GPSBabel Documentation

GPSBabel Documentation

Table of Contents

List of Tables

List of Examples

Introduction to GPSBabel

The Problem: Too many incompatible GPS file formats

There are simply too many gratuitously different file formats to hold waypoint, track, and route information in various programs used by computers and GPS receivers. GPX [http://www.topografix.com/gpx.asp] defines a standard in XML to contain all the data, but there are too many programs that don't understand it yet and too much data in alternate formats.

Perhaps you have an GPSMap 60CSx and your friend has a StreetPilot 2720 . You've collected a list of your favorite locations as waypoints and you'd like to be able to share them. Unfortunately, his copy of Garmin Mapsource won't read data created by your copy of National Geographic Topo . What you need is a program that converts data between the two programs.

GPSBabel actually solves that problem for you and much more...

The Solution

The original author of GPSBabel, Robert Lipe [/people/robertlipe.html] , needed to convert waypoints between a couple of formats, so he whipped up a converter and designed it upon an extensible foundation so that it was easy to add new formats and made the program freely available. Many others [/people/index.html] have contributed to the program since then.

Most file formats added so far have taken under 200 lines of reasonable ISO C so they can be stamped out pretty trivially. Formats that are ASCII text delimited in some fixed way can be added with no programming at all via our style mechanism.

Chapter 1. Getting or Building GPSBabel

Downloading - the easy way.

GPSBabel is distributed "ready to run" on most common operating systems via the download page [https://www.gpsbabel.org/download.html].

As GPSBabel runs on a wide variety of operating systems, be sure to visit the OS-Specific notes [https://www.gpsbabel.org/osnotes.html] for additional information.

Building from source.

For operating systems where no binary is provided, or if you want the latest development version, you will have to build it from source. The code should be compilable on any system with ISO C++17. It's tested on Ubuntu, macOS, and Windows. Less frequently, someone will build on FreeBSD, OpenBSD, Solaris, etc. Clang/LLVM, GNU C++, and MSVC are regularly exercised via automation.

You can grab a release from the GPSBabel download page [https://www.gpsbabel.org/download.html], but if you're going to be doing any development, you'll find that working from the GPSBabel Github repo [https://github.com/gpsbabel/gpsbabel] is easier. Checkouts via Git, HTTPS, SSH, and Subversion are supported.

There are external requirements for bulding.

Qt [http://qt-project.org]	Qt version 5.12 or newer is required for all builds. MacOS and Windows users can download binaries from Qt Downloads [http://qt-project.org/downloads] Fedora or CentOS users may need to 'dnf install qt5-qtbase-devel'. When in doubt, 'dnf search qt' or 'dnf search qt5' may help you find the correct package name. Ubuntu users may need to 'apt-get install qt5-default'. Package names and versions in Linux frequently change, so you may need to ask your Linux vendor for help or look in tools/Docker* for inspiration for our automated builds that use Docker.
libusb 1.0 [http://libusb.info]	is needed to communicate with use with older USB Garmins. For macOS, we use an included copy. Fedora users may need to 'yum install libusb-devel'. Ubuntu users may need to 'apt install libusb-dev' or look in tools/Docker* for inspiration from our automated Docker builds.

Brief history of internals

Parts of GPSBabel have been public since 2002, with some of the original design and original code came from 2001. It was originally in C89, not C++ and while we strongly encourage modern C++ code where we can use it, we've not gone back to those older formats - some of which we don't have the hardware to test and have fallen out of touch with original authors - and rewritten them in Modern C++ style. There are this void*'s everywhere, C String use, gross buffer abuse, and other things that look more like a C program from the 80's than a C++ program of modern date. Code that's earned it's own wings can continue to fly with us as long as it passes the tests we have. We've additionally not enforced style rules as strongly across modules as we could have. Fixing both of these is a goal for us in 2020.

Building with Qt Creator

Qt provides a lovely IDE (Integrated Development Environment) with an editor and debugger. Its use is strongly encouraged for those new to C++. Once you have Qt correctly installed, just opening the `CMakeLists.txt` from the File->Open menu in an already-running instance of Qt Creator [https://www.qt.io/development-tools] is the fastest and easiest way for most people to get to development because it handles things like build dependencies changing and class and method compilation.

A path of low resistance for some users is to use the CMakeLists.txt file from the command line. If you type cmake . [https://cmake.org/cmake/help/latest/] in our working directory it will create a buildsystem with the default generator (Unix Makefiles or Visual Studio) which you can use for development.

Building from the command line

Those familiar with the command line may be more comfortable using cmake from the command line. It is important to set the CMAKE_BUILD_TYPE when generating a build system with single configuration generators such as Unix Makefiles and Ninja. With multi-configuration generators such as Visual Studio and Xcode the build type is selected at build time instead of when generating the build system.

To create a buildsystem using Ninja:

```
cmake -G Ninja -DCMAKE_BUILD_TYPE=Release .
```

To create a visual studio project that can be built with msbuild:

```
cmake -G "Visual Studio 17 2022" .
```

To create a Xcode project:

```
cmake -G Xcode .
```

There are additional variables that can be defined cmake on the cmake command line to customize your build of GPSBabel.

GPSBABEL_WITH_LIBUSB = no|pkgconfig|system*|included*|custom	note that libusb is NOT used on windows.	
	no	build without libusb-1.0. functionality will be limited.
	pkgconfig	build with libusb-1.0 found by pkg-config.
	system	build with libusb-1.0 found on system library path and under libusb-1.0 on system include path (default, linux, openbsd).
	included	build with libusb-1.0 included with gpsbabel (default, macOS only).
	custom	build with user supplied libusb-1.0. LIBS and INCLUDEPATH may need to be set with GPSBABEL_EXTRA_LINK_LIBRARIES and GPSBABEL_EXTRA_INCLUDE_DIRECTORIES.
GPSBABEL_WITH_SHAPELIB = no|pkgconfig|included*|custom	no	build without shapelib. functionality will be limited.
	pkgconfig	build with shapelib found by pkg-config.

	included	build with shapelib included with gpsbabel (default).
	custom	build with user supplied shapelib. LIBS and INCLUDEPATH may need to be set with GPSBABEL_EXTRA_LINK_LIBRARIES and GPSBABEL_EXTRA_INCLUDE_DIRECTORIES.
GPSBABEL_WITH_ZLIB = no|pkgconfig|included*|custom	no	build without zlib. functionality will be limited.
	pkgconfig	build with zlib found by pkg-config.
	included	build with zlib included with gpsbabel (default).
	custom	build with user supplied zlib. LIBS and INCLUDEPATH may need to be set with GPSBABEL_EXTRA_LINK_LIBRARIES and GPSBABEL_EXTRA_INCLUDE_DIRECTORIES.
GPSBABEL_MAPPREVIEW	This options enables the map preview feature. With the feature disabled QtWebEngine and QtWebEngineWdigets are not used. Note that QtWebKit and QtWebKitWidgets are not longer supported.	
GPSBABEL_EMBED_MAP	Embed gmapbase.html for map preview. When using this option gmapbase.html will be compiled into the executable and does not need to be distributed.	
GPSBABEL_EMBED_TRANSLATIONS	Embed translations. When using this option the gpsbabel provided translations will be compiled into the executable and do not need to be distributed. The Qt provided translations still need to be distributed.	
GPSBABEL_ENABLE_PCH	Enable precompiled headers when building the target gpsbabel.	
GPSBABEL_EXTRA_COMPILE_OPTIONS	Extra compile options when building the target gpsbabel.	
GPSBABEL_EXTRA_INCLUDE_DIRECTORIES	Extra directories to include when building the target gpsbabel.	
GPSBABEL_LINK_LIBRARIES	Extra libraries to link when building the target gpsbabel.	
GPSBABEL_LINK_OPTIONS	Extra link options when building the target gpsbabel.	
GPSBABEL_DOCVERSION	string appended to documentation location for www.gpsbabel.org. The default value is the version string, e.g. "1.7.0". This is used by the gpsbabel.org target, you are unlikely to need it unless you are maintaining www.gpsbabel.org.	
GPSBABEL_WEB	Path where the documentation will be stored for www.gpsbabel.org. This is used by the gpsbabel.org target, you are unlikely to need it unless you are maintaining www.gpsbabel.org. The default location is "gpsbabel.org".	

Additional targets are available for special purposes.

check	Run the basic test suite.
check-vtesto	Run valgrind memcheck.
gpsbabel	Build the command line tool.
gpsbabel.hmtl	Create the html documentation.
gpsbabel.org	Create documentation for use on www.gpsbabel.org.
gpsbabel.pdf	Create the pdf documentation.
package_app	Collect the components for distribution. On Linux the gpsbabel generated components will be under gui/GPSBabelFE, any dynamically linked required libraries are not included. On macOS an app bundle will be created at gui/GPSBabelFE.app and an apple disk image will be created at gui/GPSBabelFE.dmg. On windows an image will be created in the directory gui/package, and an installer will be created gui/Setup-x.y.z-Setup.exe.

Runtime Dependencies:

On non-macOS unix builds by default we now compile in the gpsbabel generated translation files, i.e. gpsbabelfe_*.qm, gpsbabel_*.qm, as well as gmapbase.html. When compiled in these files do not need to be distributed. These are used by the GUI. Additional translation files from Qt will also be used if they are found. They may be in a package such as qt6-translations-l10n.

Chapter 2. Usage

Invocation

If you're using GPSBabel, you will need to know how to do at least two things: read data from a file, and write it to another file. There are four basic options you need to know to do those things:
Command: `-i` *`format`*
Meaning: Set input format
Command: `-f` *`filename`*
Meaning: Read file
Command: `-o` *`format`*
Meaning: Set output format
Command: `-F` *`filename`*
Meaning: Write output file

Important

Case matters. Notably `-f` (lowercase) sets the *input* file. `-F` (uppercase) sets the *output* file.

The *`format`* parameters in the above list refer to the names of formats or file types supported by GPSBabel.

```
gpsbabel -?
```

will always show you the supported file types. In this document, the various supported formats are listed in Chapter 3, *The Formats* . The name that you would use on the command line follows the format name in parentheses.

Options are *always* processed in order from left to right. In practical terms, this means that things you want to read should appear in the command before things you want to write. This sometimes surprises new users as adding options to turn on debugging at the end, for example, doesn't work as the debugging is turned on after all the interesting work is done. The reason for this strict ordering becomes more apparent once you learn about mixing formats and filters.

The *`filename`* parameters specify the name of a file to be read or written.

To use GPSBabel in its simplest form, just tell it what you're reading, where to read it from, what you're writing, and what to write it to. For example:

```
gpsbabel -i geo -f /tmp/geocaching.loc -o gpx -F /tmp/geocaching.gpx
```

tells it to read the file `/tmp/geocaching.loc` in geocaching.com format and create a new file `/tmp/geocaching.gpx` in GPX format. It's important to note that the names have nothing to do with the formats actually used.

This command will read from a Magellan unit attached to the first serial port on a Linux system (device names will vary on other OSes; typically COMx: on Windows) and write them as a geocaching loc file.

Example 2.1. Command showing Linux download from Magellan serial and writing to .loc file

```
gpsbabel -i magellan -f /dev/ttyS0 -o geo -F mag.loc
```

This second command does the same on Microsoft Windows.

Example 2.2. Command showing Windows download from Magellan serial and writing to .loc file

```
gpsbabel -i magellan -f com1 -o geo -F mag.loc
```

Optionally, you may specify `-s` in any command line. This causes the program to ignore any "short" names that may be present in the source data format and synthesize one from the long name. This is particularly useful if you're writing to a target format that isn't the lowest common denominator but the source data was written for the lowest common denominator. This is useful for writing data from geocaching.com to a GPS so my waypoints have "real" names instead of the 'GC1234' ones that are optimized for receivers of the lowest common denominator. A geocacher using Linux with a Magellan receiver may thus find commands like this useful.

```
gpsbabel -s -i geo -f geocaching.loc -o magellan -F /dev/ttyS0
```

His counterpart on Windows will find this equivalent

```
gpsbabel -s -i geo -f geocaching.loc -o magellan -F com1
```

Suboptions

Many of the available format options in GPSBabel can themselves take options. While we try to make all the formats do the most sensible thing possible without any extra options; this allows great power and flexibility in the operation of the program.

Suboptions are comma separated and immediately follow the option itself. The available suboptions are listed on the individual format pages. We'll make an example from the section called “Google Earth (Keyhole) Markup Language (kml)” :

```
gpsbabel -i gpx -f file.gpx -o kml,deficon="file://myicon.png",lines=0
-F one.kml -o kml -F two.kml
```

This command will read the GPX file `file.gpx` and create two KML files. `one.kml` will have the given icon and no lines between track and routepoints. `two.kml` will be created with the defaults used in the KML writer.

Suboptions for the various formats allow you to change serial speeds, pass arguments to filters, change the type of file written, override icon defaults, and lots of other things. The suboptions for each filetype are documented on the page in this document that describes the option itself.

Advanced Usage

Argument are processed in the order they appear on the command line and are translated internally into a pipeline that data flows through when executed. Normally one would:

read from one input
optionally apply filters
write into one output

but GPSBabel is flexible enough to allow more complicated operations such as reading from several files (potentially of different types), applying a filter, reading more data, then writing the merged data to multiple destinations.

The input file type remains unchanged until a new *-i* argument is seen. Files are read in the order they appear. So you could merge three input files into one output file with:

Example 2.3. Merging multiple files into one

```
gpsbabel -i geo -f 1.loc -f 2.loc -f 3.loc -o geo -F big.loc
```

You can merge files of different types:

Example 2.4. Merging multiple files of differing types.

```
gpsbabel -i geo -f 1.loc -i gpx -f 2.gpx -i pcx 3.pcx -o gpsutil -
F big.gps
```

Example 2.5. Writing the same data in multiple output formats.

You can write the same data in different output formats:

```
gpsbabel -i geo -f 1.loc -o gpx -F 1.gpx -o pcx -F 1.wpt
```

Route and Track Modes

Most formats supported by GPSBabel will make a reasonable attempt to work transparently with waypoints, tracks, and routes. Some formats, like garmin require the *-t* flag to work with tracks and *-r* to work with routes. *-w* is for waypoints, and is the default. So if you wanted to read all data from a Magellan Meridian GPS receiver into a gpx file, you might use a command like:

```
gpsbabel -t -r -w -i magellan -f com1: -o gpx -F backup.gpx
```

Tracks and routes are advanced features and don't try to handle every possible hazard that can be encountered during a conversion. If you're merging or converting files of similar limitations, things work very well.

Many of those hazards can be overcome with our filters but there are often compromises to be made. For example, if you have a GPX route that contains 150 turn points but you're sending the route to a GPS receiver that supports only 30 turnpoints, something has to go. One might use our 'simplify' filter to produce a route that retained the 30 most mathematically significant turnpoints but that may not really be the route you had in mind.

Tracks and routes will sometimes be converted to a list of waypoints when necessary, One example is when writing into one of the CSV formats. The inverse operation is not supported right now, so reading the converted track back from CSV will always result in a list of waypoints, not the original track.

The presence of *-s* on the command line tends to create havoc on tracks and routes since many of these formats rely on internal linkages between such points and renaming them may break those linkages. In general, don't use *-s* when tracks or routes are present.

Working with predefined options

GPSBabel can read a file on startup to set defaults for options. All module and filter options may be set this way.

The format of the file is identical to the inifile-format often seen on Windows. Here is an example:

[Common format settings]
snupper=Y
snlen=10
[gpx]
gpxver=1.1
[magellan]
baud=115200
[tiger]
[Garmin categories]
; any # from 1 to 16
1=fixed waypoints
2=temporary waypoints

Each section of the file starts with a '[section]' header followed by any number of lines formatted option=value. Leading and trailing whitespace will be automatically removed from header, option and value items. Lines starting with '#' or ';' will be treated as comments and ignored.

There are three optional sections.

- Common format settings.

  Any option from any of the formats listed here will be used by GPSBabel unless explicitly provided on the command line.

- Common filter settings.

  As above, but for filters.

- Garmin categories

  This allows you to give readable names to the numeric categories used internally in some Garmin devices and the Mapsource formats such as GDB and MPS. This is information is also used by our GPX and garmin_txt formats as well.

By default, GPSBabel tries at startup to load the file named `gpsbabel.ini` from the following locations:

- current working directory
- Windows: all paths "APPDATA", "WINDIR", "SYSTEMROOT" declared in environment.
- Unix like OS'ses: `${HOME}/.gpsbabel/` , `/usr/local/etc/` and `/etc/`

If the `-p` option is specified, the above locations are not searched. Only the filename specified by that option will be used.

There may be situations where predefined values are not usable (i.e. wrapper applications using GPSBabel in the background). The inifile mechanism can be disabled with an empty filename.

```
gpsbabel -p "" -i gpx -f something.gpx -o tiger -F -
```

Realtime tracking

Introduced in GPSBabel 1.3.1, we now have an *experimental* feature for realtime tracking via the new `-T` option. This reads position reports from selected formats and writes an output file when a position report is received.

As of this writing, Garmin's PVT protocol and NMEA are supported inputs. KML, NMEA, and the various XCSV formats are supported on output. Additional formats may be added by interested parties later.

Example 2.6. Read realtime positioning from Garmin USB, write to Keyhole Markup

```
gpsbabel -T -i garmin -f usb: -o kml -F example.kml
```

Will read the USB-connected Garmin and rewrite 'example.kml' atomically, suitable for a self-refreshing network link in Google Earth.

Example 2.7. Read realtime positioning from Wintec WBT-201 via Bluetooth on Mac, write to Keyhole Markup

```
gpsbabel -T -i nmea -f /dev/cu.G-Rays2-SPPslave-1 -o kml -F example.kml
```

Will read the Wintec WBT-201 via Bluetooth, using the name that the Mac assigned it, and rewrite 'example.kml' atomically, suitable for a self-refreshing network link in Google Earth.

Be sure to substitute an device name appropriate for your device and OS, such as `/dev/cu.usbserial` or `/dev/cu.BT-GPS-37A695-BT-GPSCOM-1` for Mac, `COM23:` for Windows, or `usb:` for Garmin USB . These names (except the "usb:" parlance for Garmin USB) are assigned by your operating system.

Batch mode (command files)

In addition to reading arguments from the command line, GPSBabel can read directions from batch (or command) files via the `-b` option.

These files are ideal for holding long command lines, long file lists, complex filters and so on. You can use all GPSBabel options and combinations when writing such files. Nesting batch files by using the `-b` option within a batch file is supported.

Here is an example demonstrating segmenting a large command line by placing the input and filtering directives in a file called 'all_my_files'.

```
gpsbabel -b all_my_files -o gdb -F all_my_tracks.gdb
```

'all_my_files' could look like this:

```
-i gpx
-f saxony_in_summer_2004.gpx -f austria_2005.gpx
-i gdb
-f croatia_2006.gdb
-x nuketypes,waypoints,routes
-x track,pack,split,title="LOG # %Y%m%d"
```

List of Options

The complete list of available options to GPSBabel can be obtained by running **gpsbabel -h** . While there are a number of options, most people will not use most of them, so don't be intimidated.

`-p` Read preferences file. On startup, GPSBabel will look for a file named `gpsbabel.ini` containing preferences you have provided. This option lets you pick a different files. See the section called “Working with predefined options” for more info.

`-s` Write "smart" names. This option influences some - but not all - of our writers to try to build "smart" waypoint names. For example, in modules that know about geocaching, it may replace "GC1234" with the actual name of the geocache.

`-r` Work on routes. This option has a subtly different meaning in different cases. As the very first formats in GPSBabel were for serial GPSes and routes and tracks were large and thus time-consuming to transfer, the default was waypoints only with this option to turn on the extra data. Some of our file formats use this option to mean "work only on routes, even if you have tracks/waypoints", but we're trying to discourage that behavior and in most cases, consider it a bug.

`-t` Work on tracks. See `-r` for usage.

`-w` Work on waypoints. This is the default.

`-T` Enable Realtime tracking. This option isn't supported by the majority of our file formats, but repeatedly reads location from a GPS and writes it to a file as described in the section called “Realtime tracking”

`-b` Process batch file. In addition to reading arguments from the command line, we can read them from files containing lists of commands as described in the section called “Batch mode (command files)”

`-x` *`filter`* Run filter. This option lets use use one of of our many data filters. Position of this in the command line does matter - remember, we process left to right.

`-D` Enable debugging. Not all formats support this. It's typically better supported by the various protocol modules because they just plain need more debugging. This option may be followed by a number. Zero means no debugging. Larger numbers mean more debugging.

`-h`, `-?` Print help.

`-V` Print version number.

Chapter 3. The Formats

? Character Separated Values (xcsv)

This format can...

- read and write waypoints

This format is a very flexible module that can be used to read or write nearly any plain-text record-based waypoint file. This flexibility is achieved by combining this format with "style" files that describe the format of the waypoint files.

There are several formats built in to GPSBabel that use the underlying xcsv machinery. Each of those formats takes the same options as the xcsv format, with the obvious exception of the `style` option. Those formats are all based on style files that can be found in the "style" directory in the GPSBabel source distribution.

`style` option

Full path to XCSV style file.

This option specifies the style file that defines the records to be read on input or written on output. This is not a valid option for the various built-in xcsv-based styles; they have prebuilt style definitions.

For information on the format of xcsv style files, see Appendix C, *GPSBabel XCSV Style Files*.

`snlen` option

Max synthesized shortname length.

This option specifies the maximum allowable length for a short name on output. This option overrides the style file.

Valid values for this option are 0 (off) and 1 (on).

`snwhite` option

Allow whitespace synth. shortnames.

When this option is specified, GPSBabel will allow whitespace (spaces or tabs) in generated short names. This option overrides the style file.

Valid values for this option are 0 (off) and 1 (on).

`snupper` option

UPPERCASE synth. shortnames.

When this option is specified, GPSBabel will make all short names contain only UPPERCASE characters. This option overrides the style file.

Valid values for this option are 0 (off) and 1 (on).

`snunique` option

Make synth. shortnames unique.

When this option is specified, GPSBabel will ensure that all short names are unique within the output file. This option overrides the style file.

Valid values for this option are 0 (off) and 1 (on).

`urlbase` option

Basename prepended to URL on output.

This option specifies the base name to prepend to a URL on output. This might be useful if an input file contains URLs in a relative format and you need them to be in an absolute format.

`prefer_shortnames` option

Use shortname instead of description.

This option causes GPSBabel to use the short name of the waypoint instead of the description. This overrides the style file.

Valid values for this option are 0 (off) and 1 (on).

`datum` option

GPS datum (def. WGS 84).

This option specifies the GPS datum to be used on read or write. Valid values for this option are listed in Appendix A, *Supported Datums*.

`utc` option

Write timestamps with offset x to UTC time.

All database fields on one tab-separated line (tabsep)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

This format, like the custom format, is mainly used for the purpose of testing GPSBabel. It is supposed to contain one field for each piece of information supported by the xcsv format writer, but it may not be entirely in sync with the documentation at Appendix C, *GPSBabel XCSV Style Files*.

For a list of fields, see the style/tabsep.style file in the GPSBabel source distribution.

Columbus/Visiontac V900 files (.csv) (v900)

This format can...

- read waypoints
- read tracks

Read-only support for the csv file format used by Visiontac VGPS-900™ and Columbus V-900™ GPS data loggers. These seem to be two brand names for the exact same product.

The the V-900 stores logs on a microSD card in a custom csv format. This format contains NULL characters and fixed length fields, and therefore can not be handled by the normal csv module in GPSBabel.

Visiontac VGPS-900 [http://www.visiontac.com/v900_specs.htm]

Example 3.1. Conversion of a v900 csv log file to a gpx format

```
gpsbabel -i v900 -f 09040400.csv -o gpx -F outfile.gpx
```

The device support logging of trackpoints, waypoints, and voice recordings (.wav files).

If you create voice recording waypoints, a link (url) to the corresponding wav file is added to the waypoint. If you happen to use this for OpenStreetMap.org project, you can easily click on a waypoint and open the wav file from within JOSM. For this you must use gpx version 1.1 as the output file. The next example shows exactly how to do that.

Example 3.2. Conversion of a v900 csv log file to a gpx 1.1 format

```
gpsbabel -i v900 -f 09040400.csv -o gpx,gpxver=1.1 -F outfile.gpx
```

Comma separated values (csv)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

There are a billion variants of Comma Separated Value data. This is the one specifically that makes DeLorme [http://www.delorme.com] S&A Deluxe 9™ happy. It's also a very simple program and useful for many other programs like spreadsheets, but contains only a very minimal information. For general purpose use, you'll probably be more happy with our universal csv (unicsv) format.

CSV is also the correct format for Lowrance MapCreate™, their commercial mapping program, or GDM6 (their free waypoint manager) for iFinder which is available at lowrance.com [http://www.lowrance.com/Software/GDM6/Default.asp]

On write, this format writes simple "latitude, longitude, name" lines, but on read it will read anything supported by our human readable definition.

For something-separated data that has headers identifying the various fields, see our universal csv format.

Example 3.3. Example 'csv' file

```
35.97203, -87.13470, Mountain Bike Heaven by susy1313
36.09068, -86.67955, The Troll by a182pilot & Family
```

```
35.99627, -86.62012, Dive Bomber by JoGPS & family
36.03848, -86.64862, FOSTER by JoGPS & Family
```

that same data written in unicsv format would appear as:

```
No,Latitude,Longitude,Name,Altitude,Description,Symbol,URL
1,35.972033,-87.134700,"GCEBB",0.0,"Mountain Bike Heaven
 by susy1313","geocache","http://www.geocaching.com/seek/
cache_details.asp?ID=3771"
2,36.090683,-86.679550,"GC1A37",0.0,"The Troll by a182pilot &
 Family","geocache","http://www.geocaching.com/seek/cache_details.asp?
ID=6711"
3,35.996267,-86.620117,"GC1C2B",0.0,"Dive Bomber by JoGPS &
 family","geocache","http://www.geocaching.com/seek/cache_details.asp?
ID=7211"
4,36.038483,-86.648617,"GC25A9",0.0,"FOSTER by JoGPS &
 Family","geocache","http://www.geocaching.com/seek/cache_details.asp?
ID=9641"
```

Custom "Everything" Style (custom)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

This format is not actually used by any real product. It is most useful for debugging purposes when developing a new format module for GPSBabel.

