index is level
#                zoom[]        zoom level
zoom = []                                 # use default
if overview:                              # modify in overview map
    levels = 4
    level[0] = 19
    level[1] = 17
    level[2] = 15
    level[3] = 14
else:                                     # modify in detailed map
```

5.8.1 Functions

- Edit of ASCII-files
- Syntax highlighting for PRJ files
- Find - and Replace-Function
- see 'Menu' and 'Keys'

5.8.2 Menu

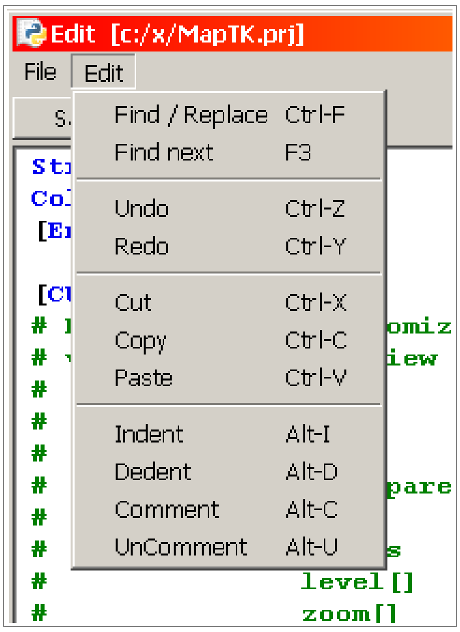
- 'File'
    - 'Save' stores the file. The prior version is overwritten. The editor remains opened. The button 'Save' has the same function.
    - 'Exit' finishes the editor. If the loaded text should have been changed, a query takes place: Storage yes/no.
    - 'Cancel' terminates. Changes are lost.
- 'Edit'
    - 'Find / Replace'



        - 'Find': It is sought for a text beginning at the cursor-position.
        - 'Replace + Find': Text is replaced and the next position is searched.
        - 'Replace': Text is replaced.
        - Replace all': The entire text is replaced. Cursor is not moved.
    - 'Finds next' repeats the searching with the same text.
    - 'Undo' does the last alteration at the text declining.
    - 'Redo' produces declining alteration done again.
    - 'Cut' cuts out marked text into the clip board.
    - 'Copy' copies marked text into the clip board.
    - 'Paste' inserts text from the clip board.
    - 'Indent' indents the text.
    - 'Dedent' disengages the text.
    - 'Comment' writes a commentary '#' to the line.
    - 'Uncomment' activates a comment line.

5.8.3 Keys

- The Cursor-Block has the usual function
- Marking with the mouse or 'Shift-Cursor'
- 'Control-C' same as menu 'Copy'
- 'Control-F' same as menu 'Find / Replace'
- 'F3' same as menu 'find next'
- 'Alt-I' same as menu 'Indent'
- 'Alt-D' same as menu 'Dedent'
- 'Control-V' same as menu 'Paste'
- 'Control-X' same as menu 'Cut'
- 'Control-Y' same as menu 'Redo'
- 'Control-Z' same as menu 'Undo'
- 'Control-End' go to the end of the file
- 'Control-Pos1' go to the start of the file
- 'Alt-C' same as menu 'Comment'
- 'Alt-U' same as menu 'Uncomment'
- 'Mouse-right' is 'Edit'-Menü as PopUp.



6 Files

The program requires, as well as generates different files.

6.1 MapTk.dat

This file is new in version 2.0. 'MapTk.dat' shares the program directory with 'MapTk.exe' and 'MapTk_*.pdf'. The file contains the icon library and some configuration data. The file is read after starting MapTk and is written back before MapTk terminates. It is a compressed binary file not to be used directly by the user. If the file is missing MapTk displays a warning and creates an empty icon library together with default configuration data. Only prior to version 2.0 the file is part of the distribution archive. For later versions an icon library can be downloaded separately and imported. This file should be mentioned in a back up concept.

6.2 PRJ file

With version 2 the format was changed. Since version 2.7 the header of the file is changed. A conversion is performed invoking the built in editor. Old versions of the PRJ file are accepted but not written.

<u>It is recommended to change the file only with the built in editor. A new file is created also with the editor.</u> Using the editor spares learning the syntax an typing errors.

In the project-file * .PRJ is stored all information to produce a set of maps (examples for download are available). The structure of the PRJ files is leaned at the MP-Format. The file is subdivided in blocks [...]. The name of a block can be written in small or capital letters. One block becomes completed with [END].

The information is stored with key-words. A key can be written in small or capital letters. Not supported keys are shown as warning and are ignored.

With texts, the string delimiter characters (' ") are allowed to left out. In this case, the space character is not used. The string delimiters, comma (,) and the commentary-character (#) are not allowed within texts. All text behind '#' is ignored. The built in editors (except 'Script') are not writing back comments.

If mistakes should appear in the script, the shown line-numbers are counted from begin of the block.

In file- and path-statements can instead of '\' also '/' be written. The REG file is stored in the folder of the TDB file or in the folder of the TYP file, preferably stored as '<product name> .reg'.

- [Project] Definition of the project, the data is inputed here in the dialog-window. In clamps {...} the name of the value
    - Name Name of the PRJ file, which is stored in the working folder.

- Product name {Product} allows to aim a new project or in Windows operation system to select from the set of maps registered already. The data of the registry is adopted into the input-fields. In the case of a new set **all** data must be entered. With the REG file generated later, the new set can be registered in MapSource using these data.
- Family ID {FamilyID} Is used for TDB-, TYP file and in the Windows registry as ID. Example: 200. Avoid conflict with other FamilyID !. Product ID is set to 1.
- Compile {Compile} controls compiling of the bonary files.
- Index {Index} controls index generation.
- TYP file {Style} controls compiling of TYP file.
- Overview map {OverviewMap} folder and name of the overview map.
- IMG files at {IMGPath} folder for the compiled maps, used by MapSource. The field may be empty or containing './' to use the working folder.
- IMG files mask {IMGfiles} Name of the IMG files in 'IMGPath' (Example: 0*.img for all IMG files, starting with '0'. The overview-map must not match these pattern):
- Version {Version} Without important function, marks the alteration-state of the set of maps. Is shown in MapSource at 'map product information'.
- Copyright {Copyright} The text is written optional into the TDB file and shown in MapSource at 'Help → Productinformation'.

- [IMG] List of the MP files, that should be converted to IMG files. The names are per line without keys, '=' and file extension. Example: 'Bozen'. Also the overview-map is written down here. Line starting with semicolon (';') are comment.
- [POI] Definition of Icons for a point in the map. The block is repeated for all types of POI. POIs of the type > 0x10000 **must** be declared in the PRJ file to be displayed at all. List of this POIs at page 65.
    - Type= Type and subtyp as 16-bit-value of the point, example: *Type=0x2f0b* for parking. The value is decimal or hexadecimal. 32 subtypes are allowed from 0x00 to 0x1f. So for services 0x2f00 up to 0x2f1f is possible. If type < 0x100 a subtyp 0x00 is assumed. Additional leading 3rd byte 0x01 specifies user defined POIs.

● String=	Optional description of the POI if no explicit description was declared, example: *String=4,Parking*. '4' is the code for a language. The description is shown instead of the default of the appliance / MapSource. *String=4," "* or *String=4,' '* is possible to overwrite default labels, but not explicit labels in the IMG file.
● TextSize=	0: standard (defined by Garmin, default), 1: no text, 2: small, 3: normal, 4: large.
● TextColor=	Color used for the text, example: *TextColor=0x0000ff* for blue. If this line is missing the default color is used.
● Color=	A sign is assigned to a color for one pixel of the picture, example: *Color=+,0xff0000* for red.
● Line=	Definition of one line of the picture, example: *Line*=--++--. The sign becomes the color from the color-table [Colors] assigned. If no color is defined for a sign, the pixel is represented transparently. The size of a symbol is 1 * 1 up to 32 * 32 pixels.
● [Polyline]	Definitions for polylines. The block is repeated for all types of polylines. Polyline of types >= 0x10000 **must** be declared in the PRJ file to be displayed at all ! List of this polylines at page 65.
● Type=	Type as 8- or 16 bit-value for the line, example: *Type=0x01* for highway. The value is decimal or hexadecimal. Additional leading 3rd byte 0x01 specifies user defined polyline.
● String=	Optional description of the POI if no explicit description was declared, example: *String=4,Highway*. '4' is the code for a language. The description is shown instead of the defaults of the appliance / MapSource. *String=4," "* or *String=4,' '* is possible to overwrite default labels, but not explicit labels in the IMG file.
● TextSize=	0: standard (defined by Garmin, default), 1: no text, 2: small, 3: normal, 4: large.
● TextColor=	Color used for the text, example: *TextColor=0x0000ff* for blue. If this line is missing the default color is used.
● LineWidth=	width of the line in pixel, example: *LineWidth=2*.
● BorderWidth=	Width of the border in pixel, example: *BorderWidth=1*. No statement means no border. If an border becomes defined no color to it however, the border is black.

- Color= Up to 2 colors can be defined, example: *Color=1,0x00ff00*. If for the line no pattern is declared, the first color is the line itself, that 2nd color the border. If a pattern is defined the color is set corresponding to the signs.
- Line= Definition a line of each polyline, example: *Line*=++++----. The sign becomes assigned the color from the color-table [Colors]. If no color is defined for a sign, the pixel is represented transparently. If the definition is less than 32 pixels the pattern repeats until 32 pixels are existing. Several parallel lines can be defined for a polyline.

- [Polygon] Definitions for polygons. The block is repeated for all types of polygon. Polygons of the type >= 0x10000 must be declared in the PRJ file to be displayed at all ! List of this polygons at page 66.
  - Type= Type as 8- or 16-bit-value of the polygon, example: *Type=0x0c*. The value is decimal or hexadecimal. Additional leading 3rd byte 0x01 specifies user defined polygon.
  - DrawOrder= The value (0 ... 7) declares the sequence drawing the polygon. A higher value covers Polygons with lower value. All Polygons **must** be classified with this line in the display-sequence. Exception: the draw order for the definition polygon 0x4b is ignored.
  - String= Optional description of the POI if no explicit description was declared example: *String=4,Industry*. '4' is the code for a language. The description is shown instead of the default of the appliance / MapSource.
    String=4," " or
    String=4,' '
    is possible to overwrite default labels, but not explicit labels in the IMG file.
  - TextSize= 0: standard (defined by Garmin, default), 1: no text, 2: small, 3: normal, 4: large.
  - TextColor= Color used for the text, example: *TextColor=0x0000ff* for blue. If this line is missing the default color is used.
  - Color= Up to 2 colors are allowed, example: *Color=1,0x0000ff.* If for the polygon no pattern is declared, the polygon is filled with the first color, the second color is ignored. If a pattern is declared, the colors are put in the pattern accordingly.
  - Line= Definition of a line of each polygon, example: *Line*=++++----. If no color is defined for a sign, the pixel is represented transparently. Several lines can be defined for a polyline. If the definition is less than 32 pixels the pattern is repeated until 32 pixels are existing. This pattern should therefore have 2, 4, 8, 16 or 32 pixels / line or lines.

- [Custom_HEADER] Python-code to adapt map properties. Header data of the project file (*.prj) can be read in [CUSTOM_HEADER]. Useful indices of project[] are 'product', 'copyright', 'version' and 'fid'.
- [Custom_POI] Python-code to adapt data in RGN10.
- [Custom_Polyline] Python-code to adapt data in RGN40.
- [Custom_Polygon] Python-code to adapt data in RGN80.
- [End] terminates a block.

6.3 PRJ file and the functions

Block / Line	**Function**			
	Script	**IMG**	**Make**	**TYP**
[Project]	**read**[6]	**no**	**yes**	**yes**
Product=			yes	yes
FamilyID=			yes	yes
Overview=			yes	no
IMGpath=			yes	no
IMGfiles=			yes	no
Compile=			yes	no
Index=			yes	no
TDB=			yes	no
Style=			optional	yes