To understand the contents of this file, look at the `style/custom.style` file in the GPSBabel source distribution as well as Appendix C, *GPSBabel XCSV Style Files*.

Data Logger iBlue747 csv (iblue747)

This format can...

- read and write tracks

This format is derived from the xcsv format, so it has all of the same options as that format.

This is the format used by the software that comes with the Transystem i-Blue747 GPS [http://www.semsons.com/i747bldalogp.html].

Notice that the iBlue 747 logs the sum of "height above sealevel" and "height of geoid above WGS84". If precise altitude matters to you, check out the height filter which allows you to compensate for this.

Data Logger iBlue757 csv (iblue757)

This format can...

- read and write tracks

This format is derived from the xcsv format, so it has all of the same options as that format.

This is the format used by the software that comes with the Transystem i-Blue757 Pro GPS [http://www.gpspassion.com/forumsen/topic.asp?TOPIC_ID=81290]. It is very similar to the iBlue747 format, apart from the date format being reversed.

The csv log file can be extracted from the GPS receiver using the BT747 software available from `http://www.bt747.org`

Field definitions:

INDEX

A sequential integer which corresponds for each logged point in the file.

example 3308

RCR

?

example 1: T

example 2: TD

DATE

Date that the point was recorded, in the format YYYY/MM/DD

example: 2011/05/14

TIME

Time that the point was recorded, 24-hr format H:MM:SS. Unsure how fractions of a second are handled.

example: 4:15:11

VALID

?

example 1: DGPS

example 2: SPS

LATITUDE

Degrees above the equator (use negative for south of the equator)

example: -33.803645

N/S

North (N)or South (S) of the equator

example: S

LONGITUDE

Degrees east of the Prime Meridian (use negative for east of the Prime Meridian/Greenwich)

example: 150.880499

E/W

East (E) or West (W) of Greenwich

example: E

HEIGHT

Height above sea level in meters

example: 99.859 m

SPEED

Speed in km/h

example: 0.302 km/h

DISTANCE

Distance covered since last point in meters

example: 0.30 m

Example File

Example 3.X. Example 'iBlue 757' file

```
INDEX,RCR,DATE,TIME,VALID,LATITUDE,N/S,LONGITUDE,E/W,HEIGHT,SPEED,DISTANCE
3308,T,2011/05/14,4:15:11,DGPS,-33.803645,S,150.880499,E,99.859 m,0.207 km/h, 0.28 m
3309,T,2011/05/14,4:15:12,DGPS,-33.803645,S,150.880499,E,100.137 m,0.362 km/h, 0.28 m
3310,T,2011/05/14,4:15:13,DGPS,-33.803644,S,150.8805,E,100.416 m,0.302 km/h, 0.30 m
```

Embedded Exif-GPS data (.jpg) (exif)

This format can...

- read and write waypoints

This format has the following options: filename, frame, name, overwrite, offset .

This format reads and writes GPS information embedded in EXIF [http://www.exif.org], the Exchangeable Image Format, data. EXIF is a standardized method of encoding data in pictures such as JPEG, TIFF, and WAV and is frequently used by mobile phones with cameras and cameras with built-in GPS.

EXIF is frequently used for Geolocating photographs so their images can be correlated with time and location.

filename option

Set waypoint name to source filename.

With this default option waypoint names are generated from source filename.

```
gpsbabel -i exif -f "C:\Pictures\IMG_1199.JPG",filename=Y -o gpx -F
OUT.GPX
```

The resulting waypoint in OUT.GPX has name IMG_1199.

frame option

Time-frame (in seconds).

Frame means the maximum time difference that we accept between the EXIF time information of a picture and the timestamp of a track-, route- and waypoint used for tagging. Without this option the maximum time frame is 10 seconds.

```
gpsbabel -i gpx -f holiday.gpx -o exif,frame=60 -F IMG0784.JPG
```

If the camera time wasn't adjusted, you should use the offset option. You may also need to use the frame option, or the interpolate filter.

name option

Locate waypoint for tagging by this name.

When you specify a name with this option we're looking for a waypoint with this name. And, if found, the GPS information of this point is used for tagging the image file.

```
gpsbabel -i gpx -f holiday.gpx -o exif,name="On the beach" -F IMG0786.JPG
```

overwrite option

!OVERWRITE! the original file. Default=N.

In the default case GPSBabel reads the output file (the file that should be tagged with GPS information) and then creates a new file with an additional .JPG extension. With this option in a final step the original file will be deleted and the new file renamed as the original filename.

offset option

Image Offset Time (+HH:MM or -HH:MM).

Uses the given value instead of the value from the tag OffsetTime, OffsetTimeOriginal or OffsetTimeDigitized. This is useful when the image doesn't contain an OffsetTime* tag and the offset is different from the local time, or when the image contains a tag that is incorrect. The format of this option should match that of the tag OffsetTime*, specifically it must be "+HH:MM" or "-HH:MM".

If the camera was using China Standard Time, e.g. in the winter in Taiwan, then you should supply an offset of "+8:00".

```
gpsbabel -i gpx -f holiday.gpx -o exif,offset=+08:00 -F IMG0784.JPG
```

ESRI shapefile (shape)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: name, url .

This format reads and writes ESRI shapefiles. When reading a shapefile Point, PointZ and PointM shapes are mapped to waypoints. PolyLine(Arc), PolyLineZ(ArcZ) and PolyLineM(ArcM) shapes are mapped to routes. When writing a shapefile waypoints are mapped to Point shapes and routes or tracks are mapped to PolyLine(Arc) shapes depending on the the objective given by the -w, -r or -t option. Because shapefiles can only contain one type of shape these options are mutually exclusive.

The minimum shape file set for read consists of a .shp, .shx and .dbf file. A .cpg file will be checked if it exists. When passing a file name for a set of shape files the name of any of the files from this set can be used. The files must be unzipped.

On read any projection format in a .prj file will be ignored. This may or may not result in a misinterpretation of the data. GPSBabel expects the coordinate system used in the shapefile to be a Geographic Coordinate System with units in decimal degrees measuring degrees of longitude (x-coordinates) and degrees of latitude (y-coordinates), e.g. WGS 84. To transform from the spatial reference system described in the .prj file to the Geographic Coordinate System WGS 84 you can use the GDAL utility ogr2ogr, e.g. "ogr2ogr -t_srs EPSG:4326 output.shp input.shp".

On read an attempt will be made to check the code page used by the .dbf file and report if it is not UTF-8. However, character data within the .dbf file is always processed as if it was encoded with UTF-8. If the .dbf file was using a different code page this may or may not cause a problem.

`name` option

Source for name field in .dbf.

This option specifies where to get name information for each shape. Without this option the name data will be fetched from a field named "NAME" if it exists in the .dbf file.

It the value is a set of digits then the name is fetched from a record with that field index in the .dbf file. Otherwise, if the value does not start with a '+' character, the name is fetched from record with that field name in the .dbf file. When the value starts with a '+' character then multiple records from the .dbf file can be combined to create the name. The plus character should be followed by a set of digit(s) representing a field index, or a field name as above. This may be followed by an additional '+' character, and another set of digits or a field name. As many records as desired can be selected in this fashion. For example shape,name=+0+4 would create a name by combining records with field indices 0 and 4. shape,name=+osm_id+name would create a name by combining records with field names 'osm_id' and 'name'.

`url` option

Source for URL field in .dbf.

This option specifies where to get URL information for each shape. Without this option the URL data will be fetched from a field named "URL" if it exists in the .dbf file.

It the value is a set of digits then the URL is fetched from a record with that field index in the .dbf file. Otherwise, the URL is fetched from record with that field name in the .dbf file.

FAI/IGC Flight Recorder Data Format (igc)

This format can...

- read and write tracks
- read and write routes

This format has the following options: timeadj, ENL, TAS, VAT, OAT, TRT, GSP, FXA, SIU, ACZ, GFO .

FAI/IGC Data File -- Used by the international gliding community to record gliding flights. IGC files can be converted to and from tracks representing recorded flights, and routes representing task declarations in other formats.

IGC Data Format Notes

Refer to Appendix 1 of http://www.fai.org:81/gliding/gnss/tech_spec_gnss.asp for the specification of the IGC data format.

A sample list of software applications that use data in IGC format can be found at http://www.fai.org:81/gliding/gnss/gnss_analysis_software.pdf

GPSBabel can be used to translate data in IGC format to and from various other formats.

Routes in other formats are used to represent IGC task declarations.

Tracks in other formats are used to represent IGC recorded flights.

Converting to IGC format

IGC files generated by GPSBabel will NOT pass security validation tests since the data they contain cannot be proven to originate from an approved flight recorder. For most software applications that use IGC files this is not an issue but for competition scoring, record and badge claims the generated files will not be accepted as proof of a flight.

A track stored in another format (GPX for example) representing a recorded flight can be converted into an IGC file:

```
gpsbabel -i gpx -f mytrk.gpx -o igc -F myflight.igc
```

If multiple track segments are provided in the input file, the one with the most points will be used.

A route stored in another format representing a task declaration can be converted into an IGC file:

```
gpsbabel -i gpx -f myrte.gpx -o igc -F mytask.igc
```

A route and a track in other formats can be included into a single IGC file:

```
gpsbabel -i gpx -f mytrk.gpx -f myrte.gpx -o igc -F myflight.igc
```

A similar result can be obtained by downloading the track log and routes directly from a GPS device connected to a PC. For example to create an IGC file from data recorded in a Garmin GPS connected to the first serial port of a PC running Linux:

```
gpsbabel -t -r -i garmin -f /dev/ttyS0 -o igc -F myflight.igc
```

For Windows operating systems:

```
gpsbabel -t -r -i garmin -f com1 -o igc -F myflight.igc
```

A waypoint file in another format containing a waypoint whose short name is "PILOT" can be merged into an IGC file. The description field of the waypoint will be used for the pilot name in the IGC file header:

```
gpsbabel -i gpx -f mytrk.gpx -f myrte.gpx -f mywpt.gpx -o igc -F
myflight.igc gpsbabel -w -t -r -i garmin -f /dev/ttyS0 -o igc -F
myflight.igc
```

Some formats such as GPX allow routes, tracks and waypoints to exist in the same file and can be used to fully populate an IGC file:

```
gpsbabel -i gpx -f myall.gpx -o igc -F myflight.igc
```

Converting from IGC format

Data in an IGC file can be converted into other formats. For example to generate OziExplorer files containing tracks representing the recorded flight (myozi.plt) and routes representing declared tasks (myozi.rte):

```
gpsbabel -i igc -f myflight.igc -o ozi -F myozi
```

Or to GPX format:

```
gpsbabel -i igc -f myflight.igc -o gpx -F myflight.gpx
```

Header information from the IGC file will be written to the description field of the track(s).

If both pressure altitude and GNSS altitude are recorded in the IGC file, two tracks will be written to the new track file, representing the two altitude tracks. The latitude, longitude and timestamps in the tracks will be identical.

Merging into IGC format

A route stored in another format can be merged with an existing IGC file that has no task declaration, to generate a new IGC file with a task declaration:

```
gpsbabel -i igc -f myflight.igc -i gpx -f myrte.gpx -o igc -F mynew.igc
```

A two dimensional (lat/lon) track recorded during a flight by a GPS receiver can be merged with a one dimensional (altitude) track recorded during the same flight by a barograph instrument. The result is a three dimensional IGC file representing the flight:

```
gpsbabel -i gpx -f baro.gpx -i igc -f my2D.igc -o igc -F my3D.igc
```

The same can be achieved by downloading directly from a barograph instrument supported by GPSBabel. For example with a Brauniger IQ Comp GPS variometer:

```
gpsbabel -i baroiq -f /dev/ttyS0 -i igc -f my2D.igc -o igc,timeadj=auto
-F my3D.igc
```

or:

```
gpsbabel -i baroiq -f com1 -i igc -f my2D.igc -o igc,timeadj=auto -
F my3D.igc
```

(Documentation contributed by Chris Jones, Aug 2004)

`timeadj` option

(integer sec or 'auto') Barograph to GPS time diff.

Sometimes there is a discrepancy between the internal clock in the barograph instrument and GPS time which can result in the altitude and ground positions not correlating correctly. This can be corrected manually by passing the time difference in seconds between the two time domains through the "timeadj" parameter. This can be any positive or negative integer:

```
gpsbabel -i gpx -f baro.gpx -i igc -f my2D.igc -o igc,timeadj=27 -F
my3D.igc
```

GPSBabel can also attempt to deduce the time difference automatically. This is done by comparing the time that it thinks that you landed on the GPS track and the barograph and adjusting accordingly:

```
gpsbabel -i gpx -f baro.gpx -i igc -f my2D.igc -o igc,timeadj=auto -
F my3D.igc
```

`ENL` option

Engine Noise (ENL; default=1).

`TAS` option

True Airspeed (TAS; default=1).

`VAT` option

Total Energy Vario (VAT; default=1).

`OAT` option

Outside Air Temperature (OAT; default=1).

`TRT` option

True Track (TRT; default=0).

GSP option

Ground Speed (GSP; default=1).

FXA option

Fix Accuracy (FXA; default=1).

SIU option

Of Sats (SIU; default=0).

ACZ option

Z Acceleration (ACZ; default=1).

GFO option

G Force? (GFO; default=0).

Flexible and Interoperable Data Transfer (FIT) Activity file (garmin_fit)

This format can...

- write waypoints
- read and write tracks

This format has the following options: allpoints, recoverymode .

GPSBabel supports reading and writing of tracks in the .fit format used by products based on the Garmin ANT+ protocol [https://www.thisisant.com/].

As in case of Garmin Training Center, FIT files contain courses with laps etc. which don't exactly match GPSBabel's waypoints, tracks, and routes. An attempt is made to extract and transform data than can be handled by GPSBabel like heart rate etc. and conversion from waypoints to course points and vice versa. Note that routes are not handled, so they should be transformed to tracks first before converting to FIT. Also, track segments are not supported, so all segments in a track get concatenated and written as a single continuous track without gaps.

When writing a FIT file, waypoints are converted to course points by inserting them at the nearest location in the track/course. By default, generic course points are written unless the waypoint name contains one of the following words in which case course points of type left/right are emitted:

left, links, gauche, izquierda, sinistra
right, rechts, droit, derecha, destro

FIT courses typically contain speed information. If the original track contains neither speed information nor timestamps which may be used to derive the speed, a speed of 10 km/h is assumed and assigned to the course.

`allpoints` option

Read all points even if latitude or longitude is missing.

This option specifies that all points in the input .fit-file should be read. The default behavior is otherwise to skip points without gps coordinates. This is especially useful for devices that do not contain a gps, e.g., Garmin Vivosmart HR.

`recoverymode` option

Attempt to recovery data from corrupt file.

In the default mode the reader will issue a fatal error if it encounters indications of a corrupt file. These indications include:

- a bad Header or File CRC
- a bad endian field
- an attempt to use a message type that hasn't been previously defined
- an attempt to read when the data section doesn't have sufficient data
- an attempt to read past the end of file

In recovery mode if we encounter a CRC error we will ignore it. If we encounter one of the other errors we will abort read processing and continue. This allows any writer to use data that was recovered previous to the read abort.

Garmin 301 Custom position and heartrate (garmin301)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

This is a very simple format that is most useful for exporting data from units that support heart rate data such as Garmin Forerunner 301™, Garmin Forerunner 305™, and Garmin Edge 305™, and to other programs for analysis. It's a simple comma delimited format that includes the timestamp, 3D position information and heart rate so you can pull it into a spreadsheet or graphing program.

Garmin G1000 datalog input filter file (garmin_g1000)

This format can...

- read and write tracks

This format is derived from the xcsv format, so it has all of the same options as that format.

Input format for Garmin G1000 integrated avionics system datalog generated by G1000 system software version 563.20 or later. This datalog is automatically generated and stored to a standard SD card if one

is inserted in the *upper* SD card slot on the G1000's MFD (right-hand display on 2-screen installations, center display on 3-screen installations).

Supports conversion of GPS track (including timestamp) and barometric altitude data to any of GPSBabel's output formats. Does not include support for G1000 stored flight plan (.fpl) route files or user waypoints. Tested on datalogs from Cessna 182T and Turbo 182T, but it should accommodate G1000 datalogs from other airframes as well. If any conversion failures or errors occur, check datalog csv file for incomplete or corrupted records/rows, delete those records/rows from the datalog file and reattempt.

Garmin MapSource - gdb (gdb)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: cat, bitscategory, ver, via, dropwpt, roadbook .

Support for the "Garmin GPS Database" format used by default in MapSource™ versions since release 6.0 of that product. By default GPSBabel creates gdb files of version 2. Version 2 is used in Mapsource 6.3 and 6.5. This format is also used by Garmin BaseCamp™ for Mac and Windows.

Garmin GPS database is an undocumented file format. The basic info for this module came from the existing MapSource conversion code.

`cat` option

Default category on output (1..16).

This option specifies the default category for gdb output. It should be a number from 1 to 16.

`bitscategory` option

Bitmap of categories.

This option is closely related to the 'category' option. While category allows you to choose a single category that waypoints should appear in, this options allows you to specify a bitmask to be used for the category. Options may be specified in either decimal or hex.

Example 3.4. Example for gdb bitcategory option to put all waypoints in categories 1 and 16.

The following two commands are equivalent. They place a the point in both the first and last of the sixteen available categories.

```
gpsbabel -i gpx -f PocketQuery.gpx -o gdb,bitscategory=32769 -F foo.gdb
```

```
gpsbabel -i gpx -f PocketQuery.gpx -o gdb,bitscategory=0x8001 -F foo.gdb
```

`ver` option

Version of gdb file to generate (1..3).

This option specifies the data format version for the output file. Version 2 is the default. Currently, the only other valid values for this option are 1 and 3.

`via` option

Drop route points that do not have an equivalent waypoint (hidden points).

This option instructs GPSBabel to drop hidden (calculated) points from routes. These points are not converted to waypoints or route points.

`dropwpt` option

Don't create waypoints for non-user points.

This option instructs GPSBabel to drop hidden (calculated) points from routes when creating waypoints. These points are not converted to waypoints, but they are converted to route points.

`roadbook` option

Include major turn points (with description) from calculated route.

If this option is specified, GPSBabel drops all calculated route points, with exception of points with a description (i.e. "Make U-turns until you know where you are."). The priority of this option is higher than of the `via` and `dropwpt` options. A value of 1 or y overwrites the `via` and `dropwpt` settings.

Example 3.5. Using gdb option `roadbook` to create simple html roadbook

```
gpsbabel -i gdb,roadbook -f sample.gdb -x nuketypes,waypoints,tracks -
x transform,wpt=rte -o html -F roadbook.html
```

Because gdb internally creates a route AND a waypoint list, you have to drop all waypoints and transform the route into waypoints in order to get a well ordered html output. We suggest these steps for all waypoint-only formats as html.

Garmin MapSource - txt (tab delimited) (garmin_txt)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: date, datum, dist, grid, prec, temp, time, utc .

This is a textual format that contains nearly all of the information contained in the MapSource™ main format, GDB. This format also contains some computed values such as distances between routepoints and trackpoints, speed, and course (heading).

The main goal of garmin_txt is to make aviation data more available. Because MapSource™ supports only the export, GPSBabel gives you the possibility to bring aviation data into MapSource™.

During the export with MapSource™, some fields are written using local settings of MapSource™ and Windows. These include grid format, gps datum, distance and temperature units, and the representation of date and time fields. GPSBabel tries to read all items automatically. Problems with date and time format can be solved with the 'date' and 'time' options.

Example 3.6. Command showing garmin_txt output with all options

```
gpsbabel -i garmin_txt,date="MM/DD/YYYY",time="hh:mm:ss xx" -f in.txt
-o   garmin_txt,date="DD.MM.YYYY",datum="WGS   72",dist=m,prec=6,tem-
p=c,time="HH:mm:ss",utc=+2 -F out.txt
```

`date` option

Read/Write date format (i.e. yyyy/mm/dd).

This option specifies the input and output format for the date. The format is written similarly to those in Windows. An example format is "YYYY/MM/DD".

`datum` option

GPS datum (def. WGS 84).

This option specifies the datum to be used on output. Valid values for this option are listed in Appendix A, *Supported Datums*.

`dist` option

Distance unit [m=metric, s=statute].

This option specifies the unit to be used when outputting distance values. Valid values are M for metric (m/km/kph) or S for statute (ft/mi/mph).

`grid` option

Write position using this grid..

This value specifies the grid to be used on write.

Table 3.1. Grid values for garmin_txt

# idx	short	file-header	sample
0	ddd	Lat/Lon hddd.ddddd	S26.25333 E27.92333
1	dmm	Lat/Lon hddd°mm.mm	N33 56.539 W118 24.471
2	dms	Lat/Lon hddd°mm'ss.s	S25 25 26.8 E28 06 07.3
3	bng	British National Grid	TQ 18919 69392
4	utm	Universal Transverse Mercator	33 U 318293 5637154
5	swiss	Swiss grid	776519 167359

Idx or short are valid params for this option.

`prec` option

Precision of coordinates.

This option specifies the precision to be used when writing coordinate values. Precision is the number of digits after the decimal point. The default precision is 3.

`temp` option

Temperature unit [c=Celsius, f=Fahrenheit].

This option specifies the unit to be used when writing temperature values. Valid values are C for Celsius or F for Fahrenheit.

`time` option

Read/Write time format (i.e. HH:mm:ss xx).

This option specifies the input and output format for the time. The format is written similarly to those in Windows. An example format is "hh:mm:ss xx".

`utc` option

Write timestamps with offset x to UTC time.

This option specifies the local time zone to use when writing times. It is specified as an offset from Universal Coordinated Time (UTC) in hours. Valid values are from -23 to +23.

Garmin POI database (garmin_poi)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

The Garmin POI loader [http://www.garmin.com/support/agree.jsp?id=927] loads custom points of interest into certain models of Garmin GPS receivers. (As of this writing, only the models introduced in 2005 and later are supported. See Garmin's site for more info.) The garmin_poi format produces csv files that can be converted into POI files by Garmin's POI loader.

This format was mostly useful when POI Loader couldn't read GPX and we couldn't write GPI. See GPSBabel's GPI doc.

Garmin Points of Interest (.gpi) (garmin_gpi)

This format can...

- read and write waypoints

This format has the following options: alerts, bitmap, category, hide, descr, notes, position, proximity, sleep, speed, unique, units, writecodec, languagecode .

The format garmin_gpi supports the binary POI (.gpi) files that are usable on newer Garmin GPS receivers. See garmin_poi for additional information about Garmin's own Poiloader program. Garmin POI-Loader [http://www.garmin.com/support/agree.jsp?id=927] is the standard application that creates GPI files with all possible features.

Some of the third party and commercial GPI files are using some kind of encryption or compression that makes the file contents completely unreadable to us. If you get an error "Unsupported code page NNN. File is likely encrypted." means we could basically recognize it as a Garmin GPI file, but it's mangled beyond what we're likely to successfully read.

If a waypoint name is annotated with a trailing '@'NNN where NNN is a number, that number will be used as the speed for POI alerts, just as with Garmin's POI Loader program. The units default to metric kilometers per hour, but this can be changed to statute via the units argument. A speed associated with a specific POI will get precedence over any 'speed' argument provided. For example, a waypoint named "Point@30" will associate a speed of 30 km/h with that specific point even if 'speed=40m' is present in the output arguments.

The layout of GPI files isn't documented and our module was created via reverse engineering. If you get a problem on reading or writing a GPI file, please provide that file (mailto:gpsbabel-misc@lists.sourceforge.net).

At this time we don't support special features as "Tour-Guide" or links to sounds and pictures.

Important

Creation timestamp issue: See the option sleep !!!

This module does not support direct transfer of .GPI files to receivers in Garmin protocol mode. For units like Nuvi, Zumo, or Streetpilot, just choose a file that's on the drive where your GPS is mounted. For units like the X series (GPSMap 60CSx, GPSMap 60Cx, Legend Hcx, etc.) you must explicitly put the unit in mass storage mode or mount the memory chip in an external reader and transfer the file directly.

Example 3.7. Command showing garmin_gpi output example

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,category="Nice Restau-
rants",bitmap=restaurant.bmp,notes -F "My Points.gpi"
```

`alerts` option

Enable alerts on speed or proximity distance.

Because speed isn't a real member of a normal waypoint, you can put the speed values into the waypoint names. "Point@30" will result in a speed value of 30. By default we assume these values are in kilometers per hour.

Proximity distance is also supported by GPX, Garmin GDB, OZI Explorer, and Universal CSV.

```
gpsbabel -i gpx -f "warnings.gpx" -o garmin_gpi,alerts=1 -F "warnings.g-
pi"
```

`bitmap` option

Use specified bitmap on output.

The bitmap (BMP) should be 24x24 (or smaller) and can be in RGB-colors (24- and 32-bit) or 8-bit indexed color format.

If you're starting from images in another format, you may need to use another tool like Gimp or ImageMagick's convert to get the image into one of the above formats to avoid errors about "Unsupported color depth".

Not all devices can support all color depths. GPSBabel (and its developers) have no way of knowing what is supported on any given model so some experimentation may be necessary on your part. It was reported that a Nuvi 3790, for example, will read the POIs only if they use 8BPP.

A color value of 0xFF00FF (blue=255, green=0, red=255), also called "Magenta", can be used for transparent areas.

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,bitmap="tux.bmp" -F
"My Points.gpi"
```

`category` option

Default category on output.

With this option you can specify the category which is primary visible on the device (default is "My points").

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,category="Best Restau-
rants" -F "My Points.gpi"
```

`hide` option

Don't show gpi bitmap on device.

For a large list of points it may be desirable if no bitmaps are displayed on the device. With this option no bitmaps are stored and displayed.

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,hide -F "My Points.gpi"
```

`descr` option

Write description to address field.