Version=			yes	no
Copyright=			optional	no
[IMG]	**no**	**no**	**yes**	**no**
<mapnname>			1 line / map	1 line / map
[POI]	**no**	**no**	**optional**	**optional**
Type=			yes	yes
String=			optional	optional
TextSize=			optional	optional
TextColor=			optional	optional
Color=			>= 1	>= 1
Line=			>= 1	>= 1
[Polyline]	**no**	**no**	**optional**	**optional**
Type=			yes	yes
String=			optional	optional
TextSize=			optional	optional
TextColor=			optional	optional

6 Data in project[], to be used in [CUSTOM_HEADER].

Block / Line	Function			
	Script	IMG	Make	TYP
LineWidth=			optional	optional
BorderWidth=			optional	optional
Color=			1 oder 2	1 oder 2
Line=			optional	optional
[Polygon]	**no**	**no**	**optional**	**optional**
Type=			yes	yes
DrawOrder=			0 ... 9	0 ... 9[7]
String=			optional	optional
TextSize=			optional	optional
TextColor=			optional	optional
Color=			1 or 2	1 or 2
Line=			optional	optional
[Custom_Header]	**optional**	**no**	**no**	**no**
<Phyton-code>	optional			
[Custom_POI]	**optional**	**no**	**no**	**no**
<Phyton-code>	optional			
[Custom_Polyline]	**optional**	**no**	**no**	**no**
<Phyton-code>	optional			
[Custom_Polygon]	**optional**	**no**	**no**	**no**
<Phyton-code>	optional			

yes	must be existing
optional	con be used
no	is ignored

6.4 MP file

The source-files have the extension *.MP. They are processed preferably with GPSMapEdit. Versions since 1.0.56 of GPSMapEdit are able to handle the 3-byte-types (Type==0x1xxxx). Not all blocks and key-words, that GPSMapEdit will generate, are supported. Not supported blocks and keys are ignored.

The supported blocks and elements of the block are in overview:

[**IMG ID**] header information

- ID= Identification of the map, is adopted as file-name for the IMG file.
- Name= Name the map like it is shown in MapSource.

7 For types> 0x1000 must a value given, other types can also in the block [DrawOrder] classified into the draw sequence.

• Transparent=	Y: The map is generated as transparent (no background, polygon and transparent flag is set) S: The map is generates as semitransparent (no background, transparent flag is not set)
• DrawPriority=	Sequence of the representation for several sets of maps. Default: 24 (topographic maps).
• Routing=	Y: MapTk will add routing information into the IMG file. N: No routing information in IMG file, even if the routing graph was defined.
• Preview=	Y: defines an overview map against the name conventions, N: defines an detailed map against the name conventions.
• Copyright=	The text is adopted into the IMG file. Shown in GPSMapEdit (optional).
• CodePage=	Code-page for labels. Example: CodePage=1252 for texts with Latin signs. Is taken on into the IMG file (optional).
• Levels=	the maximum number of zoom-levels.
• Levelx=	Definition of bit / coordinate in the Levelx. In Levelx = Levels-1 contains only the background Type 0x4b.
• Zoomx=	(optional)

[**RESTRICT**][8]

• Nod=	number of the road with turn restrictions
• TraffPoints=	list of involved routing nodes
• TraffRoads=	list of involved roads for automatic routing
• RestrParam=	list of involved vehicles

[**POI**]

• Type=	Type of POI (16 Bit) and user defined types (3-byte-types) > 0xffff are supported.
• Label=	Label of POI (optional), Geographic points may have elevation data. The POIs 0x64xx to 0x66xx has an additionally name. '~[0x1f]' separates name and value (<name>~[0x1f]<elevation>). The name may be empty. The elevation may be negative and cn have one decimal. Garmin will round the value as required. So the display may be different from data in the MP file.
• EndLevel=	Level of display (or old Levels=)
• HouseNumber=	(may contain all printable characters)
• StreetDesc=	(may contain all printable characters)
• City=Y	Marker for cities
• CityIdx=[9]	Belongs to city with this index

8 Block is completely defined by GPSMapEdit. No to be changed with a text editor !

9 Old format, CityIdx is only read.

● CountryName=	Name of the country. In GPSMapEdit wis format can be selected for writing:: Tools → Options... → Load & Save → Settings for Polish Format.
● RegionName=	Name of the region
● CityName=	Name of the city
● ZipIdx=[10]	Index of zip-code
● Zip=	Zip code
● Phone=	Phone-number (may contain all printable characters)
● Data0=	Geographic position

[**POLYLINE**]

- Type= Type of the polyline (1 Byte) and user defined types (3-byte-types) > 0xffff are supported.
- Label= Label of the polyline (optional).
  For roads only: ~[0X1F] separates the name of a trail and the number of the trail. The number is shown on the map (decided by Garmin). ~[0X1E] is not supported. Only for roads another 2 labels are optional: 'Label2=...' and 'Label3=...'.
  See page 55
- EndLevel= Level of display (or old Levels=)
- DirIndicator= Marks the direction os a polyline. This is not the definition of an oneway street !
- RoadID= unique numbering of all roads for automatic routing (is set by GPSMapEdit). Not to be changed with a test editor !
- RouteParam= a list of 12 values to specify the behavior for automatic routing. Set by GPSMapEdit or in a script (except 'on way') via special variables.
  Index of variable PARAM in script:

0	Toll
1	Ambulance
2	Delivery
3	Car / Motorcycle
4	Bus
5	Taxi
6	Pedestrians
7	Bicycle
8	Truck

- Data0= List of geographical positions
- Nod?= Node data for routing (is defined by GPSMapEdit)

10 Old format, ZipIdx is only read.

[**POLYGON**]

● Type=	Type of the polygon (1 Byte) and user defined types (3-byte-types) > 0xffff are supported.
● Label=	Label of the polygon (optional)
● EndLevel=	Level of display (or Levels=). The background (Type=0x4b) is set automatically to the highest possible level by the compiler.
● Data0=	List of geographical positions

6.5 IMG file

The IMG file is generated from the MP file and contains a tile of a set of maps or an overview map. A set of maps can contain many tiles. The tiles are referenced in the overview-map and the TDB file. A list of MP file to be compiled to IMG files from MP files is part of the PRJ file. The overview-map must contain all detail-maps in the outline as polygon of the type 0x4a. The name of this Polygon is the name of the tile with the filename of the IMG file, without '.img ', cut through '~ [0x1d] '. Example: *Bozen~[0x1d]00001006*. The restriction 256 points for polylines and polygons is not valid for overview-maps.

The polygon type 0x4a is never compiled for a detailed map. That allows the usage as domain polygon (page 51).

A topographic map contains contour lines. These contour lines are types 0x20 to 0x25. If these polylines are found in the IMG file and the TDB file is prepared the profile of routes can be displayed in MapSource or Basecamp. Elevation data for lines (0x20 to 0x25) and POIs (0x62xx to 0x66xx) must be converted to feet. The format is important and will issue warning if specific condition are not fullfilled: Conditions: allowed are the characters '-', '.', '0' to '9', 'M' and a space to separate value and dimension 'M'. For POIs 0x64xx to 0x66xx a name, separated by '~[0x1f]' in front of the elevation is allowed.

Detailed maps do not contain objects 'Type=0x00' and polygons 'Type=0x4a'.

All IMG files of a set of maps are to be stored in one folder. All IMG files, stored in this folder can be taken into a set of maps. Which files must be declared in the PRJ file under 'IMGfiles=' as a mask. Example a mask: '0* .img' for all files the name with '0' beginning. The overview-map must to match the mask.

With MapTk generated maps are optionally transparent. The coding of names and other texts in the map is always 8 bits with MapTk .That allows German 'Umlaute' for example and 'ß' (code-page 1252) or Cyrillic characters, code-page 1251.

Since version 2.7 IMG files are containing additional elements for the index (MDR file), since version 3.00 data for automatic routing is added and since version 4.0 searching for address is possible

6.6 TYP file

The necessary statements are taken from a PRJ file. TYP files reduce the speed of picture display in MapSource. The minimum requirements in PRJ files are:

- In the block [Project] the keys 'ProductID', 'FamilyID' and 'Typ' must exist. If this keys are given an additional REG file is written.

- The blocks [POI], [Polyline] and [Polygon] are used for generating the TYP file. [DrawOrder] is obsolete but also accepted.

Other blocks are ignored while generating TYP files.

Is in the block [Project] a complete project defined, a REG file is generated additionally with all necessary product-parameters for the Windows registry key 'HKEY_LOCAL_MACHINE\SOFTWARE\Garmin\MapSource\Families\'. A complete product is defined by the keys 'FamilyID', 'IMGpath', 'TYP', 'TDB' and' OverviewMap'. If missing one or more of these keys no REG file is produced.

Double click on the REG file registers a complete set of maps in the Windows registry or actualizes and existing entry.

The name for TYP files must be different for all maps on the GPS receiver !

6.7 TDB file

The TDB file contains important information about the individual tiles. Beneath other data, the size of the Subfiles is contained. Compiling of a tile can change the size of the tile so that MapSource gets problems. It is therefore important - with alteration at tiles - to generate the TDB file again, especially because the process itself with hundreds of tiles can be completed within some seconds.

Through the mask in 'IMGfiles=' in the PRJ file the maps of a set are selected. Example: 'IMGfiles=0 * .img' chooses all files from that with '0' begins. Therefore '00000030.img' but not 'topo.img' or '12345678.img' is in the list of detailed maps.

Furthermore, the TDB file contains the ProductID and FamilyID, possibly changed in the PRJ file. ProductID and FamilyID in TDB- and TYP file as well as in the Registry must match. If not MapSource refuses the service. A wrong size of a tile in the TDB file, MapSource reports at downloading that a small set of maps doesn't fit into an amply large storage!.

TDB files are containing additional elements for the index (MDR file) and automatic routing.

The TDB file will be prepared to display profiles of routes in MapSource and BaseCamp if all tiles are containing contour lines.

6.8 Index

Since version 2.7 the index is an index for searching cities and POIs optional for MapSource and GPS receivers. Since version 4.0 additional address searching is possible. This feature is only available via 'Make' to secure consistency of index, TDB file and the tiles. The index requires special sections in the IMG file. So it is required to recompile the tiles. An incompatibility is displays and the indexing is aborted. It is absolute necessary to have consistency of registry and index file.

The index is using cities and POIs in groups 0x0100 – 0x1100, 0x2800, 0x2a00 – 0x3000 and 0x6400 – 0x6600. For polylines all routable roads (in list ROUTING) are possible in the index. 'City', 'Region' and 'Country' are used to localize in MapSource, BaseCamp or the GPS device. The information to build the index must be complete and consistent.Searching works for the whole map set. The starting point of searching and the order of displayed results Is a property of Garmin. The rules are not published. The index delivers only the required data.

6.9 LOG file

The logging must be activated in 'File → preference'. For most function MapTk is writing results to a text file 'MapTk.log' located in the project folder. This file is created if an order is given. Consecutively issued orders are concatenated in one LOG file. The file contains selected lines from the status window starting with 'Input:', 'Script:', 'Info:', 'New:', 'Index', 'Update:', 'Warning:', 'Error:', 'Stopped:', and 'Make:'. Especially after 'Make' of large projects a quick overview of operation is given. Messages are modified in version 3.1 for this function.

If 'Index check' is activated for the project the log is used to inform about incomplete and inconsistent data for the index MDR.

Status window:

```
Update:  d:/karten/Südtirol/M00202_mdr.img
         1 index warnings in log file
```

Log file.

```
…
Index:   No valid city for street: KLUGHAMMER [Bozen Süd]
…
```

The message contains the problem, the object in the map and the name / ID if the tile.

A perfect index has no such warnings.

7 Python-Script

Four python-scripts are used by function 'Script' to manipulate MP files. The Scripts are stored in one PRJ file. To use or modify the scripts you should be somewhat familiar with python (http://www.python.org).

7.1 Overview

The treatment of the MP file takes place over variables:

Variable	Header	POI	Polyline	Polygon
overview	r	r	r	r
type	-	rw	rw	rw
drawpriority	rw	-	-	-
transparent	rw	-	-	-
copyright	rw	-	-	-
levels	rw	-	-	-
codepage	rw	-	-	-
level	rw indexed	rw	rw	rw
zoom	rw indexed	-	-	-
label	-	rw	rw	rw
label1, label2	-	-	rw (roads)	-
name	rw	-	-	-
ID	rw	-	-	-
domain	w indexed	return of domain(ps)	return of domain(ps)-	-
index	-	r	-	-
routing	w	-	r	-
roadclass	-	-	rw	-
speed	-	-	rw	-
restrict[ ]	-	-	rw	-
dirindicator	-	-	r	-
oneway	-	-	rw	-
project[ ]	r	-	-	-
data[ ]	-	rw	rw	rw

r	read only
rw	read and write
-	not available
return	a function call instead a variable since version 4.0

7.2 Data of the Header

The block [CUSTOM_HEADER] is called per MP file once. Global variables should be initialized here. Appraised and changed and can be:

'transparent'	will switch the map to transparent with 'S' oder 'Y'.
'copyright'	allows some text.
'codepage'	is the number of the code-page (i.e. Latin: 1252, Cyrillic: 1251).
'levels'	s the number of levels for the map in range 2 to 8.
'level[]'	is the number of bits per coordinate. Index of this array is the level from 0 to levels-1 and must be descending.
'zoom[]'	is the zoom-step is for a level. Index is 0 to levels-1. The area is 0 ... 8 and must be rising.
'name'	is the name of the map. May remain empty.
'ID'	is the identification of the map, 8 numerical signs for detailed maps. Up to 8 signs, starting with letters for overview-maps.
'domain[]'	is a list of polygon types. The label of those polygons is returned calling the function domain(ps) in section [CUSTOM_POI] and [CUSTOM_POLYLINE] if the coordinate of the POI or polyline is inside such a polygon. Used to define the city of a POI or road for example. The order of searching is defined in the list. Searching is stopped at the first occurrence. If not found an empty string is returned.
'routing'	can be set to 'Y' or 'N' to switch routing on / off for the map.
'data[]'	is a list of coordinates of the current object.

For with Python experienced users:

'ps'	is a dictionary of processed map. The keys comes from the MP file (e.g 'CityName'). The spelling is important.
'mp'	is a String-Array sent to the MP file on output.
'project'	is a dictionary containing the header data from project file (*.prj).
'status.append(text)'	displays text-parameter in the status-window of the program.

Variable, that doesn't occur in the block, remains unchanged.

7.3 Data of Objects

The blocks [CUSTOM_POI], [CUSTOM_POLYLINE] and [CUSTOM_POLYGON] are called once for each corresponding object. Appraised and changed can be:

'overview'	is *True* if an overview-map is processed.
'type'	is the code of the object.
'level'	max level for the object
'label'	is the labeling of the object.

'domain(ps)'	this function returns the label of a polygon (types defined in the header and the POI is positioned in one of those polygons). Used to define the city of a POI or road for example. The relevant coordinate for a polyline is the average of all coordinates of the polyline. The order of searching is defined in the list domain[]. Searching is stopped at the first occurrence. If not found an empty string is returned. Use domain(ps) only if required (e.g. road has no 'CityName') because it is very time consuming

POIs only:

'index'	is 'True' if the type of the POI can be indexed.

Polylines only:

ROUTING	a list of all types for automatic routing (and index)

For maps using automatic routing additional variables are available for streets and trails:

'speed'	is equal to parameter 1 of 'Routeparam' in the MP file.
'roadclass'	is equal to parameter 2 of 'Routeparam' in the MP file.
'restrict'	is equal to parameter 4 … 12 of 'Routeparam' in the MP file.
'dirindicator'	shows only the arrows in GPSMapEdit. The compiler will not evaluate 'dirindicator'.
'oneway'	is equal to parameter 3 of 'Routeparam' in the MP file. 'True' for oneway streets. Should be set to dirindicator' (script: 'oneway=dirindicator').

7.4 Some Python

Examples on page 57.

In principle, all qualities of an object are here to appraise with Python or to change, not only the described variables. That necessitates knowledge of the internal structures however. Objects are stored in an array of dictionaries. The dictionary of an object handed over to the script is ps[]. The Dictionary can be displayed with