The GPI address field is often visible in lists on the device. Use this option if you want to see the waypoint description (which can be an address too) in this lists.

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,descr -F "My Points.gpi"
```

`notes` option

Write notes to address field.

The GPI address field is often visible in lists on the device. Use this option if you want to see the waypoint notes (which can be an address too) in this lists.

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,notes -F "My Points.gpi"
```

`position` option

Write position to address field.

The GPI address field is often visible in lists on the device. Use this option if you want to see the waypoint position (coordinates) in this lists.

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,position -F "My
Points.gpi"
```

`proximity` option

Default proximity.

When no proximity data is available in the source input, GPSBabel uses this as the default proximity value. The parameter has to be in meters, or, when units=s specified, in miles. alerts are automatically enabled.

Example 3.8. Read GPX file, create GPI to alert when you're 1/2 mile from a speed camera.

```
gpsbabel -i gpx -f "SpeedCameras.gpx" -o garmin_gpi,units=s,proximi-
ty=0.5 -F "SpeedCameras.gpi"
```

Its also possible to append a specific distance unit to the parameter.

```
gpsbabel -i gpx -f "SpeedCameras.gpx" -o garmin_gpi,proximity=500m -
F "SpeedCameras.gpi"
```

Table 3.2. Supported distance units (garmin_gpi)

Unit	Description
fa	Fathoms
feet	Feet
ft	Feet
km	Kilometers
m	Meters
mi	Miles
nm	Nautical miles

`sleep` option

After output job done sleep n second(s).

The Garmin units seem to use the creation timestamp of GPI files for internal purposes. In other words, if you load GPI files with same creation timestamp on your device, strange things will happen, such as

having missing or repeated POIs. With the sleep option, GPSBabel waits a given number of seconds after the GPI file was written.

In the normal case of using GPSBabel from the command line or from the GUI, the chance of creating files with the same timestamp is in the nearly ZERO. In scripts or batch files where you are writing multiple files - even from different GPSBabel instances - the odds of this happening is rather good. The sleep option forces GPSBabel to wait after creating a file to ensure the timestamps are unique. Values are specified in seconds and can be 1 or more.

```
gpsbabel -i gpx -f "SpeedCameras.gpx" -o garmin_gpi,sleep=1 -F "Speed-
Cameras.gpi"
```

speed option

Default speed.

When no speed data is available in the source input, GPSBabel uses this as the default speed value. The parameter has to be in kilometers per hour, or, when units=s specified, in miles per hour. alerts are automatically enabled.

```
gpsbabel -i gpx -f "SpeedCameras.gpx" -o garmin_gpi,units=s,speed=30 -
F "SpeedCameras.gpi"
```

Its also possible to append a specific speed unit to the parameter.

```
gpsbabel -i gpx -f "SpeedCameras.gpx" -o garmin_gpi,speed=30mph -F
"SpeedCameras.gpi"
```

Table 3.3. Supported speed units (garmin_gpi)

Unit	Description
km/h	Kilometers per hour
kmh	Kilometers per hour
kph	Kilometers per hour
kt	Knots
knot	Knots
m/s	Meters per second
mps	Meters per second
mi/h	Miles per hour

unique option

Create unique waypoint names (default = yes).

Don't create unique names sample:

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,unique=0 -F "My
Points.gpi"
```

`units` option

Units used for names with @speed ('s'tatute or 'm'etric).

Sample command tells GPSBabel to handle speed values in miles per hour:

```
gpsbabel -i gpx -f "My Points.gpx" -o garmin_gpi,units=s -F "My Points.g-
pi"
```

`writecodec` option

codec to use for writing strings.

This lets you override the default codec of 'windows-1252' when writing strings in Garmin GPI files. This option may be removed in future versions of GPSBabel as it's not known which Garmin devices support which character sets.

`languagecode` option

language code to use for reading dual language files.

Garmin points of interest files may contain data in two languages. If you attempt to read a dual language Garmin points of interest file without specifying which language to use GPSBabel will print an error message containing the language codes used in the file. Subsequently you may use one of these codes with the languagecode option to specify which language to use.

Example 3.9. Example with unspecified language and a garmin points of interest dual language file.

```
gpsbabel -i garmin_gpi -f reference/spb_metro_norm.gpi
```

could produce the following output:

garmin_gpi: Must select language code, RU and EN found.

Example 3.10. Example for specifying language with a garmin points of interest dual language file.

```
gpsbabel -i garmin_gpi,languagecode=EN -f reference/spb_metro_norm.gpi
```

could produce the following output:

59.944070N 30.306680E About Saint-Petersburg Metro - 2016.05/This file represents information about metro stations in aint-Petersburg. Published by NAVICOM, 2016. http://navicom.ru
59.830660N 30.500100E RYBACKOE/Subway
59.934430N 30.329950E NEVSKII PROSPEKT/Subway
...

Garmin serial/USB protocol (garmin)

This format can...

- read and write waypoints
- read and write tracks

- read and write routes

This format has the following options: snlen, snwhite, deficon, get_posn, power_off, erase_t, resettime, category, bitscategory, baud, codec .

GPSBabel supports a wide variety of Garmin hardware via serial on most operating systems and USB on Windows, Linux, and OS X.

For serial models, be sure the GPS is set for "Garmin mode" in setup and that nothing else (PDA hotsync programs, gpsd, getty, pppd, etc.) is using the serial port.

Supported Garmin GPS receivers with USB include

Astro	Forerunner 205	GPSMAP 60CSx	StreetPilot 2650
Edge 205	Forerunner 301	GPSMAP 60Cx	StreetPilot 2720
Edge 305	Forerunner 305	GPSMAP 76C	StreetPilot 2730
eTrex Legend C	Foretrex 201	GPSMAP 76CS	StreetPilot 2820
eTrex Legend Cx	Foretrex 301	GPSMAP 76CSX	StreetPilot 7200
eTrex Legend H	GPS 18[1]	GPSMAP 76Cx	StreetPilot 7500
eTrex Legend HCx	GPSMAP 195	GPSMAP 96	StreetPilot c310
eTrex Summit Cx	GPSMAP 276C	GPSMAP 96C	StreetPilot c320
eTrex Summit HC	GPSMAP 295	Quest	StreetPilot c330
eTrex Venture C	GPSMAP 296C	Quest II	StreetPilot c340
eTrex Venture Cx	GPSMAP 378	Rhino 520	StreetPilot i2
eTrex Venture HC	GPSMAP 396	Rhino 530	StreetPilot i3
eTrex Vista C	GPSMAP 478	Rhino 520 HCx	StreetPilot i5
eTrex Vista Cx	GPSMAP 496	Rhino 530 HCx	
eTrex Vista H	GPSMAP 60C	StreetPilot 2610	
eTrex Vista HCx	GPSMAP 60CS	StreetPilot 2620	

the following Bluetooth Garmin products:

GPS 10[1]

and most serial Garmin GPS receivers including:

eMap	eTrex H	GPS 12	Rhino 110
eTrex Camo	Forerunner 201	GPS 12XL	Rhino 120
eTrex Legend	Foretrex 201	GPS III	Rhino 130
eTrex Summit	Geko 201	GPS III+	StreetPilot III
eTrex Venture	Geko 301	GPS II	StreetPilot III+
eTrex Vista	GPS 12CX	GPS II+	
eTrex (Basic Yellow)	GPS 12Map	GPS V	

The following Garmin GPS receivers are supported, but they do not support Garmin communication protocol and don't work with the `garmin` option. To use these receivers, read or write GPX files from the mass storage device as mounted on your computer.

eTrex 10[2]	Nuvi 255[2]	Nuvi 770[2]	Nuvi 1690T[2]
eTrex 20[2]	Nuvi 250W[2]	Nuvi 775T[2]	Nuvi 3750[2]
eTrex 30[2]	Nuvi 255W[2]	Nuvi 780[2]	Nuvi 3760T[2]
Colorado 300[2]	Nuvi 260[2]	Nuvi 785T[2]	Nuvi 3790T[2]
Colorado 400c[2]	Nuvi 265T[2]	Nuvi 880[2]	Oregon 200[2]
Colorado 400i[2]	Nuvi 265WT[2]	Nuvi 885T[2]	Oregon 300[2]

[1]This model does not support transfer of waypoints, tracks, or routes, but may be used with the realtime tracking feature.

Colorado 400t[2]	Nuvi 260W[2]	Nuvi 1200[2]	Oregon 400c[2]
Dakota 10[2]	Nuvi 270[2]	Nuvi 1250[2]	Oregon 400i[2]
Dakota 20[2]	Nuvi 275T[2]	Nuvi 1260T[2]	Oregon 400t[2]
GPSMap 62[2]	Nuvi 300[2]	Nuvi 1300[2]	Oregon 450[2]
GPSMap 62sc[2]	Nuvi 310[2]	Nuvi 1350[2]	Oregon 450t[2]
GPSMap 62stc[2]	Nuvi 350[2]	Nuvi 1370T[2]	Oregon 550[2]
GPSMap 78[2]	Nuvi 370[2]	Nuvi 1390T[2]	Oregon 550t[2]
GPSMap 78s[2]	Nuvi 465T[2]	Nuvi 1350[2]	StreetPilot c510[2]
GPSMap 78sc[2]	Nuvi 500[2]	Nuvi 1490T[2]	StreetPilot c530[2]
Montana 600[2]	Nuvi 550[2]	Nuvi 2250[2]	StreetPilot c550[2]
Montana 650[2]	Nuvi 600[2]	Nuvi 2250LT[2]	StreetPilot c580[2]
Montana 650t[2]	Nuvi 650[2]	Nuvi 2350[2]	Road Tech Zumo[2]
Nuvi 30[2]	Nuvi 650FM[2]	Nuvi 2350LT[2]	Zumo 220[2]
Nuvi 40[2]	Nuvi 660[2]	Nuvi 2360LT[2]	Zumo 450[2]
Nuvi 50[2]	Nuvi 670[2]	Nuvi 2405[2]	Zumo 500[2]
Nuvi 200[2]	Nuvi 680[2]	Nuvi 2450[2]	Zumo 550[2]
Nuvi 205[2]	Nuvi 750[2]	Nuvi 2450LM[2]	Zumo 660[2]
Nuvi 200W[2]	Nuvi 755T[2]	Nuvi 2450LT[2]	Zumo 665[2]
Nuvi 205W[2]	Nuvi 760[2]	Nuvi 2450LMT[2]	Surely any Garmin product that Garmin actually sensibly designed after 2006 or so.[2]
Nuvi 250[2]	Nuvi 765T[2]	Nuvi 2505[2]	

None of the GPSBabel developers has access to every model on that list, but we've received reports of success and/or have reasonable expectations that the above models work. If you succeed with a model that is not on that list, please send a message to the gpsbabel-misc mailing list with the details so that we may add it.

Not every feature on every model is supported. For example, while we do extract data such as heart rate and temperature from tracks on the sporting models like Edge and Forerunner, GPSBabel is not a fitness program at its core and does not support features like workouts or calorie/fitness zone data. Furthermore, sporting models don't support track upload. When trying to upload tracks to these devices, GPSBabel converts them to courses on the fly and uploads these instead. When uploading waypoints at the same time, these are converted to course points by mapping them to the nearest track point on the track/course (no matter how far away from the track they are). Since course point creation requires time stamps for the track points, they are created automatically assuming a speed of 10 km/h for tracks that lack them.

To communicate with a Garmin GPS serially, use the name of that serial port such as `COM1` or `/dev/cu.serial`.

To communicate via USB use `usb:` as the filename on all OSes. Thus, to read the waypoints from a Garmin USB receiver and write them to a GPX file:

```
gpsbabel -i garmin -f usb: -o gpx -F blah.gpx
```

If you have multiple units attached via USB, you may provide a unit number, with zero being the implied default. So if you have three USB models on your system, they can be addressed as `usb:0`, `usb:1`, and `usb:2`. To get a list of recognized devices, specify a negative number such as:

```
gpsbabel -i garmin -f usb:-1
```

[2]This unit uses GPX format, not Garmin protocol. Therefore one should communicate with it by reading and writing GPX files instead of using this format. Members of this class of products do not support realtime positioning protocol.

When reporting problems with the Garmin format, be sure to include the full unit model, firmware version, and be prepared to offer debugging dumps by adding `-D9` to the command line, like:

```
gpsbabel -D9 -i garmin -f usb: -o gpx -F blah.gpx
```

Custom icons are supported on units that support that. Neither GPSBabel nor your firmware know what is associated with any given slot number. They don't know that the picture you placed in the first slot is a happy face, they only know they're in the lowest numbered slot. GPSBabel names the them consistently with Mapsource, so they are named 'Custom 0' through 'Custom 511'.

For models where the connection on the GPS is a serial interface, be sure the GPS is set for "Garmin mode" in setup and that nothing else (PDA hotsync programs, gpsd, getty, pppd, etc.) is using the serial port.

For models connected via USB, we recommend use of the `usb:` filename. For this to work on Windows, you must install the Garmin driver. For Linux, this will fail if you have the garmin_gps kernel module loaded. See the Operating System Notes [/osnotes.html] for details.

This module also supports realtime tracking which allows realtime position reports from a Garmin GPS receiver over USB or serial.

Important

The following Garmin units do not follow the standard Garmin communications protocol and are *not supported* by GPSBabel.

Marine plotters:

GPSMap 420	GPSMap 450	GPSMap 530	GPSMap 545
GPSMap 430	GPSMap 520	GPSMap 535	GPSMap 550
GPSMap 440	GPSMap 525	GPSMap 540	GPSMap 555

The PDA products

iQue 3000
iQue 3200
iQue 3600
iQue M3
iQue M4
iQue M5

`snlen` option

Length of generated shortnames.

This option overrides the internal logic to figure out how many characters an addressed Garmin GPS will support when using the '-s' smartname option. This should be necessary only if you have a receiver type that GPSBabel doesn't know about or if you want to "dumb down" one unit to match another, such as wanting waypoint names in a StreetPilot 2720 (which supports 20 character names) to exactly match those in a 60CS (which supports 10).

`snwhite` option

Allow whitespace synth. shortnames.

This options controls whether spaces are allowed in generated smart names when using the '-s' option.

`deficon` option

Default icon name.

This option specifies the icon or waypoint type to write for each waypoint on output.

If this option is specified, its value will be used for all waypoints, not just those that do not already have descriptions. That is, this option overrides any icon description that might be in the input file.

Value specified may be a number from the Garmin Protocol Spec or a name as described in the Appendix B, *Garmin Icons*.

This option has no effect on input.

`get_posn` option

Return current position as a waypoint.

This options gets the current longitude and latitude from the attached GPS device and returns it as a single waypoint for further processing. For example, to return the current position from a USB Garmin to a KML file:

```
gpsbabel -i garmin,get_posn -f usb: -o kml -F myposition.kml
```

`power_off` option

Command unit to power itself down.

This command forces an immediate powerdown of the addressed Garmin receiver. It is ignored on hardware that does not support this command. Obviously, further processing once you have sent a "power off" command to a unit that supports it is rather futile, so place this option carefully in your command.

```
gpsbabel -o garmin,power_off -F /dev/ttyS0
```

`erase_t` option

Erase existing courses when writing new ones.

By default, GPSBabel makes effort in order to keep courses already present on the device, if any. This option allow to replace courses already present. If you don't mind to keep old courses, this option is recommended because it allows a faster transfer.

This option applies only to Garmin devices that support courses such as the Edge 305 or the Forerunner 305.

`resettime` option

Sync GPS time to computer time.

This option is experimental and was added to solve a very specific problem. Certain Garmin units (the original black and white Vista is known to have this) will sometimes scramble their clock crazy far into the future (like 2066). When this happens, the GPS itself may or may not work and later conversations with GPSBabel may fail as the time overflows the documented range. The use of `resettime` brings the GPS's internal clock back close enough to reality that the GPS itself can then "fix" it when it has next a lock.

`category` option

Category number to use for written waypoints.

This numeric option will force waypoints to be written with that category number when sending to a Garmin receiver that has category support. It is ignored on receivers without that capability.

`bitscategory` option

Bitmap of categories.

This option is closely related to the 'category' option. While category allows you to choose a single category that waypoints should appear in, this options allows you to specify a bitmask to be used for the category. Options may be specified in either decimal or hex.

Example 3.11. Example for garmin bitcategory option to put all waypoints in categories 1 and 16.

The following two commands are equivalent. They place a the point in both the first and last of the sixteen available categories.

```
gpsbabel -i gpx -f PocketQuery.gpx -o garmin,bitcategory=32769 -F usb:
```

```
gpsbabel -i gpx -f PocketQuery.gpx -o garmin,bitcategory=0x8001 -F usb:
```

baud option

Speed in bits per second of serial port (baud=9600).

Sets baud rate on some Garmin serial unit to the specified baud rate. Garmin protocol uses 9600 bps by default, but there is a rarely documented feature in Garmin binary protocol for switching baud rate. Highest option is 115200.

Download track log and waypoints 12 times faster than default:

```
gpsbabel -t -w -i garmin,baud=115200 -f /dev/ttyUSB0 -o gpx -F garmin-
serial.gpx
```

At the end of the transfer the baud rate is switched to back to the default of 9600. If the connection breaks and the unit is stuck at a high baud rate power cycling should restore the original baud rate.

This option does not affect USB transfer.

Because this feature uses undocumented Garmin protocols, it may or may not work on your device. The author reported success with eTrex Vista, GPSMAP 76s, and GPS V, but it seems likely to be problematic on older units and may be more problematic for writing to the device than reading data from the device.

codec option

override codec to use for device.

This lets you override the default codec used for your device when reading or writing strings from or to your Garmin device. The default codec is device dependent, you can see what codec is being used with your device by adding the -vs option to the command line.

```
gpsbabel -vs -w -i garmin -f usb: -o gpx -F garmin.gpx
```

```
gpsbabel -w -i garmin,codec=windows-1251 -f usb: -o gpx -F garmin.gpx
```

```
gpsbabel -w -i gpx -f garmin.gpx -o garmin,codec=windows-1251 -F usb:
```

Garmin Training Center (.tcx/.crs/.hst/.xml) (gtrnctr)

This format can...

- read waypoints
- read and write tracks

This format has the following options: course, sport .

GPSBabel supports reading and writing of tracks in the .tcx format used by Garmin Training Center™ (GTC). GTC is the successor to Garmin's Logbook™ program for their workout units. It is a free upgrade.

GPSBabel can read GTC v1 and v2 files, and can write v2 files. v2 files are most likely to have a .tcx extension. v1 files typically have a .hst or .crs extension, depending on whether they are in the "history" or "course" format.

There is a fundamental mismatch between this format and most of what we support. GPSBabel deals in waypoints, tracks, and routes. While we do record things like heart rate and temperature when we know it, the fundamentals of Training Center are different. It deals in concepts like laps and calories, which are rather alien to GPSBabel and most of the formats we support. As such, while we can describe the tracks pretty accurately, things like calories and heart zone tracking are not supported. Some of the auxiliary data, such as heart rate (not zone), cadence, and bicycling power are supported.

One of the most useful things you can do with this format is to send .tcx files found on the web or elsewhere to any supported GPS unit. You will probably want to include the transform (rte=trk) and simplify filters in this process. For example,

```
gpsbabel -i gtrnctr -f somefile.tcx -x simplify,count=50 -x transfor-
m,rte=trk -r -o garmin -F usb:
```

where you select the count not to exceed the number of available waypoints for routing on your device.

`course` option

Write course rather than history, default yes.

This flag defaults to true; it must be turned off (course=0) if you want history instead of courses.

`sport` option

Sport: Biking (deflt), Running, MultiSport, Other.

Specify which sport is associated with this activity. Valid values are Biking, Running, MultiSport, and Other.

Geocaching.com .loc (geo)

This format can...

- read and write waypoints

This format has the following options: deficon .

`deficon` option

Default icon name.

This option specifies the icon or waypoint type to write for each waypoint on output.

If this option is specified, its value will be used for all waypoints, not just those that do not already have descriptions. That is, this option overrides any icon description that might be in the input file.

There is no list of valid values for this option.

This option has no effect on input.

GeoJson (geojson)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: compact .

This module supports a subset of the GeoJSON [http://geojson.org/] format.

GeoJSON is a poor fit for GPSBabel's internal data structures as GPSBabel was designed more around common GPS features (waypoints, tracks, routes) than about GIS-style concepts like MultiPolygons or Geometry Collections. In reality, for all but the most simple uses (such as converting a format that GPSBabel supports well to something like Leaflet, you should not expect high fidelity transfers through this format.

Waypoints are mapped to a FeatureCollection of Points. The properties for name and description are written, where available. Tracks are converted to a LineString. MultiPoint are converted to Waypoints. LineString, Polygon and MultiPolygon are converted to routes. MultiLineString are converted to tracks.

The potentially nested/recursive nature of GeoJSON in general would be an awkward implementation.

Initial development was free-handed by looking at the GeoJSON RFC [https://tools.ietf.org/html/rfc7946]. Corner cases were handled by using GDAL's ogr2ogr [http://www.gdal.org/ogr2ogr.html] to convert GPX to JSON and compare the output. The results were then JSON validated [http://geojsonlint.com/] and viewed on JSON web viewer [http://geojson.io/].

`compact` option

Compact Output. Default is off..

This option, when set, reduces the amount of whitespace in the generated GeoJSON. This reduces the size, especially when uncompressed, but reduces the readability to humans.

GlobalSat DG-100/BT-335 Download (dg-100)

This format can...

- read waypoints
- read tracks

This format has the following options: erase, erase_only .

Serial download protocol for the GlobalSat DG-100™, GlobalSat BT-335™, and GlboalSat BT-338X™G-PS data loggers.

While the DG-100 has a button to record waypoints, they seem to be indistinguishable from trackpoints. Therefore, all points will be presented as trackpoints, regardless of whether they were recorded automatically or manually.

GlobalSat DG-100 [http://www.globalsat.com.tw/eng/product_detail_00000090.htm]

Example 3.12. Command showing DG-100 download and erase on Linux

```
gpsbabel -t -i dg-100,erase -o gpx /dev/ttyUSB0 outputfile.gpx
```

Example 3.13. Command showing DG-100 erase_only option on Linux

```
gpsbabel -t -i dg-100,erase_only /dev/ttyUSB0
```

The DG-100 provides a physical USB interface to the host computer, but internally it uses a Prolific PL-2303 chip to do this. So you must have drivers installed on your computer to recognize the PL-2303 and provide that data as a serial port to software like GPSBabel. Such software comes with the unit for Windows. Prolific provides software for Mac OS/X, but unfortunately their driver has a defect which makes it unstable with GPSBabel.

`erase` option

Erase device data after download.

This option erases the track log from the device after download.

`erase_only` option

Only erase device data, do not download anything.

Much like the `erase` this option erases the data in the GPS. It does not transfer data before doing so, making it much faster. This may be handy in a work flow where you want to transfer the data from the GPS, check it on a map, and then remove it from the unit.

GlobalSat DG-200 Download (dg-200)

This format can...

- read waypoints
- read tracks

This format has the following options: erase, erase_only .

Serial download protocol for the GlobalSat DG-200™GPS data loggers.

GlobalSat DG-200 [http://www.usglobalsat.com/p-677-dg-200-gps-data-logger.aspx]

Example 3.14. Command showing DG-200 download and erase on Linux

```
gpsbabel -t -i dg-200,erase -o gpx /dev/ttyUSB0 outputfile.gpx
```

Example 3.15. Command showing DG-200 erase_only option on Linux

```
gpsbabel -t -i dg-200,erase_only /dev/ttyUSB0
```

The DG-200 provides a physical USB interface to the host computer, but internally it uses a Prolific PL-2303 chip to do this. So you must have drivers installed on your computer to recognize the PL-2303 and provide that data as a serial port to software like GPSBabel. Such software comes with the unit for Windows. Prolific provides software for Mac OS/X, but unfortunately their driver has a defect which makes it unstable with GPSBabel.

`erase` option

Erase device data after download.

This option erases the track log from the device after download.

`erase_only` option

Only erase device data, do not download anything.

Much like the `erase` this option erases the data in the GPS. It does not transfer data before doing so, making it much faster. This may be handy in a work flow where you want to transfer the data from the GPS, check it on a map, and then remove it from the unit.

GlobalSat GH625XT GPS training watch (globalsat)

This format can...

- read tracks

This format has the following options: showlist, dump-file, input-is-dump-file, timezone .

Serial download protocol for the GlobalSat Sport gh625XT™ training watch.

The GlobalSat Sport GPS training device present themselves as USBserial devices. To get the training just connect the device using the supplied USB cable to your computer and the device will show up as a serial device.

The gh625XT USB cable provides a physical USB interface to the host computer, but internally it uses a Prolific PL-2303 chip to do this. So you must have drivers installed on your computer to recognize the PL-2303 and provide that data as a serial port to software like GPSBabel. Such software comes with the unit for Windows or can be downloaded.

`showlist` option

list tracks.

The showlist argument displays the list of tracks stored on the device.

Example 3.16. Command showing list of tracks on device

```
gpsbabel -i globalsat,showlist=1 -f /dev/ttyUSB0
```

`dump-file` option

Dump raw data to this file.

The dump-file option is primarily for debugging is module. It lets you provide a file which contains the raw stream of bytes coming from the device. This is useful for capturing device state to describe to a developer that can't actually access the physical device as well as mocking the entire device for automated regression testing.

```
gpsbabel
-i glboalsat,dump-file=gh625xt.bin -f /dev/ttyUSB0
```

can be used to read the device and store its state in the file `gh625xt.bin`. That file can then be distributed and someone else can read it with a command line:

```
 gpsbabel
-i globalsat,input-is-dump-file=1 -f gh625xt.bin -o gpx -F test.gpx
```

`input-is-dump-file` option

Dump raw data to this file.

This is the companion to dump-file and is used to tell the reader that the code is talking to a stored file and not physical hardware.

`timezone` option

Time zone ID.

Google Earth (Keyhole) Markup Language (kml)

This format can...

- read and write waypoints

- read and write tracks
- read and write routes

This format has the following options: deficon, lines, points, line_width, line_color, floating, extrude, track, trackdata, trackdirection, units, labels, max_position_points, rotate_colors, prec .

KML, the Keyhole Markup Language format, was used by Keyhole and is used by Google Earth [http://earth.google.com].

There are concepts in KML that GPSBabel can't support very well on read because they don't map well into other programs. For example, KML has ideas of camera views and names and descriptions can have arbitrarily complicated HTML in them. KML files may have tiered "Styles" which can identify sizing info and URLs of associated icons. Reading such files with GPSBabel - even if your goal is to write it back out as KML - can often have surprising results. Simple files with waypoints and paths (which GPSBabel represents internally as tracks) work fine.

Google Earth also uses GPSBabel internally for receiver communications and several file format imports and exports.

In general, GPSBabel's KML writer is relatively strong. GPSBabel handles simple KML on read fairly well, but if you're dealing with handcrafted KML that uses extensive features that have no analog in other formats like nested folders, ringgeometry, camera angles, and such, don't expect GPSBabel to do well with them on read.

Google Earth 4.0 and later have a feature that can surprise users of this format. Earth's "time slider" feature controls what timestamped data gets displayed. If you're using data that has timestamps (e.g. GPX points that contain time or almost any track data) this will be important to you. The time slider defaults to the far left position and fully closed. This means that only the first data point will be displayed. You can tweak Earth's settings to "view->show time->never" or you can widen the time slider to show the range of data of interest.

See Google Earth's documentation on timelines [http://earth.google.com/userguide/v4/ug_gps.html#timeline] for more info.

`deficon` option

Default icon name.

This option specifies the default name for waypoint icons

`lines` option

Export linestrings for tracks and routes.

When this option is nonzero, GPSBabel draws lines between points in tracks and routes. The default value for this option is 1, which causes lines to be drawn by default. To disable line-drawing, specify `lines=0`.

`points` option

Export placemarks for tracks and routes.

When this option is nonzero, GPSBabel draws placemarks for tracks and routes. The default value for this option is 1, which causes placemarks to be drawn. To disable drawing of placemarks, specify `points=0`.

`line_width` option

Width of lines, in pixels.

This option specifies the width of the drawn lines in pixels. The default value is six pixels.

`line_color` option

Line color, specified in hex AABBGGRR.

This option specifies the line color as a hexadecimal number in AABBGGRR format, where A is alpha, B is blue, G is green, and R is red.

`floating` option

Altitudes are absolute and not clamped to ground.

When this option is nonzero, altitudes are allowed to float above or below the ground surface. By default, this option is zero so that altitudes are clamped to the ground. Specify `floating=1` to allow them to float.

This option is more useful to pilots than to hikers.

`extrude` option

Draw extrusion line from trackpoint to ground.

This option is a boolean flag to specify whether Google Earth should draw lines from trackpoints to the ground. It defaults to '0', which means no extrusion lines are drawn. The option of '1' is, of course, most useful for points that aren't actually on the ground such as those be captured from planes.

`track` option

Write KML track (default = 0).

This is a boolean flag, defaulting to '0', that controls whether GPSBabel writes the <Track> tag that Google introduced in Earth 5.2 for tracks. You may need to turn this off if you have a KML reader that's confused by new tags or if size is critical.

Routes and tracks without sufficient time data are always drawn as Linestrings and never Tracks.

`trackdata` option

Include extended data for trackpoints (default = 1).

This is a boolean flag that controls whether GPSBabel writes extensive data for each trackpoint generated. By default computed speed, timestamps, and so on are written with the default of '1' for this option. If you are writing large tracks and do not value this information, you can reduce the size of the generated file substantially by turning this flag off by setting it to '0'.

`trackdirection` option

Indicate direction of travel in track icons (default = 0).

If set, this options creates directional icons for trackpoints. Arrows will show the direction of travel on drawn tracks and routes.

`units` option

Units used when writing comments ('s'tatute, 'm'etric,' 'n'autical, 'a'viation).

Units is a simple option. Specify 's' for "statute" (miles, feet, and other things that don't sensibly convert to each other, but are craved by Americans), 'm' for "metric", 'n' for "nautical" or 'a' for "aviation". Default is 's'.

`labels` option

Display labels on track and routepoints (default = 1).

When this option is zero, no labels are added for track and route points. This option defaults to one, so labels are added by default.

`max_position_points` option

Retain at most this number of position points (0 = unlimited).

This option allows you to specify the number of points kept in the 'snail trail' generated in the realtime tracking mode.

`rotate_colors` option

Rotate colors for tracks and routes (default automatic).

With this option GPSBabel uses different colors for each track or route. If this option is used without a value then the colors are automatically selected such that the spectrum will be rotated through once for all the tracks and once for all the routes. If this option is used with a positive value then the value is interpreted as the number of degrees in the color circle between adjacent tracks or routes. This option takes precedence over line_color.

`prec` option

Precision of coordinates, number of decimals.

This option specifies the number of digits to be used when writing geographic coordinates, .i.e., latitude and longitude. Precision is the number of digits after the decimal point. The default precision is 6. We limit the number of places we write to improve the fidelity when round-tripping geographic coordinates, reduce file size, and reduce silliness in files caused by repeating decimals in insignificant digits.

As a guideline, at the equator, five decimal places is about 1.1 m, placing it below the accuracy of commodity consumer GPS gear. Six places is 0.11 m, achievable via surveyor grade and differential corrected GPS. Seven is 11 millimeters.

This value is ignored on read and has no impact on the internal representation of data.

Google Takeout Location History (googletakeout)

This format can...

- read waypoints

- read tracks

Google gives you two options to download your location history: In Google Maps, you can go into your Timeline and download your location history a day at a time as KML files. Or, in Google Takeout, you can download all of your location history all at once as a ZIP of JSON files, one for each month.

The "googletakeout" GPSBabel format will process the location history JSON files. It can operate on individual JSON files, the folders in the ZIP that contain a year's worth of data, or the entire location history at once.

There is no official documentation from Google for this format, however, a volunteer is maintaining a schema at locationhistoryformat.com [https://locationhistoryformat.com/].

To obtain your Google Takeout History:

- Go to takeout.google.com [https://takeout.google.com/]
- Click on "Deselect all"
- Scroll down to "Location History" and check the box
- Scroll to the bottom of the page and click on "Next step"
- Click on "Create export"

In a little while, you will receive an email to download your location history, which will be one or more zipfiles. Unzip everything into the same folder. Your history will be in the folder "Takeout/Location History/Semantic Location History" within this folder.

Example 3.17. Convert entire location history to a single GPX file

```
gpsbabel -i googletakeout -f "Semantic Location History" -o gpx -F
all.gpx
```

Example 3.18. Convert a single years' location history to GPX

```
gpsbabel -i googletakeout -f "Semantic Location History/2022" -o gpx
-F 2022.gpx
```

Example 3.19. Convert a single months' location history to GPX

```
gpsbabel   -i   googletakeout   -f   "Semantic   Location   Histo-
ry/2023/2023_MARCH.json" -o gpx -F 2023-03.gpx
```

GPS Tracking Key Pro text (land_air_sea)

This format can...

- read and write tracks

This format is derived from the xcsv format, so it has all of the same options as that format.

Read-only support for the text format exported by Land Air Sea's (Windows only) Past-Track software. This may also work for importing text formatted files from Victoria GPS Tracking, GPS Tracking Key and Land Air Sea's other devices.

Implementation

The text format of the GPS Tracking Key Pro contains one route coordinate per line and is of the format:

01-24-2011,09:12:30,N 48°51'57.9738",W 123°11'48.1354",20.5mph,83.8°,357ft

GPS Babel style file correctly imports all data except for bearing (which is un-needed). Since there is no way to create waypoints or routes on the device itself, the text file is read in as one large track.

GPS TrackMaker (gtm)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

Input and output support for waypoints, tracks and routes in the GPS TrackMaker [http://www.gpstm.com] binary format.

Code implemented by Gustavo Niemeyer.

GPSBabel arc filter file (arc)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

This format is used by GPSBabel itself as the input to the arc and polygon filters. See those filters for more information.

The arc format reads two numeric fields, a latitude and a longitude, in any format recognized as human readable and writes as simple degrees decimal. It really is intended for GPSBabel's own internal use more than general use, though it turns out to be a convenient way of expressing simple polylines and polygons.

GpsDrive Format (gpsdrive)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

GpsDrive way.txt file format. A space separated format file. Tested against GpsDrive v 1.30 found at gpsdrive.de [http://www.gpsdrive.de]. Contributed by Alan Curry.

GpsDrive Format for Tracks (gpsdrivetrack)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

Format used by GpsDrive to save tracks. Like GPSDRIVE a space separated format file. See above for a link to GpsDrive. Contributed by Tobias Minich.

GPX XML (gpx)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: snlen, suppresswhite, logpoint, urlbase, gpxver, humminbirdextensions, garminextensions, elevprec .

This is one of the most capable and expressive formats of all the file formats supported by GPSBabel. It is described at topografix.com [http://www.topografix.com/gpx.asp] and is supported by EasyGPS, ExpertGPS, and many other programs described at topografix.com [http://www.topografix.com/gpx_resources.asp]

GPSBabel's reader of this module attempts to preserve tags it doesn't really understand. It also tries to glean interesting data from:

- pocket queries from Geocaching.com, [http://www.geocaching.com]
- Garmin's "gpxx" GPX extensions,
- Humminbird's "h" GPX extensions.

`snlen` option

Length of generated shortnames.

When used with the `-s` to control shortnames, the snlen suboption to GPX controls how long the generated smartname will be. This can be useful for cases like writing GPX files to a GPS that has a fixed waypoint name length.

`suppresswhite` option

No whitespace in generated shortnames.

When used with the `-s` to generate smart shortnames, this suboption controls whether whitespace is allowed in the generated shortnames.

`logpoint` option

Create waypoints from geocache log entries.

When reading Groundspeak Pocket Queries [http://www.geocaching.com], the `logpoint` option creates additional waypoints from the log entries.

A typical use for this is to get coordinates read from "corrected coordinates" logs.

`urlbase` option

Base URL for link tag in output.

This is a fairly esoteric option. If the GPX file you are reading has only base pathnames (e.g "foo.html") the value you specify to this argument will be prepended to that. For example, "-o gpx,urlbase=c:\My Documents\Whatever" would result in the link to that waypoint being written to refer to `c:\My Document\WHatever\foo.html`

`gpxver` option

Target GPX version for output.

This option specifies the version of the GPX specification to use for output. The default version is 1.0 unless one or more of the input files is GPX 1.1, then it's 1.1. The only other valid value for this option is 1.1.

Notice that this is not a full scale XML schema conversion. In particular, if you have a GPX 1.0 file that has extended namespaces in it (such as a pocket query from Geocaching.com) just writing it with this option will result in a horribly mangled GPX file as we can't convert the schema data.

`humminbirdextensions` option

Add info (depth) as Humminbird extension.

Implies `gpxver=1.1`

`garminextensions` option

Add info (depth) as Garmin extension.

Write Garmin-specific extensions, when such data is available, to the GPX. Notably, depth, temperature, proximity, display color, ambient temperature, heart rate, and cadence are read when they are available in the source data. This data can, of course, only be read by GPX readers that know about the Garmin gpxtpx and gpxx extended namespaces.

Implies `gpxver=1.1`

`elevprec` option

Precision of elevations, number of decimals.

This option specifies the number of digits to be used when writing elevation values. Precision is the number of digits after the decimal point. The default precision is 3. We limit the number of places we write to improve the fidelity when round-tripping elevation, reduce file size, and reduce silliness in files caused by repeating decimals in insignificant digits.

As a guideline, three decimal places is 1 millimeter. To achieve precision approaching this limit leveling techniques are required. Precisions beyond this are not currently obtainable.

This value is ignored on read and has no impact on the internal representation of data.

Holux M-241 (MTK based) Binary File Format (m241-bin)

This format can...

- read waypoints
- read tracks

This format has the following options: csv .

The Holux m241 is a small datalogger using the MTK chipset, with a couple small differences in the binary format. In its default configuration, it can store ~100000 trackpoints with very limited data; to configure extended logging you can use the BT747 open source software bt747 [http://bt747.wiki.sourceforge.net] Waypoint storage is possible only if "recording reason" (RCR) is enabled in the settings.

Holux GPSport 245 is a datalogger with display suitable for cycling, walking and running. It can store ~200k trackpoints with limited data. The m241 and m241-bin format is able to automatically detect GPSport 245 data and handle the differences from Holux M-241 devices. Note: GP245 does not log any quality of the position.

Use the m241 format to connect with the unit serially and m241-bin to read files saved by the device.

csv option

MTK compatible CSV output file.

Specifies a filename into which MTK-compatible CSV output will be written.

Holux M-241 (MTK based) download (m241)

This format can...

- read tracks

This format has the following options: erase, erase_only, log_enable, csv, block_size_kb .

The Holux m241 is a small datalogger using the MTK chipset, with a couple small differences in the binary format. In its default configuration, it can store ~100000 trackpoints with very limited data; to configure extended logging you can use the BT747 open source software bt747 [http://bt747.wiki.sourceforge.net] Waypoint storage is possible only if "recording reason" (RCR) is enabled in the settings.

The m-241 came in two models. The yellow one, popular with photographers, internally used the Silicon Labs CP210X chipset to transform the internal and inaccessible serial port to the USB port that was familiar on the side. The drivers for that port can be found with SiLabs CP210x serial port drivers [https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx]. The red and white one existed very briefly before Holux went out of business. GPSBabel does not work with the red and white one.

Holux GPSport 245 is a datalogger with display suitable for cycling, walking and running. It can store ~200k trackpoints with limited data. The m241 and m241-bin format is able to automatically detect GPSport 245 data and handle the differences from Holux 241 devices. Note: GP245 does not log any quality of the position.

Use the m241 format to connect with the unit serially and m241-bin to read files saved by the device.

This module is also reported to handle the Holux M1000c.

Most of the loggers cannot receive bluetooth commands, they can only send data. Since GPSBabel needs to send commands to the GPS device it won't work. Download the data using the USB cable instead.

`erase` option

Erase device data after download.

This option erases the track log from the device after download.

`erase_only` option

Only erase device data, do not download anything.

This option will only erase the logger data. No data is downloaded.

This option is typically used as a second step after the data has been downloaded and verified.

`log_enable` option

Enable logging after download.

This option will enable the logger after download.

By default the logger is re-enabled when download is finished if previously were enabled. But if the download is aborted or failed the log functionality won't be enabled again.

`csv` option

MTK compatible CSV output file.

Note that this option is a bit of an oddity in the GPSBabel arsenal. This should probably be a "real" output type of its own instead of being bolted onto an input type.

`block_size_kb` option

Size of blocks in KB to request from device.

HTML Output (html)

This format can...

- write waypoints

This format has the following options: stylesheet, encrypt, logs, degformat, altunits .

GPSBabel's HTML output generates a single HTML file of all of the waypoints in the input file. It supports a number of Groundspeak GPX extensions and filters out potentially harmful HTML from the input file while maintaining almost all of the source HTML formatting. This makes this format well suited for

generating HTML to hand to programs like Plucker for putting in a PDA and especially so for "paperless caching" for Geocachers with pocket queries.

This format is similar to the text format.

The following command line reads a GPX file with Groundspeak extensions and writes an HTML file with encrypted hints that is rendered using a custom stylesheet:

```
gpsbabel -i gpx -f 12345.gpx -o html,stylesheet=green.css,encrypt -F
12345.html
```

`stylesheet` option

Path to HTML style sheet.

Use this option to specify a CSS style sheet to be used with the resulting HTML file.

`encrypt` option

Encrypt hints using ROT13.

Use this option to encrypt hints from Groundspeak GPX files.

`logs` option

Include groundspeak logs if present.

Use this option to include Groundspeak cache logs in the created document.

`degformat` option

Degrees output as 'ddd', 'dmm'(default) or 'dms'.

When GPSBabel writes coordinates, this option is consulted to see if it should write decimal degrees ('ddd') decimal minutes ('dmm') or degrees, minutes, seconds ('dms'). The default is 'dmm'.

`altunits` option

Units for altitude (f)eet or (m)etres.

This option should be 'f' if you want the altitude expressed in feet and 'm' for meters. The default is 'f'.

Humminbird tracks (.ht) (humminbird_ht)

This format can...

- read waypoints
- read and write tracks
- read routes

See the Humminbird format for docs on this.

Humminbird waypoints and routes (.hwr) (humminbird)

This format can...

- read and write waypoints
- read tracks
- read and write routes

This format supports:

Humminbird [http://www.humminbird.com] waypoints and routes (`.hwr` files)
Humminbird [http://www.humminbird.com] tracks (`.ht` files)
Humminbird [http://www.humminbird.com] `.dat` files. (These accompany the `.png` files you get when you take snapshots. There are also `.dat` files generated when making recordings, but those are not supported here.)

If you do "save all nav data" on the device, you'll get a `data.hwr` and a `000.ht` file on the flash card (on a 797 in the `matrix` directory).

The humminbird module can read all of these file formats, but you need to tell it which ones to write. By default, you get a `.hwr` file, to get a track (`.ht` file), use the "humminbird-track" format.

Note: `.dat` files are read-only, they only make sense together with their images.

Supported models:

797c2i SI
(They should all work, but this is the only one tested so far.)

Known limits:

max 12 characters for waypoint names.
max 20 characters for route and track names.
max 50 points per route. Use simplify filter (count=50 or less) if you have routes with more points!
max 21835 points per track.

Lowrance USR (lowranceusr)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: ignoreicons, writeasicons, merge, break, wversion, title, serialnum, description .

Many Lowrance [http://www.lowrance.com] systems have the ability to output their data to an external storage device. Early models (such as iFinder Hunt) supported an MMC card. Newer models (HDS, Hook,

Hook2, etc) support either an SD card or a microSD card. When exported the data is saved to the card as a file. Some models have the ability to export data in GPX format but the format native to Lowrance [http://www.lowrance.com] units is called USR. Typically the file created by the export operation will have a ".usr" suffix if is a native format or a ".gpx" suffix if is is a GPX format file.

Lowrance [http://www.lowrance.com] units currently use five different versions of USR files. These different USR versions store the four internal elements (waypoints, routes, trails, and event marker icons) using different formats. Some units do not support all four elements (typically event marker icons are not supported). Some USR formats also have multiple element versions (i.e. USR version 4 has two different formats for storing route data). Depending on the model of device you have you may be able to select the format for export data from any of the five USR formats or even the GPX format.

The following provides a high-level description of the multiple USR formats that are supported by the Lowrance product set (this is based on information contained in the 2018 Lowrance Hook2 Series Operator Manual [ftp://software.lowrance.com/Documents/Hook2-Series_OM_EN_988-11760-001_w.pdf]) and other sources.

User Data File version 2 - Legacy file format. This is the default output USR version used by GPSBabel. It contains only basic information on waypoints, routes, and trails.

User Data File version 3 - Legacy file format. Added depth information to Route waypoints. Supports trails with a maximum of 10,000 trail-points. Last version that supports Event Marker ICONs.

User Data File version 4 - Seems to be the best option for transferring data from older Lowrance units. Many of the counts (Number of Waypoints, Number of Routes, etc) were expanded from 16-bit integer values (maximum value of 65,535) to 32-bit (maximum value 2,147,483,647) USRv4 and above support a maximum of 20,000 trail-points (actually 24K and change). USRv4 and above and GPX support trails with trail-segments.

User Data File version 5 - Lowrance introduced universally unique identifiers (UUIDs) in this version.

User Data File version 6 - Latest format. Supports trail characteristics speed and temperature.

Lowrance systems support varying numbers of waypoints, routes, and tracks dependent on the specific device capabilities. Some early systems also supported entities called event marker icons. Event icon markers are represented by symbol, latitude and longitude data only. By default, event marker icons are converted by GPSBabel to waypoints on read with their name being generated in the format "Event Marker NNN" where NNN is replaced by the sequence number of the Event Marker ICON found in the input data. You have the option to ignore Event Icon Markers effectively removing them from the output data using the input option *ignoreicons*. On output, you can use the write option *writeasicons* to create event marker icons from waypoint data present in the input file.

The following tables detail the content format of USR data files.

- Lowrance USR Data File Contents
- Lowrance USR 2 and 3 Waypoint Object Format
- Lowrance USR 4, 5 and 6 Waypoint Object Format
- Lowrance USR 2 and 3 Route Object Format
- Lowrance USR 4, 5 and 6 Route Object Format
- Lowrance USR 2 and 3 Event Marker ICON Object Format
- Lowrance USR 2 and 3 Trail Object Format
- Lowrance USR 4, 5 and 6 Trail Object Format

Table 3.4. Lowrance USR Data File Contents

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	X	X	X	USR Format	4	Identifies the USR file format (integer). Valid values are 2 (USR 2), 3 (USR 3), 4 (USR 4), 5 (USR 5) and 6 (USR 6).
-	-	X	X	X	Data Stream Version	4	Appears only in USR 4, 5, and 6. Identifies the Data Stream Version (integer).
-	-	X	X	X	File Title Length	4	Appears only in USR 4, 5, and 6. Length of file title text (integer).
-	-	X	X	X	File Title	File Title Length * 2	Appears only in USR 4, 5, and 6. File title captured in UTF-16 character set.
-	-	X	X	X	File Creation Date Length	4	Appears only in USR 4, 5, and 6. Length of File creation date string (integer).
-	-	X	X	X	File Creation Date	File Creation Date Length * 2	Appears only in USR 4, 5, and 6. File creation date string captured in UTF-16 character set.
-	-	X	X	X	File Creation Date	4	Appears only in USR 4, 5, and 6. File creation date (integer).
-	-	X	X	X	File Creation Time	4	Appears only in USR 4, 5, and 6. File creation time (integer).
-	-	X	X	X	Unknown value	1	Appears only in USR 4, 5, and 6. Unknown data value (byte).
-	-	X	X	X	Unit Serial Number	4	Appears only in USR 4, 5, and 6. Serial number of Lowrance Unit creating the data file (integer).
-	-	X	X	X	File Description Length	4	Appears only in USR 4, 5, and 6. Length of File description string.
-	-	X	X	X	File Description	File Description Length * 2	Appears only in USR 4, 5, and 6. File description captured in UTF-16 character set.
X	X	-	-	-	Number of Waypoint Objects	2	In USR versions 2 and 3, the number of Waypoint objects is represented as a short integer. If the value is zero there are no Waypoint objects present in the data.
-	-	X	X	X	Number of Waypoint Objects	4	In USR versions 4, 5 and 6, the number of Waypoint objects is represented as an integer, If the value is zero there are no Waypoint objects present in the data.
X	X	X	X	X	Waypoint Objects	Sizeof Way-	Array of Waypoint Object data. Only present if the number of objects is greater than zero.

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
						point Object * Number of Waypoint Objects	
X	X	-	-	-	Number of Route Objects	2	In USR versions 2 and 3, the number of Route objects is represented as a short integer. If the value is zero there are no Route objects present in the data.
-	-	X	X	X	Number of Route Objects	4	In USR versions 4, 5 and 6, the number of Route objects is represented as an integer, If the value is zero there are no Route objects present in the data.
X	X	X	X	X	Route Objects	Sizeof Route Object * Number of Route Objects	Array of Route Object data. Only present if the number of objects is greater than zero.
X	X	-	-	-	Number of Event Marker ICON Objects	2	The number of Event Marker ICON objects is represented as a short integer. This data is only present in USR version 2 and 3 data files. If the value is zero there are no Event Marker ICON objects present in the data.
X	X	X	X	X	Event Marker ICON Objects	Sizeof Event Marker ICON Object * Number of Event Marker ICON Objects	Array of Event Marker ICON object data. Only present if the number of objects is greater than zero.
X	X	-	-	-	Number of Trail Objects	2	In USR versions 2 and 3, the number of Trail objects is represented as a short integer. If the value is zero there are no Trail objects present in the data.

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
-	-	X	X	X	Number of Trail Objects	4	In USR versions 4, 5 and 6, the number of Trail objects is represented as an integer, If the value is zero there are no Trail objects present in the data.
X	X	X	X	X	Trail Objects	Sizeof Trail Object * Number of Trail Objects	Array of Trail Object data. Only present if the number of objects is greater than zero.

Table 3.5. Lowrance USR 2 and 3 Waypoint Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	-	-	-	Waypoint Object Sequence Number	2	Sequence number of the Waypoint Object
X	X	-	-	-	Latitude	4	Waypoint Latitude value (float)
X	X	-	-	-	Longitude	4	Waypoint Longitude value (float)
X	X	-	-	-	Altitude	4	Waypoint Altitude (float)
X	X	-	-	-	Name Length	4	Length of Waypoint Name string (integer)
X	X	-	-	-	Name	Name Length	Name text, with USR 2 and 3 format this is UTF-8 representation.
X	X	-	-	-	Creation Time	4	Description
X	X	-	-	-	Unknown Data	4	Unknown data field. Only present if USR 2 or 3 format data generated by HOOK 2 unit
X	X	-	-	-	ICON Identifier	4	ICON identifier value (integer)
X	X	-	-	-	Description Length	4	Length of Description (integer)
X	X	-	-	-	Description	Description Length	Description text represented in UTF-8 characters.
X	X	-	-	-	Type	2	Waypoint type (short integer)
-	X	-	-	-	Depth	4	Waypoint depth (float)

Table 3.6. Lowrance USR 4, 5 and 6 Waypoint Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
-	-	-	X	X	Waypoint Universally Unique ID Number	16	Appears only in USR 5 and 6 format. Universally Unique Waypoint ID captured as four integers.
-	-	X	X	X	Unit Number	4	Serial Number of Unit capturing Waypoint (integer).
-	-	X	X	X	Waypoint Object Sequence Number	8	Sequence number of the Waypoint Object captured as a long integer.
-	-	X	X	X	Stream Version	4	Waypoint Stream version (integer).
-	-	X	X	X	Name Length	4	Length of Waypoint Name string (integer)
-	-	X	X	X	Name	Name Length	Name text, with USR 4, 5 and 6 format this is UTF-16 representation.
-	-	-	X	X	Unit Number	4	Serial Number of Unit capturing Waypoint (integer). Duplicate information.
-	-	X	X	X	Longitude	4	Waypoint Longitude
-	-	X	X	X	Flag	1	Waypoint Flag
-	-	X	X	X	Waypoint ICON Identifier	2	Waypoint ICON identifier value (short)
-	-	X	X	X	Waypoint Color	2	Waypoint color value (short)
-	-	X	X	X	Description Length	4	Length of Waypoint Description (integer)
-	-	X	X	X	Description	Description Length	Waypoint description text represented in UTF-16 characters.
-	-	X	X	X	Alarm Radius	4	Waypoint alarm radius (float)
-	-	X	X	X	Creation Date	4	Waypoint creation date (integer)
-	-	X	X	X	Creation Time	4	Waypoint creation time (integer)
-	-	X	X	X	Unused Byte	1	Unused byte (character)
-	-	X	X	X	Depth	4	Waypoint depth (float)
-	-	X	X	X	Lorain GRI	4	Lorain GRI (obsolete ??)
-	-	X	X	X	Lorain Tda	4	Lorain Tda (obsolete ??)
-	-	X	X	X	Lorain Tdb	4	Lorain Tdb (obsolete ??)