```
status.append(ps)
```

in the status-window of the program. The spelling is (key:value, key:value,...). The keys in ps are e.g. 'CityName', 'RegionName', … are also keys in the MP file.

All data in this dictionary is string.

7.4.1 General

Block-building takes place through insertion with of 4 spaces.

Commentaries start with '#'. The rest of the line is ignored.

'return' can be used in these scripts only in the definition of a function.

7.4.2 Variables

Variables must not be declared but defined before using.. A variable is only visible while execution the current part of the script for the current POI, polyline or polygon.

All data for the map is stored in dictionaries. The concerning section of the script is called for all POIs, polylines and polygons. 'ps' is the dictionary of the current object.

The index for indexed variables is in parentheses (for example level[1]). The index for the dictionary ps is a string, for example ps['Label'].

ROUTING is a list if all type of polylines for automatic routing (`if type in ROUTING: ...`)

Index is boolean variable (Tue or False). True if a POI may be indexed.

7.4.3 Functions

Functions are limited on the current [CUSTOM_...] -block.

```
def test(s, val):
    s.append('Test: %s' % (val))
```

called with

```
test(status, '...')      # displays 3 dots
```

Functions defined in [CUSTOM_...] are only valid for this block.

Formating of strings is similar to 'C'. Use the %-operator.

The standard-functions upper() and lower() convert no string-constants with special characters. The string-constant must with converted with decode(' cp1252). Example:

```
s = lower('ABCÄÖÜ'.decode('cp1252'))
```

Result: 'abcäöü'

7.4.4 'status.append()'

Messages can be displayed in the status window of the program with 'status.append('......')'. Example in [CUSTOM_HEADER]:

```
status.append('Copyright:' + copyright) # displays the copyright
```

'+' concatenates strings, here the text and a string-variable. The %-operator can be used.

7.4.5 'domain(ps)'

This function takes the coordinates from the dictionary ps and looks for a polygon at this coordinate. The type of the polygons the be examined are defind in [CUSTOM_HEADER]. If a polygon was found searching is stopped and the label of that polygon is returned.

This function is very useful to add information of cities, regions or countries to POIs and roads for indexing. The problem is to get the required polygon e.g. outline of a plitcal community. Cities are normal parts of the map (e.g. Type= 0x01). Outside this defined area temporary polygons must be added to the tile. The type of the polygon can be 0x4a (never written to the IMG file of a detailed tile) and has e.g. the label '<name of the city>$<name of the region>$<name of the country>'. Assigning to a POI or road uses this code:

```
    try:
        city, region, country = label.split('$', 2) # the components
    except:                      # stop in case of wrong formatted label
        status.error('invalid polygon: %s' % label)
    ps['CityName'] = city        # change the dictionary
    ps['RegionName'] = region
    ps['CountryName'] = country
```

The polygons must be created outside the project (perhaps as an own project) from e.g. ESRI data. GPSMapEdit converts from various formats to MP. If done as a separate project the script function of MapTk is useful.

7.4.6 'bookmark(ps, message)'

Using this function it is possible to mark an POI, polyline or polygon if manual handling is required. The bookmark is written to the end of the MP file. Bookmarks are ignored by the compiler, removed at start of a script or manually in GPSMapEdit..

7.4.7 Conditions, loops

For conditional instructions see examples in the sample-PRJ files. Comparisons are '== ', '! = ', '> ', '>= ',... 'in' tests whether a value is contained in a list, an array or dictionary. The following instruction passes out type and the labeling of all objects for which a text is defined:

```
if 'Label' in ps:
    status.append('%2x: %s' % (type, label))
```

'elif' tests alternative conditions, that 'else'-part is executed if no one of the prior conditions became true.

The following loop displays in [CUSTOM_HEADER] the bits / coordinate of all levels:

```
for i in range(levels-1):
    status.append('Level%d=%d' % (i, level[i]))
```

'range' is equal to a list of integers. 'range(4)' compiles to 0,1,2,3.

The following displays out all entries of a dictionary:

```
for s in ps:
    status.append(s + '=' + ps[s])
```

8 Tipps & Tricks

8.1 Relation TYP and IMG

TYP files are

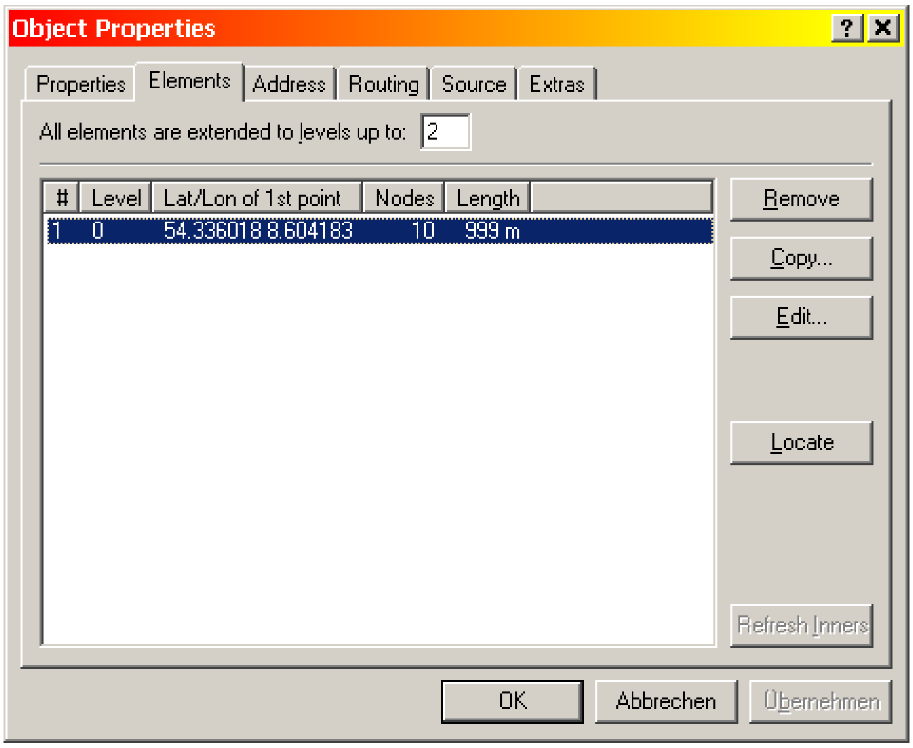
1. an extension for maps like CS, CN Topo D V1
2. mandatory for maps like Topo D V2 to bring the enhanced elements to th display.

To change the appearance of a map the IMG files are not modified. That is impossible for locked maps. The TYP file will of course only change the appearance of covered elements. Elements in the TYP file but not contained in the IMG files are not injurious. Each set of maps owns an (or no) TYP file. The name of the file is unimportant but the FamilyID must match. Each active TYP file has an entry in the Windows registry. That changes the appearance on the PC screen. The required tiles of all sets of maps are selected in MapSource. This tiles and the associated TYP files are packed into a single file (gmapsupp.img). This file is transferred to the instrument (or an memory card). This changes the appearance on the navigator. Using other tools the procedure becomes mostly more complex.

8.2 Using the Levels

Ideally, a map is shown in all pieces of information in one level, like on paper printed maps. GPS equipment and even the PC-screen don't allow a such representation. Refrained from the size of display all details became an illegible mash due to the limited resolution. Therefore, pieces of information must be left out so much to get a clear representation while zooming out. What is left out the author of the map defines by statement of the levels. The polylines are critically because they cover POIs, polygons and other polylines . Only a schematic procedure sticks with big maps. Within narrow limits are the creativity kept with it. An example: A hiker would like to see his trails (0x16) and would still like to represent them in the Level 1. Fading small streets (for example 0x06) instead of the trails the map the becomes unclear and is not longer usable.

To leave small trails and elevation lines as first, then the small streets has proven itself. The control takes place with each object over the entries 'Levels=...' or 'Endlevel=...' in the MP file. Without this statement, the object is visible in level 0 only. In GPSMapEdit stated in.'All elements are extended to levels up to:':



All elements are extended to levels up to: 2

#	Level	Lat/Lon of 1st point	Nodes	Length
1	0	54.336018 8.604183	10	999 m

The street in this example becomes displayed up to the level 2.

In order to improve the overview, only few objects can be adopted to next higher level. Maps, like the Topo Germany, have 3 active levels, containing to shown objects. That is an absolutely meaningful division. In steep terrain with 20m contours, it can be meaningful to apportion the Level 0. The 20m contour, hedges... and other, less important objects are faded from Level 1. Splitting into paved streets, 0x06... 0x09, and unpaved streets (0x0a) improves almost nothing.

The Level 0 always contains all information of the map. Separate objects in the different levels would generate unnecessary expenditure when processing the maps in the editor. One pursue should be represented optimal results one achieve the concept of all piece of information in the level 0 with statement up to which level the object and avoid mistakes. The reduction of the piece of information in the Level 1 and upper takes place automatically by reduction of the used bits / coordinate. Additionally, a simplification of polylines and polygons, that gets the parameter from the bits / coordinate of the relevant level, is performed when compiling. Polylines or polygons collapsed after compression to a single point are ignored.

8.3 Very Large Maps

Maps should be hold small. A size of 15 Mbytes for MP files is enough. Larger maps will increase most processing steps unnecessary. There are some limitations in the size of maps, in each case however a solution for the problem. The distance between two points a polyline or a polygon is a 16-bit-value limited on one. With very big tiles, the background Type=0x4b can overstep this value. An error message is issued:

```
Error:   map to large for 'Level0=xx' !
```

Either the background-polygon must be divided or the bits / coordinate must be reduced.

8.4 Complex lines and polygons

Line and polygons out of more than 255 points are displayed correctly in MapSource. On display of a GPS receiver these objects are missing. Such large objects are often reasoned after importing tracks or elevation lines. During compiling those tiles a warning is displayed using the coordinates of the first point:

```
Warning: polyline type=0x21 '120 M'
         at 47.34039,10.207110 has > 255 nodes !
```

In 'Script' a bookmark is set at the beginning of the polyline. The functions 'Reorganize' and 'Script' cuts large objects. Roads must be split manually.

In GPSMapEdit the function 'tool → Generalize', 'Remove Object Duplicates' and 'Merge Inner 'Polygons' will simplify the map. Maps for automatic routing should not be simplified before setting all routing nodes.

8.5

Building address index needs routing information. All data for the index is taken only from the IMG file of all tiles. If a tile was changed the index for the whole map must rebuild (→ Make).

An object (POI, road) for the index needs a label. If at least one road with a label is assigned to a city the address searching index is automatically build. Other roads with a label but no city will issue a warning 'no valid city … '. If the region or country is missing a warning is issued too. The warnings can be suppressed 'Index check'. But with warnings the index may give strange results. A map with POI index but without address searching should not have roads assigned to cities, regions or countries.

'CountryName' for all indexed objects **must** contain an abbreviation like in

```
CountryName=ITALIEN~[0x1d]ITA
```

to use the refinement city, region and country for searching of streets. The abbreviation is defined in GPSMapEdit in 'Postal Address Items'. Abbreviations for regions are optionally.

The chain Country → Region → City → Street / POI must be complete for a suitable index. Inconsistencies are found while building the index - but different spelling of name not. The POIs and roads should have unique names / spelling. Avoid special characters like / ( ) !.

Address searching is based on internal tables for routable polylines. Roads may have up to 3 label. Suggestion for

1. roads with a number:

label:	~[...]number of road[11]
label2:	name of the road[12] (optional append ~[0X1F]number of trail[13])
label3:	extra name for searching[14], e.g. the number of a trail

2. other roads and trails:

label:	name of the road (optional append ~[0X1F]number of trail)
label2:	extra name for searching, e.g. '123', the number of a trail
label3:	extra name for searching, e.g. 'Dolomiten Route'

11 Shield 0x01 … 0x06
12 Display on the map and index
13 Display on the map if possible, not in the index
14 In the index, not displayed on the map

Adding the number of an marked trail allows to search for this number in the street field of MapSource, BaseCamp or the GPS-device.

8.6 Folders

MapTk stores data at two places preferably:

1. Folder from that the program was called, sources:
   - MP files,
   - BAK files from 'Script' and the
   - PRJ file

   MapTk should be started from this folder. This folder should be backed up regularly.
2. Folder of maps used by MapSource:
   - IMG files,
   - TDB file
   - TYP file
   - MDR and MDX files and
   - REG file (not for but used by MapSource)

   The location of this files can be defined in the PRJ file at a different than the source folder ('Binary files at'). A common folder is clearer however. These files can be generated from the sources again at anytime. The back up of this folders is therefore not absolutely necessary.

This separation in two folders is not necessary, the clearness promotes with many files however. It is allowed to hold all files in one folder, preferred for small sets.

8.7 Working with many files

The functions of the menus 'Functions' and 'Tools' allows the batch processing. With the selection of files, several files can be chosen using the usual buttons / mouse-click-combinations. Problems occur if several hundred files appear. This will display an error message.

For the 'Make'-function all files are in a list. The possible number of the files is not limited here.

8.8 All files in one folder

All files be stored for 'Make' in the same folder. [Project] in the PRJ file looks then like follows:

```
# all in the same folder

[Project]
Product=STirol
FamilyID=202
Version=201
Compile=1
Index=1
Style=1
Overview=basemap.img
IMGfiles=002021*.img
Copyright=Mit Genehmigung von ...
[END]
```

```
[IMG]
# list of all files
[End]
```

'IMGpath=' is dropped, empty or contains './' to use the current working folder.

8.9 Script

Example script for 5 Level. Level 0 and 1 differs only through the contour type 0x20, that is not longer represented in the Level 1.

```
[CUSTOM_HEADER]
# Python script
# overview      1 overview map ols e 0
# project       dictionary of project header (lower IDs)
# name          name of the tile
# ident         ident of tile ( 8 character string )
# familyid      family ID ( 16 bit, read only )
# drawpriority  draw priority of tile ( 1 ... 31 )
# transparent   1 map is transparent olse 0
# copyright     copyright in MapSource
# project       dictionary of project header (lower IDs)
# levels        number of levels in map
# level[]       bit / coord, index is level
# zoom[]        zoom level
# domain[]      list of polygon types
# status.append(msg)    display message in status windowif overview:
# modify in overview map
    levels = 4
    level[0] = 17
    level[1] = 15
    level[2] = 13
    level[3] = 11
else:                                   # modify in detailed map
    levels = 5
    level[0] = 23
    level[1] = 22
    level[2] = 21
    level[3] = 19
    level[4] = 17                       # for polygon 0x4b only
    transparent = 1
    routing= Y                          # map for automatic routing
    domain=(1,2,3)                      # polygons type=1,2 or 3 for POI-city
copyright = 'MapTk'                     # in MapSource
[END]

[CUSTOM_POI]
# Python script: customizing points
# overview      1 overview map else 0
# project       dictionary of project header (lower IDs)
# type          code of object
# levels        number of levels in map
# level         visibility up to level, -1: remove
# label         text for object
# index         1 type has index properties index else 0
# data[]        array of coordinates [y,x]
# ps{}          dictionary of this POI
# domain(ps)    returns the label of found domain
# bookmark(ps, msg)     set a bookmark with a message
# status.append(msg)    display message in status windowif overview:
# modify in overview map
    grp = type >> 8                    # group of types
    if type == 0x2f04:                 # airport (invisible)
        type = 0x2d0b                  # -> visible
    if type < 0x0b00:
        level = 2
    elif type < 0x1200:
        level = 1
    else:
        level = 0
else:                                  # modify in detailed map
    grp = type >> 8                    # group of types
    if type == 0x2f04:                 # airport (invisible)
        type = 0x2d0b                  # -> visible
    if type < 0x0b00:
        level = 3
    elif type < 0x1200:
        level = 2
    elif type == (0x2f0b, 0x6616):     # parking, summit
        level = 2
    elif grp in (0x2c, 0x62, 0x63):    # spots
        level = 2
    else:
        level = 1
    # add first time a city name for a POI inside a city polygon
    city = domain(ps)
    if index and label and city:
        if 'CityName' not in ps:
            ps['CityName'] = city
[END]