Table 3.7. Lowrance USR 2 and 3 Route Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	-	-	-	Waypoint Object Sequence Number	2	Sequence number of the Waypoint Object

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	X	X	X	Latitude	4	Waypoint Latitude value (float)
X	X	-	-	-	Longitude	4	Waypoint Longitude value (float)
X	X	-	-	-	Altitude	4	Waypoint Altitude (float)
X	X	-	-	-	Name Length	4	Length of Waypoint Name string (integer)
X	X	-	-	-	Name	Name Length	Name text, with USR 2 and 3 format this is UTF-8 representation.
X	X	-	-	-	Creation Time	4	Description
X	X	-	-	-	Unknown Data	4	Unknown data field. Only present if USR 2 or 3 format data generated by HOOK 2 unit
X	X	-	-	-	ICON Identifier	4	ICON identifier value (integer)
X	X	-	-	-	Description Length	4	Length of Description (integer)
X	X	-	-	-	Description	Description Length	Description text represented in UTF-8 characters.
X	X	-	-	-	Type	2	Waypoint type (short integer)
-	X	-	-	-	Depth	4	Waypoint depth (float)

Table 3.8. Lowrance USR 4, 5 and 6 Route Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
-	-	-	X	X	Route Universally Unique ID Number	16	Appears only in USR 5 and 6 format. Universally Unique Route ID captured as four integers
-	-	X	X	X	Unit Number	4	Serial Number of Unit capturing Route (integer)
-	-	X	X	X	Route Object Sequence Number	8	Sequence number of the Route Object captured as a long integer
-	-	X	X	X	Stream Version	4	Route Stream version (integer)
-	-	X	X	X	Name Length	4	Length of Route Name string (integer)
-	-	X	X	X	Name	Name Length	Name text, with USR 4, 5 and 6 format this is UTF-16 representation.
-	-	-	X	X	Unit Number	4	Serial Number of Unit capturing Route (integer). Only present in USR 5 and 6 format.
-	-	X	X	X	Number of Legs	4	Number of legs in route (integer)
-	-	X	X	X	Leg Objects	Sizeof Leg Object * Num-	Array of Leg Object data.

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
						ber of Legs	
-	-	-	X	X	Unknown Data	9	Unknown data fields. Appears to be three values, two integers and one character.
-	-	X	X	X	Unknown Data	1	Unknown data field. Possibly end of Route element indicator.

Table 3.9. Lowrance USR 4, 5 and 6 Route Leg Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
-	-	-	X	X	Route Universally Unique ID Number	16	Appears only in USR 5 and 6 format. Universally Unique ID for Waypoint that represents this leg of Route captured as four integers. Used to find Lat/Long values for leg.
-	-	X	-	-	Unit Number	4	Serial Number of Unit capturing Route Leg(integer). Used with Waypoint Sequence Number to find Lat/Long values for leg.
-	-	X	-	-	Waypoint Object Sequence Number	8	Sequence number of the Waypoint Object captured as a long integer. Used with Unit Number to find Lat/Long values for leg.

Table 3.10. Lowrance USR 2 and 3 Event Marker ICON Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	-	-	-	Latitude	4	Waypoint Latitude value (float)
X	X	-	-	-	Longitude	4	Waypoint Longitude value (float)
X	X	-	-	-	ICON Identifier	4	ICON identifier value (integer)

Table 3.11. Lowrance USR 2 and 3 Trail Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	-	-	-	Name Length	4	Length of Trail Name string (integer)
X	X	-	-	-	Name	Name Length	Name text, with USR 2 and 3 format this is UTF-8 representation.

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	-	-	-	Flag	1	= 0 Trail not displayed, = 1 Trail Visible (byte)
X	X	-	-	-	Number of Points	2	Number of points on trail (integer)
X	X	-	-	-	Max Points	2	Maximum number of points on trail (integer)
X	X	-	-	-	Trail Point Objects	Sizeof Trail Point Object * Number of Points	Array of Trail Point Object data

Table 3.12. Lowrance USR 2 and 3 Trail Point Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
X	X	-	-	-	Number of Section Points	2	Number of Lat-Long Pairs in Section (integer)
X	X	-	-	-	Lat-Lon Pair Objects	8 * Number of Section Points	Array of Lat-Long Pair values for Section Points. Each Latitude value occupies 4 bytes (float) and each Longitude value occupies 4 bytes (float)

Table 3.13. Lowrance USR 4, 5 and 6 Trail Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
-	-	X	X	X	Unit Number	4	Serial Number of Unit capturing Trail (integer).
-	-	X	X	X	Trail Object Sequence Number	8	Sequence number of the Waypoint Object captured as a long integer.
-	-	X	X	X	Stream Version	4	Trail Stream version (integer).
-	-	X	X	X	Name Length	4	Length of Trail Name string (integer)
-	-	X	X	X	Name	Name Length	Name text, with USR 4, 5 and 6 format this is UTF-16 representation.
-	-	X	X	X	Flag	4	Unknown flag value
-	-	X	X	X	Color	4	Trail Display Color
-	-	X	X	X	Desc Length	4	Length of Trail Description string (integer)
-	-	X	X	X	Description	Desc Length	Description text, with USR 4, 5 and 6 format this is UTF-16 representation.

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
-	-	X	X	X	Creation Date	4	Date Trail was captured
-	-	X	X	X	Creation Time	4	Time stamp when Trail was captured
-	-	X	X	X	Flag1	1	Unknown flag byte
-	-	X	X	X	Flag2	1	Unknown flag byte
-	-	X	X	X	Flag3	1	Unknown flag byte
-	-	X	X	X	Count	4	Unknown Count (integer)
-	-	X	X	X	Flag4	1	Unknown flag byte
-	-	X	X	X	Flag5	1	Unknown flag byte
-	-	X	X	X	Flag6	1	Unknown flag byte
-	-	X	X	X	Number of Trail Points	4	Number of Trail Point Objects (integer)
-	-	X	X	X	Trail Points Array	Sizeof Trail Point Object * Number of Trail Points	Array of Trail Point Objects

Table 3.14. Lowrance USR 4, 5 and 6 Trail Point Object Format

Present In USR Version					Field	Size (Bytes)	Description
2	3	4	5	6			
-	-	X	X	X	Unknown1	2	Unknown Trail Point value
-	-	X	X	X	Unknown2	1	Unknown Trail Point value
-	-	X	X	X	Creation Time	4	Trail Point creation time
-	-	X	X	X	Long-Lat Pair	8	Long-Lat Pair values for Trail Points. Latitude value occupies 4 bytes (float) and Longitude value occupies 4 bytes (float)
-	-	X	X	X	Attribute Count	4	Number of Trail Point Attributes.
-	-	X	X	X	Attribute Data	Attribute Count * 5	Unknown Data Array. Each entry has a one byte type identifier followed by a four byte value.

Some Lowrance units have the ability to export GPX [https://www.topografix.com/gpx.asp] (GPS Exchange Format) formatted data. Lowrance only provides minimal support for GPX export data on their HOOK2 series (only data available to the author). Refer to the official GPX 1.1 Schema Documentation [https://www.topografix.com/gpx/1/1/] for the complete schema.

Example 3.20. Lowrance GPX Export Data

```
<metadata>
  <time>xsd:dateTime</time>
  <depthunits>meters</depthunits>
  <tempunits>C</tempunits>
  <sogunits>m/s</sogunits>
</metadata>
<wpt lon="longitudeType" lat="latitudeType">
  <time>xsd:dateTime</time>
  <name>xsd:string</name>
  <sym>xsd:string</sym>
</rte>
  <name>xsd:string</name>
  <rtept lon="longitudeType" lat="latitudeType">
    <time>xsd:dateTime</time>
    <name>xsd:string</name>
    <sym>xsd:string</sym>
  </rtept>
</rte>
<trk>
  <name>xsd:string</name>
  <trkseg>
    <trkpt lon="longitudeType" lat="latitudeType">
      <time>xsd:dateTime</time>
    </trkpt>
  </trkseg>
</trk>
```

`ignoreicons` option

(USR input) Ignore event marker icons on read.

This option instructs GPSBabel to not convert event marker icons found in the input data to waypoints, but to instead disregard them altogether. These are present only in USR version 2 and 3 formats.

`writeasicons` option

(USR output) Treat waypoints as icons on write.

(USR output) This option causes waypoint information read from the input to be converted to event marker icons in the output.

`merge` option

(USR output) Merge into one segmented trail.

(USR output) This option merges all tracks into a single track with multiple segments.

`break` option

(USR input) Break segments into separate trails.

(USR input) Breaks multi-segment tracks into multiple separate tracks.

`wversion` option

(USR output) Write version.

While GPSBabel is capable of supporting all five USR formats for input processing it is currently only able to generate USR format 2, 3, or 4 due to the lack of detailed information on some of the data contained in the other formats.

By default, GPSBabel will generate USR version 2 data if the output format is specified as *lowranceusr*.

```
gpsbabel ... -o lowranceusr,wversion=2 -F blah.usr
```

is exactly the same as

```
gpsbabel ... -o lowranceusr -F blah.usr
```

Newer (post 2006 or so) Lowrance devices added a version three of their USR file format that adds waypoint depth. Lowrance recommends that this USR version be used when transferring user data to systems such as LMS, LCX, and so on. Specify "3" to output USR version 3 data on write.

To create a USR version 3 file that contains waypoint depth information use these output options:

```
gpsbabel ... -o lowranceusr,wversion=3 -F blah.usr
```

`title` option

(USR output) Output file title string.

This option can be used when writing a version 4 or higher Lowrance USR file to set the file title. If the option is not used the title will be "GPSBabel generated USR data file".

`serialnum` option

(USR output) Device serial number.

This option can be used when writing a version 4 or higher Lowrance USR file to set the device serial number. If the option is not used and the source of the data is a Lowrance USR file with a valid serial number then that serial number will be used, otherwise a value of 0 will be used.

`description` option

(USR output) Output file content description.

This option can be used when writing a version 4 or higher Lowrance USR file to set the description text. If the option is not used the description text will be "Waypoints, routes, and trails".

MiniHomer, a skyTraq Venus 6 based logger (download tracks, waypoints and get/set POI) (miniHomer)

This format can...

- read waypoints
- read tracks

This format has the following options: baud, dump-file, erase, first-sector, initbaud, last-sector, no-output, read-at-once, Home, Car, Boat, Heart, Bar, gps-utc-offset, gps-week-rollover .

Serial download protocol for GPS data loggers called "miniHomer". These loggers are based on Skytraq Venus 5 and Venus 6 chipsets, but with modified firmware. The miniHomer logger has five POI (or better: Point-to-Return, PTR?), which can be set programatically. The miniHomer module in gpsbabel is an extension of the skytraq module.

Following a list of devices which should be supported by this module (Note that not all of them have actually been tested, so if you can confirm that additional models work, please mail the gpsbabel-misc group with your success, tips, and any pertinent links for your model.)

Table 3.15. Devices supported by miniHomer module

Manufacturer	Model	USB (baud)	Bluetooth (baud)
Navin [http://navin.com.tw/miniHomer.htm] Z:NEX [http://www.znex.de/minihomer-details.html]	miniHomer	up to 230400	this device does not have bluetooth

Example 3.21. Command showing miniHomer download of tracks and erasing the logger on Linux

```
gpsbabel -i miniHomer,erase -f /dev/ttyUSB0 -o gpx -F out.gpx
```

Example 3.22. Command showing miniHomer erasing the logger without download on Linux

```
gpsbabel -i miniHomer,erase,no-output -f /dev/ttyUSB0
```

miniHomer has five POI called Home, *Car*, *Boat*, *Heart*, *Bar*. You can set the lla (Latitude, Longitude, Altitude) for each of the POI. The format is *<name>=<lat>:<lng>[:<alt>]* Once the according POI symbol is selected on miniHomer, the display shows you the direction and distance to the POI.

Example 3.23. Command showing miniHomer setting Car and Home POI

```
gpsbabel -i miniHome-
r,Car=36.790145:-6.352898,Home=-3.066667:37.359167:5895 -f /dev/
ttyUSB0 -o gpx -F out.gpx
```

Sets the Car/Home symbols' latitude longitude and altitude. If you select the Car/Home symbol on miniHomer, the display will show the direction and distance to this location as soon as it has a satellite fix.

baud option

Baud rate used for download.

The following baud rates can be used: 4800, 9600, 19200, 38400, 57600, 115200, 230400. Note that your logger might not support all of them (especially 230400 which isn't documented in the chipset manual, though there are known devices that are capable of this speed).

If

```
baud=0
```

(zero) download takes place at the baud rate the device is currently set to. This is especially useful for Bluetooth connections (if available) since they often don't allow changing the baud rate.

`dump-file` option

Dump raw data to this file.

This function is identical to the dump-file function of skytraq module: Writes raw data as it is read from the logger to the file given as this option's argument (additional to decoding it as usual). The resulting binary files can be read and decoded by the skytraq-bin format. Mainly useful for debugging/development purposes.

`erase` option

Erase device data after download.

Erase log buffer.

`first-sector` option

First sector to be read from the device.

This function is identical to the first-sector function of skytraq module.

The logger's memory is organized in sectors, serially numbered starting at 0. Each sector takes 4096 bytes of data. Typical devices hold about 250 sectors. The memory is always filled from sector 0 on, until it is full or the device being erased again by the user.

Normally you can safely omit this option. However, it might be useful to read data from erased devices: we observed that on erase, only the first two sectors are actually cleared. The following example shows how to read the remaining data:

Example 3.24. Command showing how to read data from an erased device

```
gpsbabel -i miniHomer,first-sector=2 -f /dev/ttyUSB0 -o gpx -F out.gpx
```

`initbaud` option

Baud rate used to init device (0=autodetect).

This function is identical to the init-baud file function of skytraq module.

The "initbaud" option might be helpful if autodetection fails or takes too long. With this option you can tell GPSBabel the baud rate the device is currently set to. In contrast, the option "baud" specifies the rate at which the actual download should take place. If it is different than "initbaud" (or the autodetected rate, if initbaud wasn't given), the initial setting will be restored after finishing the download.

Please note that miniHomer by default uses 38400bps and does not autodetect the port speed. If you need autodetect, start as

```
gpsbabel -i miniHomer,initbaud=0 -f /dev/ttyUSB0 -o gpx -F out.gpx
```

`last-sector` option

Last sector to be read from the device (-1: smart read everything).

A value of -1 (the default) enables automatic mode, i.e. reading is stopped when an empty sector is encountered. We observed that sometimes the device doesn't report the correct number of used sectors, which confuses the Windows software, so that it might not get all trackpoints. In contrast, our algorithm ensures that everything is being read (please report if it doesn't work for you).

`no-output` option

Disable output (useful with erase).

If this option is given, no GPS log data will be read from the device (unless "dump-file" is given too; in that case only decoding will be disabled).

`read-at-once` option

Number of sectors to read at once (0=use single sector mode).

If

```
read-at-once
```

>= 1, batch mode is enabled with that many sectors being read at a time. A value of zero disables batch mode and switches to single read mode. Not all devices support batch mode; in that case gpsbabel automatically switches to single read mode.

Under normal circumstances, the larger this number the faster the transfer. Reducing

```
read-at-once
```

or even switching to single sector mode might help when you get transmission errors/aborts.

`Home` option

POI for Home Symbol as lat:lng[:alt].

The device provides a location finder display supporting five locations "Home", "Car", "Boat", "Heart", "Bar". You can program the location of each either by a keypress on the device (which uses the actual position) or with GPSBabel (which lets you use any position) You can set the location of "Home" with the 'Home' option. Use ':' as the delimiter between latitude, longitude and altitude. You can leave altitude out, in which case it is assumed to be zero. Note that GPSBabel terminates after writing the location info to the device, i.e. no logging data will be read from it.

Example 3.25. Set the target location of the miniHomer Home POI

```
gpsbabel -i miniHomer,Home=-3.066667:37.359167:5895 -f /dev/ttyUSB0 -
o unicsv -F -
```

Sets the Home symbols' latitude to 3.066667S longitude to 37.359167E and altitude to 5895m. If you select the Home symbol on miniHomer, the display will show the direction and distance to this location as soon as it has a satellite fix.

`Car` option

POI for Car Symbol as lat:lng[:alt].

The device provides a location finder display supporting five locations "Home", "Car", "Boat", "Heart", "Bar". You can program the location of each either by a keypress on the device (which uses the actual position) or with GPSBabel (which lets you use any position) You can set the location of "Car" with the 'Car' option. Use ':' as the delimiter between latitude, longitude and altitude. You can leave altitude out, in which case it is assumed to be zero. Note that GPSBabel terminates after writing the location info to the device, i.e. no logging data will be read from it.

Example 3.26. Set the target location of the miniHomer Car POI

```
gpsbabel -i miniHomer,Car=-25.272309:153.235330 -f /dev/ttyUSB0 -o
unicsv -F -
```

Sets the Car symbols' latitude to 25.272309S longitude to 153.235330E and altitude to 0m. If you select the Car symbol on miniHomer, the display will show the direction and distance to this location as soon as it has a satellite fix.

`Boat` option

POI for Boat Symbol as lat:lng[:alt].

The device provides a location finder display supporting five locations "Home", "Car", "Boat", "Heart", "Bar". You can program the location of each either by a keypress on the device (which uses the actual position) or with GPSBabel (which lets you use any position) You can set the location of "Boat" with the 'Boat' option. Use ':' as the delimiter between latitude, longitude and altitude. You can leave altitude out, in which case it is assumed to be zero. Note that GPSBabel terminates after writing the location info to the device, i.e. no logging data will be read from it.

Example 3.27. Set the target location of the miniHomer Boat POI

```
gpsbabel -i miniHomer,Boat=32.29287:-64.77527 -f /dev/ttyUSB0 -o unicsv
-F -
```

Sets the Boat symbols' latitude to 32.29287N longitude to 64.77527E and altitude to 0m. If you select the Home symbol on miniHomer, the display will show the direction and distance to this location as soon as it has a satellite fix.

`Heart` option

POI for Heart Symbol as lat:lng[:alt].

The device provides a location finder display supporting five locations "Home", "Car", "Boat", "Heart", "Bar". You can program the location of each either by a keypress on the device (which uses the actual position) or with GPSBabel (which lets you use any position) You can set the location of "Heart" with the 'Heart' option. Use ':' as the delimiter between latitude, longitude and altitude. You can leave altitude out,

in which case it is assumed to be zero. Note that GPSBabel terminates after writing the location info to the device, i.e. no logging data will be read from it.

Example 3.28. Set the target location of the miniHomer Heart POI

```
gpsbabel -i miniHomer,Heart=36.1269:-115.1698 -f /dev/ttyUSB0 -o unicsv
-F -
```

Sets the Heart symbols' latitude to 36.1269N longitude to 115.1698W and altitude to 0m. If you select the Heart symbol on miniHomer, the display will show the direction and distance to this location as soon as it has a satellite fix.

`Bar` option

POI for Bar Symbol as lat:lng[:alt].

The device provides a location finder display supporting five locations "Home", "Car", "Boat", "Heart", "Bar". You can program the location of each either by a keypress on the device (which uses the actual position) or with GPSBabel (which lets you use any position) You can set the location of "Bar" with the 'Bar' option. Use ':' as the delimiter between latitude, longitude and altitude. You can leave altitude out, in which case it is assumed to be zero. Note that GPSBabel terminates after writing the location info to the device, i.e. no logging data will be read from it.

Example 3.29. Set the target location of the miniHomer Bar POI

```
gpsbabel -i miniHomer,Bar=38.99809:-86.34662 -f /dev/ttyUSB0 -o unicsv
-F -
```

Sets the Bar symbols' latitude to 38.99809N longitude to 86.34662W and altitude to 0m. If you select the Bar symbol on miniHomer, the display will show the direction and distance to this location as soon as it has a satellite fix.

`gps-utc-offset` option

Seconds that GPS time tracks UTC (0: best guess).

`gps-week-rollover` option

GPS week rollover period we're in (-1: best guess).

Mobile Garmin XT Track files (garmin_xt)

This format can...

- read tracks

This format has the following options: ftype, trk_header .

`ftype` option

Garmin Mobile XT ([ATRK]/STRK).

`trk_header` option

Track name processing option ([0]-nrm/1-ign).

MTK Logger (iBlue 747,...) Binary File Format (mtk-bin)

This format can...

- read waypoints
- read tracks

This format has the following options: csv .

Binary file protocol converter for MTK based GPS loggers. This format reads the raw binary format created by the MTK Windows application and outputs to other formats supported by GPSBabel When using the csv option a MTK application compatible output file will also be created.

It has been tested with Transystem i-Blue 747™ but other devices should work as well (Qstarz BT-Q1000, iTrek Z1, ...)

All position items (including button push) will be listed as trackpoints in the output. Log items due to button push are presented as waypoints. In theory we would not add waypoints to the list of trackpoints. But as the MTK logger restart the log session from the button press we would loose a trackpoint unless we include/duplicate it.

Transystem i-Blue 747 [http://www.transystem.com.tw/p-gps-iblue747.htm]

Example 3.30. Convert MTK binary trackpoints to GPX

```
gpsbabel -t -i mtk-bin,csv=extra.csv -f data.bin -o gpx -F out.gpx
```

Additionally a CSV output file is created.

`csv` option

MTK compatible CSV output file.

Specifies a filename into which MTK-compatible CSV output will be written.

Note that this option is a bit of an oddity in the GPSBabel arsenal. This should probably be a "real" output type of its own instead of being bolted onto an input type.

MTK Logger (iBlue 747,Qstarz BT-1000,...) download (mtk)

This format can...

- read waypoints

- read tracks

This format has the following options: erase, erase_only, log_enable, csv, block_size_kb .

This format is the serial download protocol for the MTK chips. Mediatek's MT3301/3179 (MTKv1) and MT3318 (MTKv2) chips are used in a large number of products sold under different names.

Many GPS products, especially of the data logger variety, expose the Mediatek protocol to the user via USB. Some modify Mediatek's protocol in minor ways, but the core protocol is very commonly seen in loggers.

The Holux M-241 and GPSport 245 are examples of a device using an incompatible variation of the MTK protocol.

The following products are known or are expected to work with this module. As the products are typically very low cost, they tend to have a short product life cycle and are often imported to different areas under different names. Keeping track of the list is difficult. Often the "same" GPS is sold in different plastic or with different Windows software or different options such as compass or motion sensors or charging cables with different model numbers. If you can confirm success with others, please share with us.

Table 3.16. Devices supported by MTK module

Product	Confirmed to work	Notes
iBlue 821	Yes	Available from Semsons [http://www.semsons.com/i821ulblgp-sr.html]
iBlue 747, 747A+	Yes	Available from Semsons [http://www.semsons.com/i74blgpsda-lo.html]
QStarz BT-1000, BT-Q1000X, BT-1000eX	Yes	
iTrek Z1		

The Mediatek chip offers a native serial port. Data logger designers frequently pair this with commodity USB/Serial converter internally. So these devices typically look like Prolific, FTDI, or Silab usb/serial devices to the host OS. You'll need drivers for that whatever chip your product uses for your operating system. For the "A+ GPS Recorder", the Silicon Labs CP210x chip [http://www.silabs.com/products/mcu/pages/usbtouartbridgevcpdrivers.aspx] is used. On OS/X, you'll get awesome device names like /dev/cu.usbmodem1d13410 - fortunately, our GUI makes that multiple choice so you don't have to guess.

Downloaded data will be stored in data.bin file in the current directory together with the chosen output format. This is a rather uncommon feature in GPSBabel's formats and is likely to change in future versions.

See mtk-bin on how trackpoints/waypoints are handled

Example 3.31. Command showing MTK download track and waypoints and erase on Linux

```
gpsbabel -t -w -i mtk,erase -f /dev/ttyUSB0 -o gpx -F out.gpx
```

For more info and tweaks on MTK based loggers: MTK Tips and Tweaks [http://www.gpspassion.com/forumsen/topic.asp?TOPIC_ID=81990] and iBlue 747 Logger [http://www.gpspassion.com/forumsen/top-

ic.asp?TOPIC_ID=81315] For info about the used log format, see MTK binary format [http://spreadsheets.google.com/pub?key=pyCLH-0TdNe-5N-5tBokuOA&gid=5]

Most of the loggers cannot receive bluetooth commands; they can only send data. Since GPSBabel needs to send commands to the GPS device it won't work. Download the data using the USB cable instead.

`erase` option

Erase device data after download.

This option erases the track log from the device after download.

`erase_only` option

Only erase device data, do not download anything.

This option will only erase the logger data. No data is downloaded.

This option is typically used as a second step after the data has been downloaded and verified.

`log_enable` option

Enable logging after download.

This option will enable the logger after download.

By default the logger is re-enabled when download is finished if previously were enabled. But if the download is aborted or failed the log functionality won't be enabled again.

`csv` option

MTK compatible CSV output file.

This option will create an additional CSV output file. The CSV file is compatible with the original MTK logger application.

`block_size_kb` option

Size of blocks in KB to request from device.

National Geographic Topo .tpg (waypoints) (tpg)

This format can...

- read and write waypoints

This format has the following options: datum .

National Geographic Topo! Waypoint and Route Format. This module reads and writes .TPG files created by various editions of NG Topo! Reading/writing of route data is not supported yet.

Contributed by Alex Mottram.

`datum` option

Datum (default=NAD27).

The option 'datum="datum name"' can be used to override the default of NAD27 ("N. America 1927 mean") which is correct for the continental U.S.

Any legal datum supported by GPSBabel may be used. For example, points in Hawaii should use "Old Hawaiian_mean".

National Geographic Topo 2.x .tpo (tpo2)

This format can...

- read tracks

This module reads tracks from .TPO files created by National Geographic Topo! version 2.x

Contributed by Steve Chamberlin.

National Geographic Topo 3.x/4.x .tpo (tpo3)

This format can...

- read waypoints
- read tracks
- read routes

This module reads .TPO files created by National Geographic Topo! version 3.x and 4.x. It will read tracks, routes, waypoints, map notes, symbols, and text notes. The latter three are converted to waypoints.

Contributed by Curt Mills.

NMEA 0183 sentences (nmea)

This format can...

- read and write waypoints
- read and write tracks

This format has the following options: snlen, gprmc, gpgga, gpvtg, gpgsa, date, get_posn, pause, append_positioning, baud, gisteq, ignore_fix .

This format is the file representation of the NMEA (National Marine Electronics Association) 0183 log and waypoint format for GPS devices. Some hardware and software that work with NMEA-0183 formatted data include:

GPS Data Logger [http://home-VisualGPS [http://www.visualgp-SparkFun GPS Datalogger [http://
pages.tig.com.au/~robk/ s.net/VisualGPSce/default.htm] www.sparkfun.com/]
datalogger.html]
GPS TrackMaker [http://www.gp-GPS Utility [http://www.gp-AMOD 3080 GPS
stm.com/eng/features_eng.htm] su.co.uk/]

GPSMaster [http://www.gpsmas-ter.nl/]	GeoConv [http://www.kolum-bus.fi/eino.uikkanen/geocon-vgb/index.htm]	Wintec WPL-1000 GPS
NMEAlog [http://www.sil-com.com/~rwhately/index.html]	CommLinx GPS recorder [http://www.comm-linx.com.au/GPS_recorder.htm]	Sony GPS_CS1

This module also supports realtime tracking which allows realtime position reports from a GPS, such as one connected serially, over Bluetooth, or a USB module emulating a serial port, to be used with selected output formats. Just specify an input file that is the device name such as COM1: for Windows or a device-dependent name like /dev/cu.usbserial for Mac or /dev/ttyUSB0 for Linux. (Note that serial device names vary on Mac and Linux.)

When used in realtime tracking mode, if GPSBabel does not sense incoming NMEA sentences arriving from the port, it will send Sirf "reset to NMEA" commands to the port at a variety of speeds in an attempt to communicate with an attached GPS. This lets devices like the Microsoft GPS or Pharos GPS that are Sirf chips with an integrated USB/Serial adapter work with this input format.

`snlen` option

Max length of waypoint name to write.

This option specifies the maximum length to be used for waypoint names in the GPWPL sentence. Longer names will be shortened to no more than this length, but all waypoint names will remain unique.

`gprmc` option

Read/write GPRMC sentences.

This option tells GPSBabel whether to read (on input) or write (on output) GPRMC sentences. The default is to read or write GPRMC sentences. To disable GPRMC sentences, specify `gprmc=0`.

GPRMC sentences contain the "recommended minimum" positional information, including date and time, heading, and velocity. Note that they do not include altitude. For altitude, you will have to include GPGGA sentences.

`gpgga` option

Read/write GPGGA sentences.

This option tells GPSBabel whether to read (on input) or write (on output) GPGGA sentences. The default is to read or write GPGGA sentences. To disable GPGGA sentences, specify `gpgga=0`.

GPGGA sentences contain the location and quality of the GPS position fix.

`gpvtg` option

Read/write GPVTG sentences.

This option tells GPSBabel whether to read (on input) or write (on output) GPVTG sentences. The default is to read or write GPVTG sentences. To disable GPVTG sentences, specify `gpvtg=0`.

GPVTG sentences contain information about the heading and the speed at the time of the fix. They do not contain any location information; for that you will need either or both of GPGGA or GPRMC.

gpgsa option

Read/write GPGSA sentences.

This option tells GPSBabel whether to read (on input) or write (on output) GPGSA sentences. The default is to read or write GPGSA sentences. To disable GPGSA sentences, specify `gpgsa=0`.

GPGSA sentences contain information on the quality of the positional fix and the individual satellites from which it was derived. However, GPSBabel neither reads nor writes the individual satellite data. On input, the satellite fields are ignored and on output they are left blank.

date option

Complete date-free tracks with given date (YYYYMMDD)..

On input, track points with times but no dates will have this date applied.

This is necessary because some NMEA sentences contain times but no dates. If this option is not specified and the date cannot be determined from one or more of the available NMEA sentences, the tracks will be discarded.

get_posn option

Return current position as a waypoint.

This options, when specified, returns the current position as a single waypoint.

pause option

Decimal seconds to pause between groups of strings.

This option tells GPSBabel to pause between individual track records when used on output. This may be used with appropriate external software or hardware to simulate a GPS receiver for testing purposes. On Unix, for example, you may use a named pipe to feed the output from GPSBabel to gpsd.

If a value for this option is specified, it is in seconds and it may be either a whole number of seconds or a fraction (e.g. 0.5 for a 1/2 second pause between trackpoints.)

If this option is specified with a negative value, the time between adjacent trackpoints will be computed and used for the length of the pause. That is, if your trackpoints are 5 seconds apart, GPSBabel will pause 5 seconds between trackpoints.

Note that very long tracks may be subject to clock drift, as GPSBabel does not take into account the amount of time it may take to write the NMEA sentences. Also, there is no guarantee that it will pause for exactly the specified number of seconds between samples; different operating systems will allow greater or lesser precision for timers, so actual precision may be as much as plus or minus 100 milliseconds.

If you are using this option with compressed or simplified tracks from your handheld GPS receiver, you might find the interpolate filter useful.

append_positioning option

Append realtime positioning data to the output file instead of truncating.

When writing NMEA realtime positioning data, append to the output file instead of truncating it on each successive position fix.

baud option

Speed in bits per second of serial port (baud=4800).

To the "nmea" module, the "baud" option specifies the baud rate of the serial connection when used with the real-time tracking option.

gisteq option

Write tracks for Gisteq Phototracker.

This option writes the Gisteq format - which has the extension of .GPS - to allow third-party GPS hardware with the Gisteq PhotoTrackr software.

The Gisteq PhotoTrackr is a GPS data logger hardware and software package that allows one to easily record the locations of where the user has taken photos. The PhotoTrackr software works by comparing EXIF timestamps in digital photos with the timestamps in the tracking data. In doing so, the software plots the locations of the photos using Google Maps. The logging format used by the Gisteq hardware is very close to NMEA format, but with a few small quirks.

More information can be found at the Gisteq [http://www.gisteq.com/] site.

ignore_fix option

Accept position fixes in gpgga marked invalid.

OpenStreetMap data files (osm)

This format can...

- read and write waypoints
- write tracks
- read and write routes

This format has the following options: tag, tagnd, created_by .

This format is used to exchange data with the OpenStreetMap [http://www.openstreetmap.org] project. The main goal of this collaborative project is to create free editable maps.

These data files are XML based. Every GPS element (way or node) described by the files has a unique number as identifier. When we write OSM data files and don't know something about the id's, negative numbers will be used as identifier. This has been tested with JOSM [http://wiki.openstreetmap.org/index.php/JOSM].

Because the resulting timestamps of OSM ways differ from real GPS tracks, we read OSM ways into routes. On the output side we write all available routes and tracks into the osm target file.

tag option

Write additional way tag key/value pairs.

With this option you can preset OSM features [http://wiki.openstreetmap.org/index.php/Map_Features] (tags) on all exported ways.

```
gpsbabel -i gdb -f ways.gdb -o osm,tag="highway:motorway" -F ways.osm
```

tagnd option

Write additional node tag key/value pairs.

With this option you can preset OSM features [http://wiki.openstreetmap.org/index.php/Map_Features] (tags) on every written nodes.

```
gpsbabel -i gdb -f nodes.gdb -o osm,tagnd="amenity:pub;building:yes" -
F nodes.osm
```

created_by option

Use this value as custom created_by value.

Use this value as custom created_by value in an OSM file.

With this option, the given string is added as the 'created_by' field in all the created nodes and ways.

```
gpsbabel -i INTYPE -f INFILE -o osm,created_by=somestring -F out.osm
```

If an empty string is given, the 'created_by' tag is omitted altogether.

```
gpsbabel -i INTYPE -f INFILE -o osm,created_by= -F out.osm
```

OziExplorer (ozi)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: pack, snlen, snwhite, snupper, snunique, wptfgcolor, wptbgcolor, proximity, altunit, proxunit, codec .

OziExplorer Waypoint Format - Another CSV format file. Tested against OziExplorer v 3.90.3a / Shareware. Contributed by Alex Mottram

pack option

Write all tracks into one file.

In normal case GPSBabel creates for each track a separate file (track.plt, track-1.plt, ...). With this option all tracks will be written into one file. A '1' in the third field of the trackpoint record signals the beginning of a new track.

```
gpsbabel -i gpx -f tracks.gpx -o ozi,pack -F track
```

snlen option

Max synthesized shortname length.

This option allows you to specify the length of waypoint names written to this format when used with the `-s` option.

snwhite option

Allow whitespace synth. shortnames.

This option forces waypoint names generated with `-s` to allow whitespace in the names.

snupper option

UPPERCASE synth. shortnames.

When specified, this option will force generated shortnames to be in all uppercase letters.

snunique option

Make synth. shortnames unique.

When specified, this option will force the generated waypoint names to be unique.

wptfgcolor option

Waypoint foreground color.

This option allows you to specify a foreground color of a waypoint. You can specify it as either a decimal number or one of the standard web colors.

wptbgcolor option

Waypoint background color.

This option allows you to specify a background color of a waypoint. You can specify it as either a decimal number or one of the standard web colors.

proximity option

Proximity distance.

This option, specified in meters, allows you to set the proximity of written waypoints.

altunit option

Unit used in altitude values.

By default the ozi module uses feet as altitude unit. With this option you can specify also 'Meters' (m) as unit for altitude values.

`proxunit` option

Unit used in proximity values.

By default the proximity values are handled in meters. With this option you can now specify (m)iles, (k)ilometers or (n)autical miles as the units for proximity when reading or writing ozi files.

codec option

codec to use for reading and writing strings (default windows-1252).

This lets you override the default codec of 'windows-1252'. As an input option the codec should correspond to the encoding of the input file. As an output option it sets the encoding of the output file.

Qstarz BL-1000 (qstarz_bl-1000)

This format can...

- read waypoints
- read tracks

This format is used by the Qstarz BL-1000GT and BL-1000ST.

These Qstarz devices are file based. GPS logs are saved as *.BIN files to the SESSION/GPSLog folder.

See You flight analysis data (cup)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

This format supports flight analysis data from the See You [https://naviter.com/seeyou-makes-you-a-better-pilot/] program.

Position information is preserved, but the aviation-specific information such as runway length and airport frequency, are written as blanks and ignored on read.

Tasks are not supported.

SkyTraq Venus based loggers (download) (skytraq)

This format can...

- read waypoints

- read tracks

This format has the following options: erase, targetlocation, configlog, baud, initbaud, read-at-once, first-sector, last-sector, dump-file, no-output, gps-utc-offset, gps-week-rollover .

Serial download protocol for GPS data loggers based on Skytraq Venus 5 and Venus 6 chipsets. This chipset is used by a number of devices from different manufacturers. If your logger came with the Windows software iTravelTech GPS Photo Tagger, chances are that you can use this format to read its memory.

Following a list of devices which should be supported by this module (Note that not all of them have actually been tested, so if you can confirm that additional models work, please mail the gpsbabel-misc group with your success, tips, and any pertinent links for your model.):

Table 3.17. Devices supported by skytraq module

Manufacturer	Model	USB (baud)	Bluetooth (baud)
SJA	"3-in-1" GPS logger	up to 230400	9600
Navilock	BT-455PDL	untested	untested
Polaris	Travel Honey	up to 230400	9600
Pearl Diffusion	Keymate STV-5	untested	untested
Canmore	GT-730FL-S	untested	n/a
Canmore	GT-750F	untested	untested
Gisteq	DPL900	up to 230400	untested
Adapt Mobile	Keychain Pro	untested	untested
Adapt Mobile	Keychain Pro	9600	9600

Windows users of GPSBabel version 1.5.2 or less may have to explicitly specify a bit rate of 115200 or lower.

Example 3.32. Command showing skytraq download of tracks and erasing the logger on Linux

```
gpsbabel -i skytraq,erase -f /dev/ttyUSB0 -o gpx -F out.gpx
```

Example 3.33. Command showing skytraq erasing the logger without download on Linux

```
gpsbabel -i skytraq,erase,no-output -f /dev/ttyUSB0
```

If available, reading the logger using bluetooth should also work. However, many devices support only one specific baud rate over bluetooth, e.g. 9600. In that case you should use the option

```
baud=0
```

to tell GPSBabel to use that default baud rate:

Example 3.34. Command showing skytraq download tracks via bluetooth on Linux

```
rfcomm bind 0 <bdaddr>
```

```
gpsbabel -i skytraq,baud=0 -f /dev/rfcomm0 -o gpx -F out.gpx
```

erase option

Erase device data after download.

targetlocation option

Set location finder target location as lat,lng.

The device provides a location finder built from eight LEDs and can use those LEDs to guide you to a location. You can set the target location with the 'targetlocation' option. Use ':' as the delimiter between latitude and longitude. Note that GPSBabel terminates after writing the location info to the device, i.e. no logging data will be read from it.

Example 3.35. Set the target location of the Skytraq location finder

```
gpsbabel -i skytraq,targetlocation=12.34:-56.78 -f /dev/ttyUSB 0 -o
unicsv -F -
```

Sets latitude and longitude of the location finder to N12.34 and W56.78 respectively. The arrows on the device will point you to this location as soon as it has a satellite fix.

configlog option

Configure logging parameter as tmin:tmax:dmin:dmax.

Set the logging configuration as tmin:tmax:dmin:dmax. Here tmin and tmax are in seconds, and dmin and dmax in meters. With dt = time since last log, dx = distance since last log, and v the current speed, the device logs if

(dt > tmin and dx >= dmin and v >= vmin) or dt > tmax or dx > dmax or v > vmax

If you use this option, vmin is fixed at 0 and vmax at 65535 km/h.

Example. Set the device to log every 6 seconds (or 10km, whichever happens first!)

Example 3.36. Set the logging parameters for Skytraq device

```
gpsbabel -i skytraq,configlog=6:3600:0:10000 -f /dev/ttyUSB0
```

baud option

Baud rate used for download.

The following baud rates can be used: 4800, 9600, 19200, 38400, 57600, 115200, 230400. Note that your logger might not support all of them (especially 230400 which isn't documented in the chipset manual, though there are known devices that are capable of this speed).

If

```
baud=0
```

(zero) download takes place at the baud rate the device is currently set to. This is especially useful for Bluetooth connections since they often don't allow changing the baud rate.

`initbaud` option

Baud rate used to init device (0=autodetect).

The "initbaud" option might be helpful if autodetection fails or takes too long. With this option you can tell GPSBabel the baud rate the device is currently set to. In contrast, the option "baud" specifies the rate at which the actual download should take place. If it is different than "initbaud" (or the autodetected rate, if initbaud wasn't given), the initial setting will be restored after finishing the download.

`read-at-once` option

Number of sectors to read at once (0=use single sector mode).

If

```
read-at-once
```

>= 1, batch mode is enabled with that many sectors being read at a time. A value of zero disables batch mode and switches to single read mode. Not all devices support batch mode; in that case gpsbabel automatically switches to single read mode.

Under normal circumstances, the larger this number the faster the transfer. Reducing

```
read-at-once
```

or even switching to single sector mode might help when you get transmission errors/aborts.

`first-sector` option

First sector to be read from the device.

The logger's memory is organized in sectors, serially numbered starting at 0. Each sector takes 4096 bytes of data. Typical devices hold about 250 sectors. The memory is always filled from sector 0 on, until it is full or the device being erased again by the user.

Normally you can safely omit this option. However, it might be useful to read data from erased devices: we observed that on erase, only the first two sectors are actually cleared. The following example shows how to read the remaining data:

Example 3.37. Command showing how to read data from an erased device

```
gpsbabel -i skytraq,first-sector=2 -f /dev/ttyUSB0 -o gpx -F out.gpx
```

`last-sector` option

Last sector to be read from the device (-1: smart read everything).

A value of -1 (the default) enables automatic mode, i.e. reading is stopped when an empty sector is encountered. We observed that sometimes the device doesn't report the correct number of used sectors, which confuses the Windows software, so that it might not get all trackpoints. In contrast, our algorithm ensures that everything is being read (please report if it doesn't work for you).

`dump-file` option

Dump raw data to this file.

Writes raw data as it is read from the logger to the file given as this option's argument (additional to decoding it as usual). The resulting binary files can be read and decoded by the skytraq-bin format. Mainly useful for debugging/development purposes.

`no-output` option

Disable output (useful with erase).

If this option is given, no GPS log data will be read from the device (unless "dump-file" is given too; in that case only decoding will be disabled).

`gps-utc-offset` option

Seconds that GPS time tracks UTC (0: best guess).

gps-utc-offset is used to override the built-in table of offsets of the offset between GPS time and UTC time. This chipset reports only GPS time to the host and relies on software to know every time an adjustment is made. Since GPS time offsets can change without a new version of GPSBabel is released, those that care about total accuracy can override it.

`gpsbabel`
`-i skytraq.bin,gps-utc-offset=15 -f` *`filename.bin`*

Indicates that GPS is ahead of UTC by fifteen seconds, as was the case in 2009.

Consult formal explanation of GPS time vs. UTC time [http://tycho.usno.navy.mil/leapsec.html] if you're into that.

`gps-week-rollover` option

GPS week rollover period we're in (-1: best guess).

gps-week-rollover is used to override the best-guessing of GPS week rollover period we're currently in: skytraq log data contains dates in the form of GPS weeks, which roll over to 0 every 1024 weeks (close to 20 years).

Table 3.18. GPS week rollover dates

Starting from:	**gps-week-rollover value:**
1980-01-06 00:00:00 UTC	0
1999-08-21 23:59:47 UTC	1
2019-04-06 23:59:42 UTC	2

The default behavior when gps-week-rollover isn't given (or is a negative number) is to assume the input data has been logged within the preceding 1024 weeks from the time gpsbabel is run, which should be perfectly fine in almost all cases.

The following example:

```
gpsbabel -i skytraq.bin,gps-week-rollover=1 -f filename.bin
```

indicates that logged data is assumed to be from the period between 21/22 Aug 1999 and 6/7 April 2019.

SkyTraq Venus based loggers Binary File Format (skytraq-bin)

This format can...

- read waypoints
- read tracks

This format has the following options: first-sector, last-sector, gps-utc-offset, gps-week-rollover .

Reads the binary format of GPS data loggers based on Skytraq Venus 5 and Venus 6 chipsets. This can be used to read raw binary files created with the "dump-file" option of the skytraq format. Mainly useful for debugging/development purposes.

`first-sector` option

First sector to be read from the file.

`last-sector` option

Last sector to be read from the file (-1: read till empty sector).

`gps-utc-offset` option

Seconds that GPS time tracks UTC (0: best guess).

`gps-week-rollover` option

GPS week rollover period we're in (-1: best guess).

SubRip subtitles for video mapping (.srt) (subrip)

This format can...

- write tracks

This format has the following options: video_time, gps_time, gps_date, format .

This is a write-only format for geotagging videos. It is used for videomapping, i.e. filming a trip while creating a GPS trace. It will produce a subtitle file in SubRip (.srt) format.

Unless the video and the GPS trace start at exactly the same time, you will need to synchronize both. For this purpose, film the display of your GPS receiver (or any other device) showing GPS time. (Important: you need precise GPS time for this; local time, especially from an inaccurate clock, will not do for this.)

Determine the position in the video at which the GPS time is visible (in hours, minutes and seconds from the beginning of the video) and the GPS date and time shown. Specify these as command line options; you will need to do this once for each video file.

To use these files, choose the same name as for the associated video, changing just the extension to .srt, and place the srt file in the same directory as the video. Open the video in a media player and the GPS coordinates will be shown as subtitles (tested on VLC, your mileage may vary).

`video_time` option

Video position for which exact GPS time is known (hhmmss[.sss], default is 00:00:00,000).

Video position (relative to beginning of video) for which the corresponding GPS timestamp is known.

Format is hhmmss. If omitted, 0:00:00 (beginning of video) is assumed.

The GPS timestamp can be set with the `gps_time` and `gps_date` options.

`gps_time` option

GPS time at position video_time (hhmmss[.sss], default is first timestamp of track).

The time part of the GPS timestamp which corresponds to a known position in the video.

Format is hhmmss. This option must be used together with `gps_date`; if one or both are missing, the timestamp of the first GPS trackpoint is used.

The video position to which the timestamp corresponds can be set with the `video_time` option.

`gps_date` option

GPS date at position video_time (yyyymmdd, default is first timestamp of track).

The date part of the GPS timestamp which corresponds to a known position in the video.

Format is yyyymmdd. This option must be used together with `gps_time`; if one or both are missing, the timestamp of the first GPS trackpoint is used.

The video position to which the timestamp corresponds can be set with the `video_time` option.

`format` option

Format for subtitles.

Format for output subtitles.

Table 3.19. Supported format characters for subrip

format char	description
%s	speed in km/h

format char	description
%e	elevation in meters
%v	vertical speed in m/s
%t	timestamp
%l	coordinates
%c	pedal cadence
%h	heart rate
%g	road gradient
\n	newline

Default format (used when option isn't specified) is "%s km/h %e m\n%t %l". Suggested format for bicycle video is "'%s km/h %h ❤\n %e m %c rpm".

Tab delimited fields useful for OpenOffice (openoffice)

This format can...

- read and write waypoints

This format is derived from the xcsv format, so it has all of the same options as that format.

Tab separated export-all (except geocaching data) file format. Intended to serve as source for number-processing applications like OpenOffice, Ploticus and others. Tab was chosen as delimiter because it is a) supported by both OpenOffice and Ploticus and b) is not ',', so you can use

```
sed -i "s/./,/g" <x>.csv'
```

to adapt it to locales where ',' is used as decimal separator. Contributed by Tobias Minich.

Textual Output (text)

This format can...

- write waypoints

This format has the following options: nosep, encrypt, logs, degformat, altunits, splitoutput .

This is a simple human readable version of the data file, handy for listings of any type of waypoint files.

The following command line reads a GPX file with Groundspeak extensions and writes a text file with encrypted hints:

```
gpsbabel -i gpx -f 12345.gpx -o text,encrypt -F 12345.txt
```

nosep option

Suppress separator lines between waypoints.

To suppress the dashed lines between waypoints, use this option.

`encrypt` option

Encrypt hints using ROT13.

Use this option to encrypt hints from Groundspeak GPX files.

`logs` option

Include groundspeak logs if present.

Use this option to include Groundspeak cache logs in the created document.

`degformat` option

Degrees output as 'ddd', 'dmm'(default) or 'dms'.

When GPSBabel writes coordinates, this option is consulted to see if it should write decimal degrees ('ddd') decimal minutes ('dmm') or degrees, minutes, seconds ('dms'). The default is 'dmm'.

`altunits` option

Units for altitude (f)eet or (m)etres.

This option should be 'f' if you want the altitude expressed in feet and 'm' for meters. The default is 'f'.

`splitoutput` option

Write each waypoint in a separate file.

Splits output into separate files for each waypoint by appending a decimal number to the output filename.

Example 3.38. Example for splitoutput option to text format

If "MyPQ.gpx" contains five waypoints,

```
gpsbabel -i gpx -f MyPocketQuery -o text,split -F blah
```

will result in files named blah1 ... blah5, each containing info from one of those waypoints.

Universal csv with field structure in first line (unicsv)

This format can...

- read and write waypoints
- read and write tracks
- read and write routes

This format has the following options: datum, grid, utc, format, filename, fields, codec .

Unicsv examines the first line of a file to determine the field order and field separator in that file. On write, it tries to figure out what data it has and writes headers and all the data it can.

Fields may be enclosed in double quotes. To include a double quote inside quotes escape it with another double quote.

If the first line contains any unenclosed tabs then the data lines are assumed to be tab separated. Otherwise if the first line contains any unenclosed semicolons then fields are assumed to be separated by semicolons. Otherwise if the first line contains any unenclosed vertical bars then fields are assumed to be separated by vertical bars. Otherwise the fields are assumed to be separated by commas.