[CUSTOM_POLYLINE]
# Python script: customizing polylines [RGN40]
# overview   1 overview map else 0
# project    dictionary of project header (lower IDs)
# type       code of object
# levels     number of levels in map
# level      visibility up to level, -1: removes
# label      text for object
# label2     text for object, roads only
# label3     text for object, roads only
# data[]     array of coordinates [y,x]
# ROUTING    list of road types
# ps{}       dictionary of this POLYLINE
# domain(ps) returns the label of found domain
# bookmark(ps, msg)     set a bookmark with a message
# status.append(msg)    display message in status windowif overview:
# modify in overview map
    if type in (1, 0x1e):
        level = 2
    elif type in (2, 0x1c):
        level = 1
    elif type in (3, 0x18, 0x1a, 0x1b, 0x1f, 0x26):
        level = 0
    else:
        level = -1                     # remove
else:                                  # modify in detailed map
    if type == 0x18 and label <> '':   # with name -> Level 2
        type = 0x1f
    if type in (0x20, 0x21, 0x22):      # assign land contour
        if  label.find('000 ') != -1:
            type = 0x22
        elif label.find('00 ') != -1:
            type = 0x21
        else:
            type = 0x20
    # change routing parameter for Italian highways to toll / limited speed
    if routing:
        city = domain(ps)               # inside polygon of a city
        if label and city:
            if 'CityName' not in ps:    # do not overwrite
                ps['CityName'] = city
        oneway = dirindicator           # is oneway if polyline has direction
        if type == 1:
            speed = 5                   # average of 90 km/h
            restrict[0] = 1             # is toll
    if type in (1, 2, 3, 4, 5, 0x14, 0x1c, 0x1e):
        level = 3
    elif type in (6, 7, 8, 9, 0x0a, 0x0b, 0x0c, 0x13, 0x1a, 0x1b,
        0x1f, 0x27, 0x28):
        level = 2
    elif type in (0x16, 0x18, 0x1d, 0x21, 0x22, 0x24, 0x25, 0x26,
        0x29):
        level = 1
    else:
        level = 0
[END]

[CUSTOM_POLYGON]
# Python script: customizing polygons
# overview      1 overview map else 0
# project       dictionary of project header (lower IDs)
# type          code of object
# levels        number of levels in map
# level         visibility level, -1: remove
# label         text for object
# data[]        array of coordinates [y,x]
# ps{}          dictionary of this POLYGON
# domain(ps)    returns the label of found domain
# bookmark(ps, msg)     set a bookmark with a message
# status.append(msg)    display message in status windowif overview:
# modify in overview map
    if type == 0x4b:
        level = 3
    elif type in (1, 2, 0x28, 0x3c, 0x3f, 0x40, 0x42, 0x43, 0x44,
        0x46, 0x47, 0x48, 0x4a):
        level = 2
    elif type in (3,):
        level = 1
    else:
        level = 0
else:                               # modify in detailed map
    if type == 0x4b:                # background
        level = 4
    elif type in (1, 0x28, 0x3c, 0x3d, 0x3e, 0x3f, 0x40, 0x42,
        0x43, 0x44, 0x46, 0x47, 0x48):
        level = 3
    elif type in (2, 3, 4, 5, 6, 7, 8, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e,
                  0x13, 0x14, 0x15, 0x18, 0x19, 0x50):
        level = 2
    elif type in (0x16, 0x17, 0x1a, 0x41, 0x49, 0x4d, 0x4e, 0x4f,
                  0x51, 0x51, 0x52):
        level = 1
    else:
        level = 0                    # only level 0
[END]
```

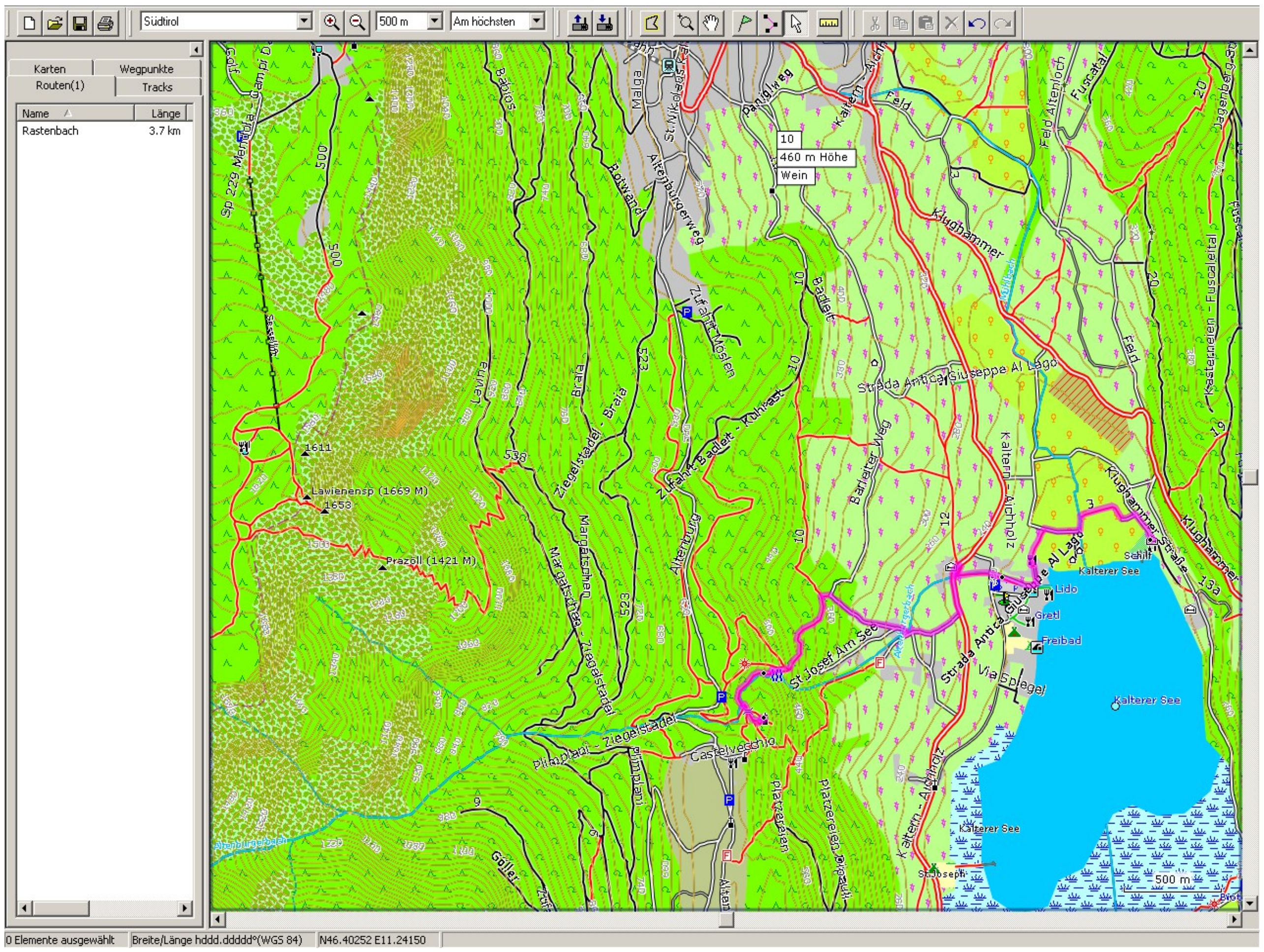
9 Appendix

9.1 Things to know

'Things to know' may help in this summary starting development of maps first time. That are needs or only hints for better overview or laziness.

- In MapTk a set of maps is a project, whether there are 1 or 123 tiles. Source and data for MapSource / BaseCamp may located in different folders, but must not. Using several PRJ files allows several sets of maps in one folder. Preferred: 1 project = 1 folder. MapTk should be called with the PRJ file.
- Tiles of a set of maps have an ID (ID=...). Detailed tiles: a decimal number of exactly 8 digits (e.g. 01234567), basemap: up to 8 characters starting with a letter (preferrable: 'basemap'). The name (Name=...) of a tile or basemap is free and may use the name of the MP file for a better overview.
- The unique family ID (FID) is only defined once in the header of the project. The FID is written (by 'Make') to the TDB, MDX, the TYP file and to the registry. These FID must be unique for PC or GPS device.
- Sets of maps for MapSource or BaseCamp need an basemap. The basemap may be created simply from all MP files of the tiles. The basemap is entered into the list of all 'Maps' at least after creating. The basemap must be compiled (e.g. using 'Make'). The basemap is never transferred to GPS receiver.
- 'Make' will actualize after changing something the concerned files, considering the dependencies. After changing e.g. the FID the TYP, TDB, MDX, Index and REG file are actualized or new created.
- Experience says, that during developing a map the test on PC using MapSource / BaseCamp is good enough. A final test using the GPS device should be performed anyhow.

9.2 Map Example



The map was generated with MapTk, beautified with a TYP file and automatic route calculated.

9.3 'Standard' POI types / symbols

Types for index: 0x01xx … 0x11xx, 0x28xx, 0x2axx ,,, 0x30xx, 0x64xx … 0x66xx. S

0x0000 : 'None',
0x0100 : 'City (Capitol)',
0x0200 : 'City (Capitol)',
0x0300 : 'City (Capitol)',
0x0400 : 'City (Large)',
0x0500 : 'City (Large)',
0x0600 : 'City (Large)',
0x0700 : 'City (Large)',
0x0800 : 'City (Medium)',
0x0900 : 'City (Medium)',
0x0a00 : 'City (Medium)',
0x0b00 : 'City (Medium)',
0x0c00 : 'City (Medium)',
0x0d00 : 'City (Small)',
0x0e00 : 'City (Small)',
0x0f00 : 'City (Small)',
0x1000 : 'City (Small)',
0x1100 : 'City (Small)',
0x1602 : 'Short Tower',
0x1610 : 'Flag, Red',