The list of keywords include:

```
     alt =      Elevation (in meters).  For feet use "alt ft",
"altft", "alt feet", or "altfeet".
     arch =     Geocache archived flag
     avail =    Geocache available flag
     bng_e =    British National Grid's easting
     bng =      full coordinate in BNG format (zone easting northing)
     bng_pos =  full coordinate in BNG format (zone easting northing)
     bng_n =    British National Grid's northing
     bng_z =    British National Grid's zone
     caden =    Cadence
     comment =  Notes
     cont =     Geocache container
     cour =     Heading / Course true
     date =     Date (yyyy/mm/dd)
     depth =    Depth (in meters).  For feet use "depth ft",
"depthft", "depth feet", or "depthfeet".
     desc =     Description
     diff =     Geocache difficulty
     ele =      Elevation (in meters).  For feet use "ele ft",
"eleft", "ele feet", or "elefeet".
     e/w =      'e' for eastern hemisphere, 'w' for western
     found =    Geocache last found date
     fix =      3d, 2d, etc.
     gcid =     Geocache cache id. This accepts GC-ID ("575006") and
GC-Code ("GC1234G").
     geschw =   Geschwindigkeit (speed)
     hdop =     Horizontal dilution of precision
     head =     Heading / Course true
     heart =    Heartrate
     height =   Elevation (in meters).  For feet use "height ft",
"heightft", "height feet", or "heightfeet".
     hint =     Geocache cache hint
     icon =     Symbol (icon) name
     lat =      Latitude
     lon =      Longitude
     name =     Waypoint name ("Shortname")
     n/s =      'n' for northern hemisphere, 's' for southern
     notes =    Notes
     pdop =     Position dilution of precision
     placer =   Geocache placer
     placer_id =Geocache placer id
```

```
     power =    Cycling power (in Watts)
     prox =     Proximity (in meters).  For feet use "prox ft",
"proxft", "prox feet", or "proxfeet".
     sat =      Number of sats used for fix
     speed =    Speed, in meters per second. (See below)
     symb =     Symbol (icon) name
     tempf =    Temperature (degrees Fahrenheit)
     temp =     Temperature (degrees Celsius)
     terr =     Geocache terrain
     time =     Time (hh:mm:ss[.msec])
     type =     Geocache cache type
     url =      URL
     utc_d =    UTC date
     utc_t =    UTC time
     utm_c =    UTM zone character
     utm_e =    UTM easting
     utm =      full coordinate in UTM format (zone zone-ch easting
northing)
     utm_pos =  full coordinate in UTM format (zone zone-ch easting
northing)
     utm_n =    UTM northing
     utm_z =    UTM zone
     vdop =     Vertical dilution of precision
     x =        Longitude
     x_pos =    Longitude
     y =        Latitude
     y_pos =    Latitude
     z =        Elevation (in meters)
```

We support some enhanced Garmin attributes. They are also available in gpx, gdb, garmin_gpi and partly garmin_txt. These entities are currently not visible in MapSource™ (6.12.4), but are NOT dropped when working with GDB (version 3) or GPX files.

Please note, that these do NOT provide a geocoding service; don't expect to "convert" a street address to a latitude and longitude.

```
     addr =     Street address
     city =     City
     country =  Country
     faci =     Facility (not available in GPX)
     phone =    Phone number
     post =     Postal code
     state =    State
```

Fuller spellings (i.e. "longitude") may be used. You can also use keywords with a whitespace instead of an underscore.

A typical file may be:

```
    Name, Latitude, Longitude, Description
```

```
GCEBB,35.972033,-87.134700,Mountain Bike Heaven by susy1313
GC1A37,36.090683,-86.679550,The Troll by a182pilot & Family
```

If processing data from the UK, GPSBabel can process coordinates using X,Y values (often referred to as Eastings/Northings) as shown in Example 3.39, “CSV input for UK data with XY coordinates” or the full 12 figure alpha numeric, as shown in Example 3.40, “CSV input for UK data with alphanumeric coordinates”. Note in Example 3.40, “CSV input for UK data with alphanumeric coordinates” you need to split your original X,Y values into the 100Km 2 character code, eastings and northing values.

Example 3.39. CSV input for UK data with XY coordinates

```
bng_e,bng_n,name,date
353729,177210,id_001,2018/02/03
356025,181221,id_002,2018/02/03
357962,181528,id_003,2018/03/03
```

Example 3.40. CSV input for UK data with alphanumeric coordinates

```
bng_z,bng_e,bng_n,name,date
ST,53729,77210,id_001,2018/02/03
ST,56025,81221,id_002,2018/02/03
ST,57962,81528,id_003,2018/03/03
```

On the output side unicsv writes fixed number of columns (waypoint index, latitude and longitude) followed by a variable column list depending on internal data.

With at least ONE valid timestamp in data a unicsv output may look like that:

```
    No,Name,Latitude,Longitude,Description,Date,Time
    1,"GCEBB",35.972033,-87.134700,"Mountain Bike Heaven by
susy1313",2003/06/29,09:00:00
    2,"GC1A37",36.090683,-86.679550,"The Troll by a182pilot &
Family",,
```

For speed, a units specifier can be added to override the default. Here are some values, but check parse_speed() in parse.cc for the authoritative list.

- m/s, mps: meters per second
- km/h, kmh: kilometers per hour
- kt, knots: knots
- mph, mi/h, mih: miles per hour

`datum` option

GPS datum (def. WGS 84).

This option specifies the datum to be used on output. Valid values for this option are listed in Appendix A, *Supported Datums*.

`grid` option

Write position using this grid..

This value specifies the grid to be used on write. It is similar to the grid option of garmin_txt (see Table 3.1, “Grid values for garmin_txt”). The only difference is that unicsv does not write a degree sign (°) into the output file.

Without this option unicsv writes the coordinates as simple numbers like in the samples above.

`utc` option

Write timestamps with offset x to UTC time.

This option specifies the local time zone to use when reading and writing times. It specifies a Universal Coordinated Time (UTC) offset in hours. For example, in the winter in Sweden the UTC offset is UTC+1, which corresponds to an option utc=1. Valid values are from -14 to +14.

`format` option

Write name(s) of format(s) from input session(s).

When this option is enabled, we generate an additional 'Format' column. The values of this column are filled with names of previous input formats.

Example 3.41. Example for unicsv format option to write names of input formats.

The next example ...

```
gpsbabel -i gpx -f file1.gpx -i gdb -f file2.gdb -o unicsv,format=y -
F result.txt
```

... could produce following output:

```
No,Latitude,Longitude,Name,Description,Symbol,Date,Time,Format
1,51.075139,12.463689,"578","578","Waypoint",2005/04/26,16:27:23,"gdb"
2,51.081104,12.465277,"579","579","Waypoint",2005/04/26,16:27:23,"gdb"
3,50.844126,12.408757,"Gosel","Gosel","Exit",2005/02/26,10:10:47,"gpx"
4,50.654763,12.204957,"Greiz",,"Exit",2005/02/26,09:57:04,"gpx"
```

`filename` option

Write filename(s) from input session(s).

When this option is enabled, we write an additional column called 'Filename'. The values of this column are filled with filenames of previous input formats.

This can be very helpful for locating specific waypoints (i.e. using the position filter) in more than one file.

Example 3.42. Example for unicsv filename option to write filenames of input formats.

The next example ...

```
gpsbabel -i gpx -f file1.gpx -i gdb -f file2.gdb -o unicsv,filename=1
-F result.txt
```

... could produce following output:

```
No,Latitude,Longitude,Name,Date,Time,Filename
1,51.075139,12.463689,"578",2005/04/26,16:27:23,"reference/gdb-sample.gdb"
2,51.081104,12.465277,"579",2005/04/26,16:27:23,"reference/gdb-sample.gdb"
3,50.844126,12.408757,"580",2005/02/26,10:10:47,"reference/gdb-sample.gpx"
4,50.654763,12.204957,"581",2005/02/26,09:57:04,"reference/gdb-sample.gpx"
```

`fields` option

Name and order of input fields, separated by '+'.

This option lets you specify the field names of your input file from the command line instead of relying on the first line of your input file describing the file. Field names are separated by a '+' character. The list of field names is exactly that allowed in the first line of a unicsv file without this option.

Example 3.43. Example for unicsv fields option to describe input file.

For example ...

```
gpsbabel -i unicsv,fields=lat+lon+description -f file.csv -o gpx -F
file.gpx
```

declares that file.csv has three fields, latitude, longitude, and description, in that order.

codec option

codec to use for reading and writing strings (default UTF-8).

This lets you override the default codec of 'UTF-8'. As an input option the codec should correspond to the encoding of the input file. As an output option it sets the encoding of the output file.

Vcard Output (for iPod) (vcard)

This format can...

- write waypoints

This format has the following options: encrypt .

The vCard output is intended to be in a format that enables waypoints to be viewed with an Apple iPod. This is achieved by mapping waypoint fields into vCard fields that can be displayed as 'Contacts' on the iPod. With the iPod mounted as a hard disk (see your iPod manual for instructions), the resulting VCF file should be moved into the iPod 'Contacts' folder. As an alternative, Mac OS X users may prefer to drag the VCF file into their address book and synchronize with the iPod using iSync.

`encrypt` option

Encrypt hints using ROT13.

By default geocaching hints are unencrypted; use this option to encrypt them.

Chapter 4. Data Filters

GPSBabel supports data filtering. Data filters are invoked from the command line via the '-x' option. It should be noted that data filters are invoked in the internal pipeline at the point that corresponds to their position on the command. This implies that specifying a filter before reading any data ('-x <filter> -f <file>'), despite being legal, will not have any effect. The advantage is that filters can be used intermittently between several variations of input and output functions. It should also be noted that filtering data from different input types can sometimes produce undesirable results due to differences in the native data formats.

Beware that most filters only apply to a certain kind of data. This is usually indicated below by referring to points, tracks or routes in the first sentence which describes each filter or in the table at gpsbabel.org [https://www.gpsbabel.org/capabilities.html] .

Add points before and after bends in routes (bend)

The bend filter modifies each route replacing each point inside a curve with two points: one at a given distance in the direction of the previous point, and another at the same distance in the direction of the next point in the route. It only replaces points where there is a change in heading big enough.

When creating a route, points are usually created inside curves or intersections. That means that, while navigating that route using a GPS unit, the course pointer would aim to the inside of that curve or intersection, and only when you have passed that point will the GPS aim to the next waypoint in the route. This behavior is useful in marine navigation but when biking, for instance, it may be a bit late to decide where to turn to in an intersection.

This filter tries to solve that creating a waypoint before and after where there is a change in direction.That way, the course pointer will point to the direction you should turn to ahead in time.

For this filter to work correctly, the route should be simple enough that there is only one waypoint inside each curve or intersection. Because of that, it is usually a good idea to use the simplify filter before this one.

This command line reads route.gpx and replaces each point with other two points: one 25 meters before and another 25 meters after the original point. It replaces a point only if there is a change of direction larger than 5 degrees.

```
gpsbabel -i gpx -f route.gpx -x bend,distance=25,interpolate,minangle=5
-o gpx -F newroute.gpx
```

distance option

Distance to the bend in meters where the new points will be added.

Distance in meters to the original point where the new points will be added.

The new points will be created at this distance. The first one in the direction of the previous point, ant the second one in the direction of the next point in the route.

minangle option

Minimum bend angle in degrees.

Minimum curve angle in degrees.

The substitution will only be made if the change in the heading is greater than this value. This avoids replacing a point if the GPS unit is already pointing in the correct direction, or if the route reaches a certain point and goes back the same road.

Include Only Points Inside Polygon (polygon)

The polygon filter includes points if they are inside of a polygon. A polygon file looks like an arc file, except that the arc it describes must be a closed cycle. That is, for a simple polygon, the first and last points must be the same. Here's a square:

```
# A square (not really) polygon
41.0000       -85.0000
41.0000       -86.0000
42.0000       -86.0000
42.0000       -85.0000
41.0000       -85.0000
```

Polygons may include islands and holes. To include an island or a hole, just append it to the main polygon.

```
# A square polygon with a triangular hole
41.0000       -85.0000
41.0000       -86.0000
42.0000       -86.0000
42.0000       -85.0000
41.0000       -85.0000
# The hole begins here
41.5000       -85.5000
41.6000       -85.5000
41.6000       -85.6000
41.5000       -85.5000
```

As with the arc filter, you define a polygon by giving the name of the file that contains it, using the `file` option.

Note that this filter currently will not work properly if your polygon contains one or both poles or if it spans the line of 180 degrees east or west longitude.

Example 4.1. Using the polygon filter

Suppose you have a polygon file that defines the border of your county, called mycounty.txt. This command line will give you only the points in your county:

```
gpsbabel -i geo -f 1.loc -x polygon,file=mycounty.txt -o mapsend -F 2.wpt
```

Example 4.2. Using the polygon and arc filters to find points in or nearly in a polygon

Because the polygon and arc filters use the same file format, you can use them together to find all points that are "in or nearly in" a polygon. This can be useful if your waypoints or the boundaries of your polygon

are not quite perfect, so you want to provide a buffer zone around it in case there are points nearby that should be in the polygon but aren't quite.

```
gpsbabel -i gpx -f points.gpx -x stack,push -x polygon,file=mycounty.txt
-x stack,swap -x arc,file=mycounty.txt,distance=1k -x stack,pop,append
-x duplicate,shortname -o gpx -F nearmycounty.gpx
```

This command makes a copy of the points, finds the ones that are in your your county, swaps that result with the copy of the original set of points, finds the ones from that set that are within 1 km of the border of the county, puts the two lists together, and then filters out any points that appear twice (This step is necessary because points inside the county but near the county line will be kept by both the polygon and the arc filter.)

file option

File containing vertices of polygon.

This option is required.

This option specifies the name of the file containing the polygon to use for filtering. The format of the file is as described above.

GPSBabel supports converting any route or track to a file usable by this filter; simply read it in the normal way and write it using the arc file format. Afterward, you will need to make sure that the first point and the last point in the file are the same, as the polygon filter depends on that. You can do so with any text editor.

exclude option

Exclude points inside the polygon.

When this option is specified, the usual sense of the polygon filter is reversed. That is, points that are inside the polygon are discarded while points that are further away are kept.

Include Only Points Within Distance of Arc (arc)

This filter keeps or removes waypoints based on their proximity to an arc, which is a series of connected line segments similar to a route or a track but without any associated data other than the coordinates. Optionally, it can move each non-deleted waypoint over the closest segment of the arc.

The arc may defined in a file whose name must be provided with the `file`, or the tracks or routes that have already been read. That file contains pairs of coordinates for the vertices of the arc, one coordinate pair per line. Comments may be included by preceding them with a '#' character. An arc file looks something like this sample:

```
# Lima Road/SR3 north of Fort Wayne, Indiana
41.150064468    -85.166207433
41.150064468    -85.165371895
41.149034500    -85.165157318
41.147832870    -85.164771080
41.146631241    -85.164384842
```

```
41.144270897    -85.163655281
41.141953468    -85.162882805
```

An arc file may optionally contain gaps in the arc. You may specify such a gap by inserting a line containing "#break" either on a line by itself or after the coordinates of the starting point of the new arc segment.

Example 4.3. Using the arc filter

Assuming the arc above is in a file called `lima_rd.txt`, the following command line would include only points within one mile of the section of Lima Road covered by the arc. The output is reordered by distance to the arc, closest points first.

```
gpsbabel -i geo -f 1.loc -x arc,file=lima_rd.txt,distance=1 -o mapsend
-F 2.wpt
```

file option

File containing vertices of arc.

This option specifies the name of the file containing the arc to use for filtering. The format of the file is as described above.

GPSBabel supports converting any route or track to a file usable by this filter; simply read it in the normal way and write it using the arc file format.

rte option

Route(s) are vertices of arc.

When this option is specified the routes contains the vertices of the arc. If there are several routes then it is assumed that there is a gap between each of them.

trk option

Track(s) are vertices of arc.

When this option is specified the tracks contains the vertices of the arc. If there are several tracks then it is assumed that there is a gap between each of them.

distance option

Maximum distance from arc.

This option is not required, but if it is not specified the distance defaults to zero miles, which isn't very useful.

This option specifies the maximum distance a point may be from the arc without being discarded. Points that are closer to the arc are kept, while points that are further away are discarded.

The units may be specified by appending a suffix to the supplied number:

'm' for meters, e.g. 3500.0m

'ft' or 'feet' for feet, e.g. 11483ft
'k' or 'km' for kilometers, e.g 3.5000km
'nm' for nautical miles, e.g. 1.8898nm
'mi' for miles, e.g. 2.1748mi
'fa' for fathoms, e.g. 1913.82fa

If no units are specified, the units are assumed to be miles.

exclude option

Exclude points close to the arc.

When this option is specified, the usual sense of the arc filter is reversed. That is, points that are closer than `distance` are discarded while points that are further away are kept.

points option

Use distance from vertices not lines.

When this option is specified, only points that are within the specified distance of one of the vertices of the arc are kept. This differs from the normal mode of operation in that in the normal mode, points that are close to the lines between points are also kept.

This option makes the arc filter act like a multi-point version of the radius filter.

project option

Move waypoints to its projection on lines or vertices.

When this option is specified, each non deleted waypoint is moved over the closest segment, or over the nearest point if `points` option is used.

With either the `rte` or `trk` option the altitude and creation_time of the projected waypoints are updated, if possible, by interpolation from the route or track data.

This is most useful if you are trying to obtain the closest points in a road to some places. Or if you want to know the step times on some places over the tracks. Also to transform waypoints in Garmin course points (see gtrnctr and garmin format).

Include Only Points Within Radius (radius)

This filter includes or excludes waypoints based on their proximity to a central point. All waypoints more than the specified distance from the specified point will be removed from the dataset.

By default, all remaining points are sorted so that points closer to the center appear earlier in the output file.

Example 4.4. Using the radius filter to find points close to a given point

This example command line would include only points within 1 1/2 miles of N30.000 W 90.000

```
gpsbabel -i geo -f 1.loc -x radius,distance=1.5M,lat=30.0,lon=-90.0 -
o mapsend -F 2.wpt
```

lat option

Latitude for center point (D.DDDDD).

This option is required.

This option specifies the latitude of the central point in decimal degrees. South latitudes should be expressed as a negative number. Valid values for this option are from -90 to 90.

lon option

Longitude for center point (D.DDDDD).

This option is required.

This option specifies the longitude of the central point in decimal degrees. West longitudes should be expressed as a negative number. Valid values for this option are from -180 to 180.

distance option

Maximum distance from center.

This option is required.

This option specifies the maximum distance a point may be from the central point in order to remain in the dataset. Points closer than this distance will be kept and points further away will be removed (unless the `exclude` option is specified.)

The units may be specified by appending a suffix to the supplied number:

'm' for meters, e.g. 3500.0m
'ft' or 'feet' for feet, e.g. 11483ft
'k' or 'km' for kilometers, e.g 3.5000km
'nm' for nautical miles, e.g. 1.8898nm
'mi' for miles, e.g. 2.1748mi
'fa' for fathoms, e.g. 1913.82fa

If no units are specified, the units are assumed to be miles.

exclude option

Exclude points close to center.

If this option is included, the action of the radius filter will be reversed: points within the given distance will be removed, and points further away will be kept.

nosort option

Inhibit sort by distance to center.

If this option is specified, the radius filter will not sort the remaining points by distance from the center. They will remain in whatever order they were originally.

maxcount option

Output no more than this number of points.

This option specifies the maximum number of points that the radius filter may keep. If there are more than this number of points within the specified distance of the center, the more distant points will be discarded even though they are within the specified distance. If this option is not specified, all points are kept regardless of how many there are.

Note that if the `nosort` option is also specified, this option will instead keep points based on their position within the input file rather than on their distance from the center. This may or may not be what you want.

Note, too, that this option may be used with the `exclude` option, but the results might not be what you expect. In particular, the results will not be the same as if you had kept all of the points you'd otherwise throw away. You will still get no more than `maxcount` points, but they will all be at least `distance` away from the center. (And possibly sorted.)

asroute option

Put resulting waypoints in route of this name.

This option specifies the name of a route. If this option is specified, the radius filter puts all points that are kept into a route with the given name. The order of points in the route is by distance from the center (unless the `nosort` option is also specified.)

Note that this route is not necessarily the most efficient route to visit all of the points. In fact, for some data sets, it might be the least efficient route.

Interpolate between trackpoints (interpolate)

This filter modifies any tracks so that either the distance or the time between consecutive points is no more than the specified interval. Where points are missing, the filter fills them in by following a straight line (actually a great circle) between the adjacent points. You must specify either the `distance` or the `time` option.

Example 4.5. Using the interpolate filter

This command line reads track.gpx and inserts points wherever two adjacent trackpoints are more than 10 seconds apart:

```
gpsbabel -i gpx -f track.gpx -x interpolate,time=10 -o gpx -F new-
track.gpx
```

This command reads track.gpx and inserts points wherever two adjacent trackpoints are more than 15 kilometers apart:

```
gpsbabel -i gpx -f track.gpx -x interpolate,distance=15k -o gpx -F
newtrack.gpx
```

This command reads track.gpx and inserts points wherever two adjacent trackpoints are more than 2 miles apart:

```
gpsbabel -i gpx -f track.gpx -x interpolate,distance=2m -o gpx -F new-
track.gpx
```

time option

Time interval in seconds.

This option specifies the maximum allowable time interval between adjacent points in the track. If two points in the track are further apart than this value, new points will be inserted between them.

This value is always specified in units of seconds. Examples: 31, 1.5.

Either this option or the `distance` must be specified.

distance option

Distance interval.

This option specifies the maximum allowable distance between adjacent points in the track. If two points in the track are further apart than this value, new points will be inserted between them.

The units may be specified by appending a suffix to the supplied number:

'm' for meters, e.g. 3500.0m
'ft' or 'feet' for feet, e.g. 11483ft
'k' or 'km' for kilometers, e.g 3.5000km
'nm' for nautical miles, e.g. 1.8898nm
'mi' for miles, e.g. 2.1748mi
'fa' for fathoms, e.g. 1913.82fa

If no units are specified, the units are assumed to be miles.

Either this option or the `time` must be specified.

route option

Interpolate routes instead.

If this option is specified, the interpolate filter interpolates routes rather than tracks. Because route points do not have time stamps, it is an error to use this option with the `time` option.

Manipulate altitudes (height)

The height filter allows the correction of altitude values. At least one popular gps logger does store the ellipsoidal height (sum of the height above mean see level and the height of the geoid above the WGS84 ellipsoid) instead of the height above sea level, as it can be found on maps. The height filter allows for the correction of these altitude values. This filter supports two options: `wgs84tomsl` and `add`. At least one of these options is required, both can be combined.

Example 4.6. This option subtracts the WGS84 geoid height from every altitude. For GPS receivers like the iBlue747 the result is the height above mean see level.

```
gpsbabel -i gpx -f in.gpx -x height,wgs84tomsl -o gpx -F out.gpx
```

The coordinates and altitude vales must be based an the WGS84 ellipsoid for this option to produce sensible results

Example 4.7. This options adds a constant value to every altitude.

```
gpsbabel -i gpx -f in.gpx -x height,add=10.2f -o gpx -F out.gpx
```

You can specify negative numbers to subtract the value. If no unit is specified meters are assumed. For feet you can attach an "f" to the value.

add option

Adds a constant value to every altitude.

Adds a constant value to every altitude. You can specify negative numbers to subtract the value.

The units may be specified by appending a suffix to the supplied number:

'm' for meters, e.g. 3.5m
'ft' or 'feet' for feet, e.g. 11.483ft
'k' or 'km' for kilometers, e.g 0.0035km
'nm' for nautical miles, e.g. 0.0018898nm
'mi' for miles, e.g. 0.0021748mi
'fa' for fathoms, e.g. 1.9138fa

If no units are specified, the units are assumed to be meters.

wgs84tomsl option

Converts WGS84 ellipsoidal height to orthometric height (MSL).

Subtracts the WGS84 geoid height from every altitude.

For GPS receivers like the iBlue747 this corrects the logged altitudes to height above mean sea level.

Manipulate track lists (track)

WARNING: This filter always drops empty tracks.

This filter performs various operations on track data.

move option

Correct trackpoint timestamps by a delta.

This option changes the time of all trackpoints. This might be useful if your track must be moved by one or more hours because of an incorrect time zone. It can also be useful to correct tracks for week number roll over problems.

The value of this option is a series of integer and unit pairs. Each integer may include a leading '+' or '-' sign. Positive integers shift the tracks later, while negative integers shift the tracks earlier. If no sign is provided the integer is assumed to be nonnegative. Possible units are w for weeks, d for days, h for hours, m for minutes, s for seconds and z for milliseconds.

Example 4.8. Time-shifting a track with the track filter

The following command line will shift all tracks to be one hour later.

```
gpsbabel -t -i gpx -f in.gpx -x track,move=+1h -o gpx -F out.gpx
```

Example 4.9. Time-shifting a track with the track filter to correct WNRO

The following command line will shift all tracks to be 1024 weeks later. Because the GPS Week Number is transmitted modulo 1024 there is the possibility that the recovered timestamp will be off by a multiple of 1024 weeks.

```
gpsbabel -t -i gpx -f in.gpx -x track,move=+1024w -o gpx -F out.gpx
```

Example 4.10. Time-shifting a track with the track filter with combined units

The following command lines will each shift all tracks to be 1 hour and 1 minute earlier, i.e. 61 minutes earlier.

```
gpsbabel -t -i gpx -f in.gpx -x track,move=-1h-1m -o gpx -F out.gpx
```

```
gpsbabel -t -i gpx -f in.gpx -x track,move=-61m -o gpx -F out.gpx
```

pack option

Pack all tracks into one.

This option causes all tracks to be appended to one another to form a single track. This option does not work if any two tracks overlap in time; in that case, consider using the `merge` option.

This option is most useful for rejoining tracks that might have been interrupted by an equipment malfunction or an overnight stop.

If no other option is given to the track filter, this option is assumed.

split option

Split by date or time interval.

The input track will be split into several tracks depending on date of track points. If there is more than one track, use the pack option before before using this. To split a single tracks into separate tracks for each day and name them, use this:

```
gpsbabel -t -i gpx -f in.gpx -x track,split,title="ACTIVE LOG # %Y%m
%d" -o gpx -F out.gpx
```

If the input has multiple tracks, pack them together before splitting them back apart per day thusly:

```
gpsbabel -t -i gpx -f in.gpx -x track,pack,split,title="ACTIVE LOG #
%D" -o gpx -F out.gpx
```

Additionally you can add an interval to the split option. With this the track will be split if the time between two points is greater than this parameter. The interval must be numeric and can be int days, hours, minutes or seconds, expressed as one of the character "d", "h", "m", or "s".

For example, to split a track based on an four hour interval, use this:

```
gpsbabel -t -i gpx -f in.gpx -x track,pack,split=4h,title="LOG # %c"
-o gpx -F out.gpx
```

sdistance option

Split by distance.