```
0x1610 : 'Pin, Red',
0x1611 : 'Flag, Green',
0x1611 : 'Pin, Blue',
0x1611 : 'Pin, Green',
0x1615 : 'Flag, Blue',
0x2700 : 'Exit',
0x2a00 : 'Restaurant',
0x2a01 : 'Restaurant',
0x2a02 : 'Restaurant',
0x2a04 : 'Restaurant',
0x2a06 : 'Restaurant',
0x2a07 : 'Fast Food',
0x2a08 : 'Restaurant',
0x2a0a : 'Pizza',
0x2a0b : 'Restaurant',
0x2a0c : 'Restaurant',
0x2a0f : 'Restaurant',
0x2a10 : 'Restaurant',
0x2a11 : 'Restaurant',
0x2a12 : 'Restaurant',
0x2b00 : 'Lodging',
0x2b01 : 'Lodging',
0x2b02 : 'Lodging',
0x2b03 : 'Campground',
0x2b04 : 'Lodge',
0x2b0f : 'Tunnel',
0x2c00 : 'Scenic Area',
0x2c01 : 'Amusement Park',
0x2c01 : 'RV Park',
0x2c02 : 'Museum',
0x2c04 : 'Dot, White',
0x2c05 : 'School',
0x2c06 : 'Park',
0x2c07 : 'Zoo',
0x2c08 : 'Fitness Center',
0x2c09 : 'Building',
0x2c0a : 'Winery',
0x2c0b : 'Church',
0x2d01 : 'Live Theater',
0x2d02 : 'Bar',
0x2d03 : 'Movie Theater',
0x2d05 : 'Golf Course',
0x2d06 : 'Skiing Area',
0x2d07 : 'Bowling',
0x2d08 : 'Ice Skating',
0x2d09 : 'Swimming Area',
0x2d0a : 'Ball Park',
0x2d0a : 'Stadium',
0x2d0b : 'Airport',
0x2e00 : 'Shopping Center',
0x2e01 : 'Department Store',
0x2e04 : 'Restroom',
0x2e05 : 'Pharmacy',
0x2e06 : 'Convenience Store',
0x2f00 : 'Car',
0x2f01 : 'Gas Station',
0x2f02 : 'Car',
0x2f03 : 'Car Repair',
0x2f05 : 'Post Office',
0x2f06 : 'Bank',
0x2f07 : 'Car Rental',
0x2f08 : 'Ground Transportation',
0x2f09 : 'Anchor',
0x2f0a : 'Wrecker',
0x2f0b : 'Parking Area',
0x2f0c : 'Restroom',
0x2f16 : 'Truck Stop',
0x2f18 : 'Ground Transportation',
0x3000 : 'City Hall',
0x3001 : 'Police Station',
0x3002 : 'Medical Facility',
0x3003 : 'City Hall',
0x3006 : 'Toll Booth',
0x3008 : 'Civil',
0x3009 : 'Church',
0x4500 : 'Restaurant',
0x4600 : 'Bar',
0x4900 : 'Park',
0x4a00 : 'Picnic Area',
0x4c00 : 'Information',
0x4f00 : 'Shower',
0x5000 : 'Drinking Water',
0x5100 : 'Bell',
0x5200 : 'Scenic Area',
0x5300 : 'Skiing Area',
0x5400 : 'Swimming Area',
0x5903 : 'Airport',
0x5904 : 'Heliport',
0x5a00 : 'Mile Marker',
0x5b00 : 'Bell',
0x6300 : 'Summit',
0x6400 : 'Triangle, Red',
0x6401 : 'Crossing',
0x6402 : 'Building',
0x6402 : 'Residence',
0x6403 : 'Cemetery',
0x6404 : 'Church',
0x6405 : 'Civil',
0x6406 : 'Crossing',
0x6408 : 'Medical Facility',
0x640b : 'Military',
0x640c : 'Mine',
0x640d : 'Oil Field',
0x6411 : 'Tall Tower',
0x6412 : 'Bike Trail',
0x6413 : 'Tunnel',
0x6416 : 'Scenic Area',
0x6419 : 'Trail Head',
0x6500 : 'Triangle, Blue',
0x6508 : 'Triangle, Blue',
0x6600 : 'Triangle, Green',
0x660a : 'Forest',
0x660b : 'Triangle, Green',
0x6611 : 'Summit',
0x6616 : 'Summit',
0x6619 : 'Scenic Area',
```

9.4 User defined Types

The types are not predefined. The usage is different in different sets of maps. The following list of objects are new in 'Topo Germany V2' or differently used as in the version 1. The new objects are stored in the IMG files at another position as usually. To the differentiation, the bit 24 is set in MapTk, for example 0x1101d for coniferous forest. Peculiarities of these types:

- They **must be declared** in the TYP file in order to be represented at all.
- They can be displayed and processed with GPSMapEdit since version 1.0.56.
- They can be adopted into an IMG file since MapTk version 2.6.
- They are not taken to the index.

No special remarks for processing with a PRJ file in MapTk.

GPSMapEdit displays these types using the skin file also while selecting an object by code, text and icon. Before editing a map the types should be defined in MapTk and a new skin file introduced for GPSMapEdit.

9.4.1 POI

11500	SLUICE
11501	BUILDING
11502	CAVE
11503	ADIT/PIT ENTRANCE
11504	CRANE
11505	PUMP
11506	WINDMILL
11507	ARCHAEOLOGICAL SITE
11508	MONUMENT
11509	CROSS/WAYSIDE SHRINE
1150a	MILESTONE
1150b	TREE
1150c	PLACE
1150d	HELICOPTER BASE
1150e	PIER
1150f	BEACON
11510	LIGHTS
11511	FORD
11512	CHAINAGE
11513	AERIAL MAST
11514	STOP
11515	MINING TRAIL

9.4.2 Polyline

00d00	PEDESTRIAN AREA
00e00	TRAIL
00f00	MAJOR TRAIL
01300	STEEP TRACK
10e00	SUBTERRANEAN STREAM
10e01	SEEP SECTION
10e02	DAM
10e03	WEIR
10e04	SLUICE
10e05	LOCK CHAMBER
10e06	BANK REINFORCEMENT
10e07	QUAY WALL
10e08	GROYNE
10e09	FASCINE
10e0a	MOLE
10e0b	BULKHEAD
10e0c	WATERFALL
10e0d	NARROW-GAUGE RAILWAY
10e0e	SUBURBAN RAILWAY
10e0f	TRAMWAY
10e10	METRO
10e11	MOUNTAIN RAILWAY
10e12	RECREATIONAL RAILWAY
10e13	MAGLEV
10e14	CABLE RAILWAY
10e15	CHAIR LIFT
10e16	SKI LIFT
10e17	MATERIAL CABLE RAILWAY
10e18	TUNNEL

10e19	TUNNEL
10e1a	CULVERT
10e1b	CULVERT
10e1c	PIER
10e1d	BRIDGE
10e1e	FORD
10e1f	CONVEYOR
10f00	BANK
10f01	WALL
10f02	SUPPORTING WALL
10f03	FENCE
10f04	ROCK
10f05	SALINA
10f06	CRANE
10f07	ARCHAEOLOGICAL SITE
10f08	MONUMENT
10f09	RACETRACK
10f0a	SKI JUMP
10f0b	TREE ROW
10f0c	HEDGE
10f0d	HEAP
10f0e	NATIONAL PARK
10f0f	PROTECTED AREA
10f10	AREA
10f11	MARKED TRAIL

9.4.3 Polygon

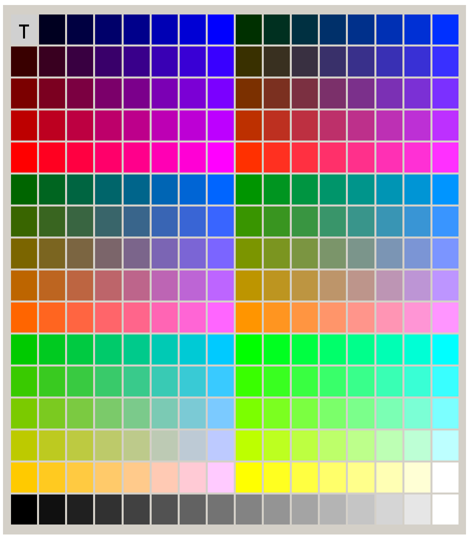
10f00	BANK
10f01	BANK REINFORCEMENT
10f02	BARRAGE
10f03	BRIDGE/OVERPASS
10f04	BUILDING
10f05	BUILDING
10f06	PUBLIC BUILDING
10f07	BULKEAD
10f08	BUSINESS AREA
10f09	CAMPGROUND
10f0a	CHUTE
10f0b	SPECIALISED CROP
10f0c	TREE NURSERY
10f0d	HOPS
10f0e	VINEYARD
10f0f	FRUIT TREES
10f10	DAM
10f11	DEBRIS
10f12	DRY-DOCK
10f13	FAIRGROUND
10f14	FASCINE
10f15	FIELD/RINK
10f16	GRASSLAND
10f17	GRAVEL
10f18	GROVE
10f19	GROVE (SOFTWOOD)
10f1a	GROVE (HARDWOOD)
10f1b	GROYNE
10f1c	HEATHLAND
10f1d	HELICOPTER BASE
10f1e	LOCK
10f1f	LOCK CHAMBER
11000	MIXED AREAS

11001	MODEL AIRPLANE AREA
11002	MOLE
11003	OPEN PIT/STONE PIT
11004	PEDESTRIAN AREA
11005	PIER
11006	PORT
11007	PORT BASIN
11008	QUAY WALL
11009	RECREATIONAL FACILITIES
1100a	RESIDENTIAL AREA
1100b	REST AREA
1100c	ROCK
1100d	SAND
1100e	SANDBANK
1100f	SEDIMENTATION TANK/SLURRY POND
11010	SEWAGE FIELD
11011	SHIP LIFT
11013	SOLAR CELLS
11014	SPECIAL AREA
11015	STADIUM/SPORTS FIELD
11016	STATION AREA
11017	SWIMMING BATH
11018	TIDAL FLAT
11019	TIDAL GULLY
1101a	WATER ENGINE
1101b	WATERFALL
1101c	WET GROUND
1101d	CONIFEROUS FOREST
1101e	DECIDUOUS FOREST
1101f	MIXED FOREST

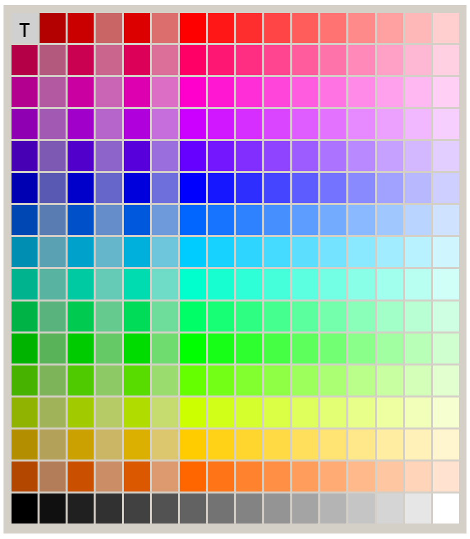
9.5 Color-palette (256 colors)

Combinations (240 colors)			grey	
red	green	blue		
00	00	00	000000	black
39	30	20	101010	
7B	65	41	202020	
BD	95	6A	313131	
FF	CA	8B	414141	
	FF	B4	525252	
		D5	626262	
		FF	737373	
			838383	
			949494	
			A4A4A4	
			B4B4B4	
			C5C5C5	
			D5D5D5	
			E6E6E6	
			FFFFFF	white
			D0D0D0	transparent

These colors are used by Garmin for the customer defined POIs. Using this colors should be work also on older GPS device. Brighter colors may be chosen in the header of the project.



Garmin colors



MapTk colors

Only the colors of the actual palette are selectable using the graphical editor. Other colors in the PRJ file are rounded to the nearest color of the actual palette during loading the object. To avoid accidental change of colors the mixing of different palettes in a project should omitted. The current (perhaps rounded) colors are written to the PRJ file with 'Save' . 'Cancel' will close the window without changes.

9.6 Language codes

0	Unspecified	9	Basque	18	Czech	27	Latvian
1	French	10	Catalan	19	Croatian	28	Romanian
2	German	11	Galican	20	Hungarian	29	Albanian
3	Dutch	12	Welsh	21	Polish	30	Bosnian
4	English	13	Gaelic	22	Turkish	31	Lithuanian
5	Italian	14	Danish	23	Greek	32	Serbian
6	Finnish	15	Norwegian	24	Slovenian	33	Macedonian
7	Swedish	16	Portuguese	25	Russian	34	Bulgarian
8	Spanish	17	Slovak	26	Estonian		

Index

3-byte-types..................................5f., 42ff.
3-point-average......................................11
Analyze..6
appearance..16
automatic routing...............................5, 22
background..18
Backup registry......................................14
BorderWidth....................................20, 39
Build project...8
Catmull-Rom spline filter........................11
cities..47
Clear cache...7
code page..7
CodePage.......................................16, 43
Color...39f.
color-definition.......................................13
color-palette...19
Colorado..18
colors...68
Combine...10
Combine GPX files................................10
Command line..5
comments..37
Comments...17
Compile..38
Compile map...8
Compile TYP..13
conditional instructions..........................52
contour line..45
contour lines...46
control characters..................................17
copy protection...6
Copyright..................................16, 38, 43
CP1251..7
CP1252..7
Data0...17
Delete from registry...............................14
detail-maps..18
dialog-box..8
dictionary...50
dirindicator...50
DirIndicator..44
display-sequence...................................40
domain..................................48, 50, 57ff.
Douglas-Peuker filter.............................11
DrawOrder...40
DrawPriority................................9, 16, 43
Edit project file...8
Edit text file..7
elevation..43
Elevation..16
elif..52
else..52
Exit...8
family ID..26
FamilyID..38
feet...17, 45
folder..6
for...52
functions..16, 51
GMAP...5, 9f.
gmapsupp.img................................10, 53
GPSMap60C..18
GPSMapEdit...............................4, 17, 23
GPSMapEdit skin...................................14
GPX..10
GPX track filter.......................................11
GPX track to MP....................................12
Hausdorf-Distance.................................11
Help..15
homepage..15
Icon library.......................................29, 37
if..52
IMG...8, 53
IMG analysis..8
IMG file...8, 18
IMG file reset transparent........................9
IMG file set transparent............................9
IMG set 'DrawPriority'...............................9
IMGfiles..38
IMGPath...38
Import..27
in..52
index..................5, 10, 15, 19, 26, 45f., 58
Index..38
Index check.....................................27, 47
Index file..26
indexed variables...................................51
info...15
info.xml..9f.
Installation..4
invisible..20
language...7
LblCoding..16
level..49
Levels...16, 53

Levelx..16
Line..39f.
LineWidth...20, 39
locked...6, 8
logging...8, 47
Make..8, 18
MapTk_*.pdf....................................15, 37
MapTk.dat...4, 37
MapTk.exe..37
MapTk.exe..15
MapTk.prj..8, 18
Marine maps..6
Mask GPX tracks by MP........................12
MDR...5
meter...17
MP file...18, 42
MP to GPX..10
Name...37
name conventions....................................6
near nodes...24
New project file..8
night-colors...13
oneway...50
Oregon...18
Origin0...17
orking directory..8
overview...49
overview map..............................9, 18, 38
OverviewMap..38
Parabolic spline filter..............................11
PDF..4
PDF viewer...8
POI...20, 29, 47
points..20
Polygon..20, 33
Polyline..20, 31
pre-processor...5
Preferences..7
Preview...16
PRJ file.....................................17, 19, 37
process...15
Product...38
profile of routes....................................45f.
project-file...37
Python..50
Python script...16
raster image...13
REG file..19
registry..9, 14, 19
Rename...7
Reorganize...8
restrict...50, 59
Restrict...23
RESTRICT...43
Restrictions...22f.
return..50
RGN10...17
RGN20...17
RGN40...17
RGN80...17
RoadID..23, 44
roads..22
RouteParam....................................23, 44
routing...57, 59
Routing..16, 23
routing graph..23
routing information.................................17
script.....................................8, 16, 34, 37
Script..8
set of maps...6
speed...50, 59
spline filter..11
Split GPX..10
standard-function....................................51
status.append...51
Stop..15
String..39f.
Style...38
tag 'cmt'..11
TDB analysis...9
tdb.dict...8f.
TextColor.......................................20, 39f.
TextSize...20, 39f.
tile cache..8
track filter...11
transparent...9
Transparent..16
typ..49
TYP...13, 53
TYP analysis..13
TYP file...19, 45
Type...38ff.
Type=0x00...45
Type=0x4a...45
used types..14
variables...51
Verify Map..23
Version...38
Vista..4
Windows...4

working directory....................................25
Wow6432Node.......................................19
XOR IMG file...9
....67
....17
.gma..9
'IMG get 'DrawPriority'..............................1
'roadclass...50
[Cities]..17
[Countries]...17
[Custom_HEADER]................................41
[Custom_POI]...41
[Custom_Polygon]..................................41
[Custom_Polyline]..................................41
[End]..41
[IMG ID]...16, 42
[IMG]..38
[POI]..38, 43
[Polygon]..40
[POLYGON]..45
[Polyline]...39
[POLYLINE]..44
[Project]...8, 37
[Regions]...17
*_mdr.img..5, 19
~[0x1f]...43, 45