The input track will be split into several tracks if the distance between successive track points is greater than the distance given as a parameter. The distance must be numeric and can be in miles or kilometers, expressed as one of the character "k", or "m". If sdistance is given no parameters, this option has the same effect as the split option without parameters. If there is more than one track, use the pack option before before using this.

For example, to split the track if the distance between points is greater than 100 meters, use this:

```
gpsbabel -t -i gpx -f in.gpx -x track,pack,sdistance=0.1k -o gpx -F
out.gpx
```

The sdistance option can be combined with the split option. The track then will be split only if both time and distance interval exceeds the supplied values. This technique can be used to filter out gaps from the tracklog. The gap is kept only if the gps device is without signal for longer time than that given and during that time it moves a distance over that given. This example splits the track if the device is without signal for at least 5 minutes and during this time moves more than 300 meters:

```
gpsbabel -t -i gpx -f in.gpx -x track,pack,sdistance=0.3k,split=5m -
o gpx -F out.gpx
```

merge option

Merge multiple tracks for the same way.

This option puts all track points from all tracks into a single track and sorts them by time stamp. Redundant points with identical time stamps will be dropped.

Example 4.11. Merging tracks with the track filter

Suppose you want to merge tracks recorded with two different GPS devices at the same time. To do that, use this command line:

```
gpsbabel -t -i gpx -f john.gpx -i gpx -f doe.gpx -x track,merge,ti-
tle="COMBINED LOG" -o gpx -F john_doe.gpx
```

name option

Use only track(s) where title matches given name.

With the name option you can filter out a track by title.

The comparison is always non-case-sensitive. Wildcards are allowed.

start option

Use only track points after or at this timestamp.

This option is used along with the `stop` to discard trackpoints that were recorded outside of a specific period of time. This option specifies the beginning of the time period.

If this option is not specified, the time period is assumed to begin at the dawn of time or January 1, 1970, whichever was later. The time for this option is expressed in UTC.

The value of this option must be in the form of YYYYMMDDHHMMSS.SSS, but it is not necessary to specify the smaller time units if they are not needed. That is, if you only care about points logged between 10 AM and 6 PM on a given date, you need not specify the minutes or seconds.

Example 4.12. Extracting a period of time with the track filter

To get only the parts of a track that were mapped on 20 July 2005 between 10 AM and 6 PM, use this command line:

```
gpsbabel -t -i gpx -f in.gpx -x track,start=2005072010,stop=2005072018
-o gpx -F out.gpx
```

stop option

Use only track points before or at this timestamp.

This option is used in conjunction with the `start` option to discard all trackpoints outside of a given period of time. This option defines the end of the time period.

If this option is not specified, the time period is assumed to end at the end of civilization as we know it or the year 2038, whichever comes first. The time for this option is expressed in UTC.

See the `start` option for the format of this value and an example of usage.

title option

Basic title for new track(s).

This option specifies a title for tracks generated by the track filter. By default, the title of the new track is composed of the start time of the track appended to this value.

If this value contains a percent (%) character, it is treated as a format string for the POSIX strftime function, allowing custom time-based track names.

fix option

Synthesize GPS fixes (PPS, DGPS, 3D, 2D, NONE).

This option sets the GPS fix status for all trackpoints to the specified value. Valid values for this option are PPS, DGPS, 3D, 2D, or NONE.

This option is most useful when converting from a format that doesn't contain GPS fix status to one that requires it.

course option

Synthesize course.

This option computes (or recomputes) a value for the GPS heading at each trackpoint. This is most useful with trackpoints from formats that don't support heading information or for trackpoints synthesized by the interpolate filter. The heading at each trackpoint is simply the course from the previous trackpoint in the track. The first trackpoint in each track is arbitrarily assigned a heading of 0 degrees.

speed option

Synthesize speed.

This option computes a value for the GPS speed at each trackpoint. This is most useful with trackpoints from formats that don't support speed information or for trackpoints synthesized by the interpolate filter.

The speed at each trackpoint is the average speed from the previous trackpoint (distance divided by time). The first trackpoint in each track is assigned a speed of "unknown."

The unit of speed is meters per second.

seg2trk option

Split track at segment boundaries into multiple tracks.

This option splits tracks at segment boundaries into multiple tracks. This is useful to restore the behavior of GPSBabel versions up to 1.3.6 which didn't support track segment markers and automatically put each segment into a separate track.

trk2seg option

Merge tracks inserting segment separators at boundaries.

This option merges multiple tracks, inserting segment separators at track boundaries. It expects the tracks to already be in the right order for merging, i.e. it does not check timestamps and reorder track points so that their timestamps are monotonically increasing.

segment option

segment tracks with abnormally long gaps.

faketime option

Add specified timestamp to each trackpoint.

This option assigns a time value to each trackpoint.

The value of this option must be in the form of fYYYYMMDDHHMMSS+SS.

The parameter f (force) is optional and means that the time value of each trackpoint is replaced. If f is not specified, the time value of each trackpoint is only replaced when the trackpoint contains no time value.

YYYYMMDDHHMMSS is the pattern for the timestamp and is required.

The plus sign is the delimiter between the timestamp and the step time in seconds. The first trackpoint receives the time value of the timestamp and each following trackpoint receives the timestamp incremented by the step time. The specification of the steptime is optional.

The parameter was added because some software products (e.g. garmin training center) require a time value for each trackpoint.

Example 4.13. Replace time values of a track

Replace all time values with new time values. Start at the 5 th of July, 2010 at 8 PM and increment 2 seconds between each trackpoint:

```
gpsbabel -i kml -f in.kml -x track,faketime=f20100705200000+2 -o gtrnctr
-F out.tcx
```

Example 4.14. Add time values to a track

Add a time value to a trackpoint, if the trackpoint contains no time value. Start at the 6 th of May, 2010 at 6 AM and increment 5 seconds between each trackpoint:

```
gpsbabel -i kml -f in.kml -x track,faketime=20100506060000+5 -o gtrnctr
-F out.tcx
```

discard option

Discard track points without timestamps during merge.

This option is used in conjunction with the merge option to discard track points with missing timestamps instead of aborting with the "Found track point at lat,lon without time!" error.

Example 4.15. Merging tracks with missing timestamps with the track filter

Suppose you want to merge tracks that may have missing timestamps. To do that, use this command line:

```
gpsbabel -t -i gpx -f john.gpx -f doe.gpx -x track,merge,discard -o
gpx -F john_doe.gpx
```

minimum_points option

Discard tracks with fewer than these points.

Eliminates any remaining tracks with fewer than this number of trackpoints.

This step is performed last by this filter and is used to clean up earlier simplifications that may have left tracks with so few points as to be useless, such as a track taken while stationary but with GPS wander.

Rearrange waypoints, routes and/or tracks by resorting (sort)

This filter sorts waypoints, routes and/or tracks by the selected field(s).

description option

Sort waypoints by description.

This option causes the waypoints to be sorted in alphabetical order by description.

This option is not valid in combination with gcid, shortname, and time.

gcid option

Sort waypoints by numeric geocache ID.

If the data contains Groundspeak geocache IDs, this option causes the waypoints to be sorted in numerical order by geocache ID.

This option is not valid in combination with description, shortname, and time.

shortname option

Sort waypoints by short name.

This option causes the waypoints to be sorted in alphabetical order by short name.

This option is not valid in combination with description, gcid, and time.

time option

Sort waypoints by time.

This option causes the waypoints to be sorted in chronological order by creation time.

This option is not valid in combination with description, gcid, and shortname.

rtedesc option

Sort routes by description.

This option causes the routes to be sorted in alphabetical order by description.

This option is not valid in combination with rtename and rtenum.

rtename option

Sort routes by name.

This option causes the routes to be sorted in alphabetical order by name.

This option is not valid in combination with rtedesc and rtenum.

rtenum option

Sort routes by number.

This option causes the routes to be sorted in numerical order by number.

This option is not valid in combination with rtedesc and rtename.

trkdesc option

Sort tracks by description.

This option causes the tracks to be sorted in alphabetical order by description.

This option is not valid in combination with trkname and trknum.

trkname option

Sort tracks by name.

This option causes the tracks to be sorted in alphabetical order by name.

This option is not valid in combination with trkdesc and trknum.

trknum option

Sort tracks by number.

This option causes the tracks to be sorted in numerical order by number.

This option is not valid in combination with trkdesc and trkname.

Remove all waypoints, tracks, or routes (nuketypes)

There are three main types of data that GPSBabel deals with: waypoints, tracks, and routes. The nuketypes filter allows removing all the data of any or all of those three types.

Example 4.16. Filtering data types with nuketypes

If you have a GPX file that contains routes, tracks, and waypoints and you want a GPX file that contains only tracks, you may use this filter to remove the waypoints and the routes with this command:

```
gpsbabel -i gpx -f bigfile.gpx -x nuketypes,waypoints,routes -o gpx -
F tracksonly.gpx
```

waypoints option

Remove all waypoints from data stream.

This option causes the nuketypes filter to discard all waypoints that are not associated with a track or route.

tracks option

Remove all tracks from data stream.

This option causes the nuketypes filter to discard all track data.

routes option

Remove all routes from data stream.

This option causes the nuketypes filter to discard all route data.

Remove Duplicates (duplicate)

The duplicate filter is designed to remove duplicate points based on their short name (traditionally a waypoint's name on the GPS receiver), and/or their location (to a precision of 6 decimals). This filter supports two options that specify how duplicates will be recognized, `shortname` and `location`. At least one of these options is required.

Example 4.17. Using the duplicate filter to suppress points with the same name and location

This command line removes points that have duplicate short names and duplicate locations. The result would be a gpx file that more than likely contains only unique points and point data.

```
gpsbabel -i gpx -f 1.gpx -f 2.gpx -x duplicate,location,shortname -o
gpx -F merged_with_no_dupes.gpx
```

shortname option

Suppress duplicate waypoints based on name.

This option is the one most often used with the duplicate filter. This option instructs the duplicate filter to remove any waypoints that share a short name with a waypoint that has come before. This option might be used to remove duplicates if you are merging two datasets that were each created in part from a common ancestor dataset.

location option

Suppress duplicate waypoint based on coords.

This option causes the duplicate filter to remove any additional waypoint that has the same coordinates (to six decimal degrees) as a waypoint that came before. This option may be used to remove duplicate waypoints if the names are not expected to be the same. It also might be used along with the `shortname` option to remove duplicate waypoints if the names of several unrelated groups of waypoints might be the same.

all option

Suppress all instances of duplicates.

When this option is specified, GPSBabel will remove all instances of a duplicated waypoint, not just the second and subsequent instances. If your input file contains waypoints A, B, B, and C, the output file will contain waypoints A, B, and C without the `all` option, or just A and C with the `all` option.

Example 4.18. Using the duplicate filter to implement an "ignore list."

This option may be used to implement an "ignore list." In the following example, the duplicate filter is used to remove a list of waypoints to be ignored from a larger collection of waypoints:

```
gpsbabel -i gpx -f waypoints.gpx -i csv -f to_ignore.csv -x dupli-
cate,shortname,all -o gpx -F filtered.gpx
```

correct option

Use coords from duplicate points.

This option is used to change the locations of waypoints without losing any of the other associated information. When this option is specified, the latitude and longitude from later duplicates will replace the latitude and longitude in the original waypoint.

As an example, this option may be used to adjust the locations of "puzzle" geocaches in a Groundspeak pocket query:

Example 4.19. Using the duplicate filter to correct the locations of "puzzle" geocaches

```
gpsbabel -i gpx -f 43622.gpx -i csv -f corrections.csv -x dupli-
cate,shortname,correct -o gpx -F 43622-corrected.gpx
```

After this command is run, the waypoints in the output file will have all of the descriptive information from `43622.gpx`, but waypoints that were also found in `corrections.csv` will have their coordinates replaced with the coordinates from that file.

Remove Points Within Distance (position)

This filter removes points based on their proximity to each other. For waypoints a point is removed if it is within the specified distance of a preceding point. For routes and tracks consecutive points are removed until the distance between the bracketing points is greater than the specified distance.

Example 4.20. Using the position filter to suppress close points

The following command removes multiple points that are within one foot of each other, leaving just one.

```
gpsbabel -i geo -f 1.loc -f 2.loc -x position,distance=1f -o mapsend
-F 3.wpt
```

distance option

Maximum positional distance.

This option specifies the minimum allowable distance between two points. If two points are closer than this distance, only one of them is kept.

The units may be specified by appending a suffix to the supplied number:

'm' for meters, e.g. 3500.0m
'ft' or 'feet' for feet, e.g. 11483ft
'k' or 'km' for kilometers, e.g 3.5000km
'nm' for nautical miles, e.g. 1.8898nm
'mi' for miles, e.g. 2.1748mi
'fa' for fathoms, e.g. 1913.82fa

If no units are specified, the units are assumed to be feet.

all option

Suppress all points close to other points.

This option causes the position filter to remove all points that are within the specified distance of one another, rather than leaving just one of them.

This option may be used to entirely remove clusters of points.

time option

Maximum time in seconds between two points.

Specifies the maximum time in seconds between any two points. If the time difference is larger than what's specified here, the points will not be discarded.

This is useful if you have multiple tracks of the same course and you'd like the filter to consider the tracks the same.

Remove unreliable points with high hdop or vdop (discard)

This filter is used to "fix" unreliable GPS data by discarding points that are believed to be unreliable. You may specify an HDOP and/or VDOP above a specified limit, a minimum number of satellites that must have been in view for a fix to be considered, or both.

HDOP and VDOP are measures of the best possible horizontal or vertical precision for a given configuration of GPS satellites. Higher numbers indicate a higher dilution of precision and therefore mathematically less useful.

Example 4.21. Using the discard filter for HDOP and VDOP.

```
gpsbabel -i gpx -f in.gpx -x discard,hdop=10,vdop=20,hdopandvdop -o gpx
-F out.gpx
```

You may specify a minimum number of satellites.

Example 4.22. Using the discard filter to require at least three satellites.

```
gpsbabel -i gpx -f in.gpx -x discard,sat=3 -o gpx -F out.gpx
```

Contributed by Tobias Minich and Serge Droz.

hdop option

Suppress points with higher hdop.

This option specifies the maximum allowable Horizontal Dilution of Precision (HDOP). By default, any point with an HDOP in excess of this value will be discarded regardless of its VDOP, but see hdopandvdop.

vdop option

Suppress points with higher vdop.

This option specifies the maximum allowable Vertical Dilution of Precision (VDOP). By default, any point with an VDOP in excess of this value will be discarded regardless of its HDOP, but see hdopandvdop.

hdopandvdop option

Link hdop and vdop suppression with AND.

If this option is used, only points that exceed both the maximum allowable HDOP and the maximum allowable VDOP will be discarded. This option requires that both the hdop and vdop options be specified.

sat option

Minimum sats to keep points.

This option specifies the minimum required number of satellites.

fixnone option

Suppress points without fix.

This option is similar to the 'sat' option. There are times when some GPSes will know how many satellites are in view, but not yet computed a valid fix. This option allows you to discard those points.

fixunknown option

Suppress points with unknown fix.

This option is similar to the 'sat' option. Some GPSes will log points with a fix value of 'unknown'. This option allows you to discard those points.

elemin option

Suppress points below given elevation in meters.

This option drops waypoints with an altitude lower than the specified value (in meters). Although GPS altitude isn't very accurate, GPS devices may log faulty waypoints from time to time, such as when near tall buildings. Elevation values that are way off may signify such waypoints. Use this option to the filter to toss known rogue points.

elemax option

Suppress points above given elevation in meters.

This option drops waypoints with an altitude higher than the specified value (in meters). See elemin for an explanation why this may be useful.

matchname option

Suppress points where name matches given name.

This option discards points that have shortnames that match the provided regular expression.

Example 4.23. Discarding specific point by regular expression

For example geocaches typically have names starting with GC followed by an alphanumeric sequence of variable length. To remove all six character long IDs that between (and including) GC1000 and GC2FFF, you could use

```
gpsbabel -i geo -f geocaching.loc -x discard,matchname=GC[1-2]???
```

to discard all GCs followed by exactly three characters.

matchdesc option

Suppress points where description matches given name.

Like `matchname`, but instead matches on the description.

matchcmt option

Suppress points where comment matches given name.

Like `matchname`, but instead matches on the comment.

matchicon option

Suppress points where type matches given name.

Like `matchname`, but instead matches on the icon description.

Resample Track (resample)

The resampling filter can be used to change the sample rate of a track. It is intended to be used with track points that have been sampled at a constant rate. It can be used to change the sample rate by a rational factor. It can also be used to smooth a track with or without changing the sample rate. The filter works across the antimeridian.

Example 4.24. Interpolation with the resampling filter

This examples doubles the sample rate. The data is filtered after interpolation regardless of the order of the options.

```
gpsbabel -t -i unicsv -f data.csv -x resample,interpolate=2,average=2
-o unicsv,utc=0 -F fast.csv
```

Example 4.25. Decimation with the resampling filter

This examples reduces the sample rate by a factor of 4. The data is filtered before decimation regardless of the order of the options.

```
gpsbabel -t -i unicsv -f data.csv -x resample,average=4,decimate=2 -
o unicsv,utc=0 -F slow.csv
```

Example 4.26. Averaging with the resampling filter

This examples averages the adjacent points. A running average filter of length two samples is applied in the forward and reverse directions.

```
gpsbabel -t -i unicsv -f data.csv -x resample,average=2 -o unicsv,utc=0
-F smooth.csv
```

decimate option

Decimate, decrease sample rate by a factor of n.

This options is used to decrease the sample rate. The value of this option is an integer factor to decrease the sample rate by. If using this option the minimum value is two. Normally decimation would also use averaging, but it is not required. This option may be useful in the case of very long tracks that were sampled at an inappropriately high rate.

interpolate option

Interpolate, increase sample rate by a factor of n.

This options is used to increase the sample rate. The value of this option is an integer factor to increase the sample rate by. If using this option the minimum value is two. If using this option the average option must be used with average value greater than or equal to the interpolate value. It is recommended to use an average value that is an integer multiple of the interpolate value.

average option

Running average of n points.

This options is used to control the amount of filtering. The value of this option is the length of the running average filter that is used to smooth the data. The running average filter is applied once in the forward direction and once in the backwards direction. If using this option the minimum value is two.

Reverse stops within routes (reverse)

The reverse filter is used to reverse tracks and routes. It's mostly useful for those few formats where track/route sequence matters and there isn't a way to reverse them using the program itself.

The reversal is performed in the laziest way possible. Timestamps are kept with the original waypoints so the resulting track or route will have the interesting characteristic that time runs backwards. This tends to make Magellan Mapsend, in particular, do a weird thing and place each waypoint on a separate day.

Additionally, if you're using this to reverse a route that navigates, say, an exit ramp or a one way street, you will be in for unpleasant ride. application cares about timestamps

Save and restore waypoint lists (stack)

This filter is designed to solve advanced problems that involve shuffling multiple lists of waypoints, tracks, or routes.

The stack filter can be used to save the current state of the entire collection of data. That state is placed on top of a stack of collections, so you can simultaneously have as many stored collections of data as you can fit in your computer's memory.

The stack filter can be used in conjunction with other filters to implement a "union" or "logical or" functionality. The basic idea is to use the stack to store copies of the original list of waypoints, then use the 'swap' function to replace each copy with a filtered list. Finally, append all of the filtered lists to create one big list, which is then output. The following example finds a list of all points that are either inside county A or inside county B. Any points that are inside both counties are duplicated (but the duplicates can be removed with the DUPLICATE filter; see above.)

```
gpsbabel -i gpx -f in.gpx -x stack,push,copy -x polygon,file=coun-
ty_a.txt -x stack,swap -x polygon,file=county_b.txt -x stack,pop,append
-o gpx -F out.gpx
```

This example reads a large list of waypoints and extracts the points within 20 miles of each of two cities, writing the waypoint descriptions into two different PalmDoc files and exporting all of the points to the GPS receiver:

```
gpsbabel    -i    gpx    -f    indiana.gpx    -x    stack,push,copy    -
x    radius,lat=41.0765,lon=-85.1365,distance=20m    -o    palmdoc,db-
name=Fort\   Wayne   -F   fortwayne.pdb   -x   stack,swap   -x   ra-
dius,lat=39.7733,lon=-86.1433,distance=20m   -o   palmdoc,dbname=Indi-
anapolis  -F  indianapolis.pdb  -x  stack,pop,append  -o  magellan  -F
fwaind.wpt
```

push option

Push waypoint list onto stack.

This is one of three "primary" options to the stack filter.

When this option is specified, the current state is pushed onto the top of the stack. By default, the current state is then cleared, but the `copy` option can be used to cause it to be saved.

pop option

Pop waypoint list from stack.

This is one of three "primary" options to the stack filter.

This option "pops" the collection of data from the top of the stack. By default, the saved state replaces the current state, but see the `discard` and `append` options for alternatives.

swap option

Swap waypoint list with <depth> item on stack.

This is one of three "primary" options to the stack filter.

When this option is specified, the current state is swapped with a saved state from the stack. By default, it is swapped with the top of the stack, but the `depth` can be used to specify a different saved state.

copy option

(push) Copy waypoint list.

This option is only valid when used with the `push` option. When this option is specified, a copy of the current state is pushed onto the stack but the current state is left unchanged. Otherwise, the push operation clears the current data collection.

append option

(pop) Append list.

This option is only valid in conjunction with the `pop`. When it is specified, the topmost collection of data from the stack is appended to the current collection of data.

discard option

(pop) Discard top of stack.

This option is only valid when used with the `pop` option. When this option is specified, the popped state is discarded and the current state remains unchanged.

replace option

(pop) Replace list (default).

This option is only valid when used with the `pop` option. This is the default behavior of the `pop` option, so you should never need to specify it, but it is included for the sake of readability. When this option is specified, the popped state replaces the current state.

depth option

(swap) Item to use (default=1).

This option is only valid when used along with the `swap` option. If specified, it indicates which item on the stack should be swapped with the current state. The default value is 1, which corresponds to the top of the stack.

Simplify routes (simplify)

The Simplify filter is used to simplify routes and tracks for use with formats that limit the number of points they can contain or just to reduce the complexity of a route.

The filter attempts to remove points from each route until the number of points or the error is within the given bounds, while also attempting to preserve the shape of the original route as much as possible.

The quality of the results will vary depending on the density of points in the original route and the length of the original route.

For example, suppose you have a route from Street Atlas 2003 that you wish to use with a Magellan GPS receiver that only supports up to 50 points in a route:

```
gpsbabel -r -i saroute -f RoadTrip.anr -x simplify,count=50 -o magellan
-F grocery.rte
```

count option

Maximum number of points in route.

This option specifies the maximum number of points which may appear in the simplified route. For example, if you specify "count=50", all resulting routes will contain 50 points or fewer.

You must specify either this option or the `error` option.

error option

Maximum error.

This option specifies the maximum allowable error that may be introduced by removing a single point. Used with the `length` and `crosstrack` methods, the value of this option is a distance, specified in miles by default. You may also specify the distance in kilometers by adding a 'k' to the end of the number, meters by adding a 'm', or feet by adding 'ft'. For the `relative` method it is a dimensionless quantity.

How the error is determined depends on whether the `length`, `crosstrack`, or `relative` method is used. If you are using the length method, the error is the change in the length of the route introduced by removing a point. If you are using the crosstrack method, the error is the distance from the point to the line that results if that point is removed. If you are using the relative method, the error is the ratio between the crosstrack error and the horizontal accuracy (derived from HDOP data).

crosstrack option

Use cross-track error (default).

This option instructs GPSBabel to remove points that have the smallest overall effect on the overall shape of the route. Using this method, the first point to be removed will be the one that is closest to a line drawn between the two points adjacent to it.

If neither this option nor the `length` option is specified, this is the default.

length option

Use arclength error.

This option instructs GPSBabel to simplify by removing points that cause the smallest change in the overall length of the route first.

relative option

Use relative error.

Similar to the `crosstrack` method, but the error introduced by removing a point is set into relation to its associated horizontal accuracy, determined as 6m * HDOP. If there is timestamp information, the

distance to the interpolated point between the two neighboring points is used instead of the distance to their connecting line.

The effect of the relative method is similar to a combination of the crosstrack method with the discard filter: points are removed preserving the overall shape of the route (track), but preferably those that are unreliable.

Swap latitude and longitude of all loaded points (swap)

Simple filter to swap the coordinate values (latitude and longitude) of all points. This can be helpful for wrong defined/coded data. Or if you think, you can use one of our xcsv formats, but latitude and longitude are in opposite order.

Transform waypoints into a route, tracks into routes, ... (transform)

This filter can be used to convert GPS data between different data types.

Some GPS data formats support only some subset of waypoints, tracks, and routes. The transform filter allows you to convert between these types. For example, it can be used to convert a pile of waypoints (such as those from a CSV file) into a track or vice versa.

The following example show you how to create a route from a waypoint table.

```
gpsbabel -i csv -f waypts.txt -x transform,rte=wpt -o gpx -F route.gpx
```

Only the first letter of option value decides which transformation will be done. Depending on the used option it can be only 'W' for waypoints, 'R' for routes or 'T' for tracks.

wpt option

Transform track(s) or route(s) into waypoint(s) [R/T].

This option selects the destination type of this filter to be waypoints. Choose this when you want to convert tracks or routes into waypoints.

Example 4.27. Converting a track to a sequence of waypoints

Say you you have a KML file that contains a track but you want to convert it to a CSV file that can contain only waypoints, perhaps to import into a spreadsheet. Use the following command:

```
gpsbabel -i kml -f blah.kml -x transform,wpt=trk -o csv -F blah.txt
```

rte option

Transform waypoint(s) or track(s) into route(s) [W/T].

This option selects the destination type of this filter to be routes. Choose this when you want to convert tracks into waypoints routes. A single route will be created in the sequence they appear in the input.

Example 4.28. Converting a pile of waypoints to a GPX route

Say you you have a data file that came from CSV file that you want to convert to a GPX route that can be loaded into Basecamp. Use the following command:

```
gpsbabel -i csv -f blah.txt -x transform,rte=wpt -o gdb -F blah.gdb
```

trk option

Transform waypoint(s) or route(s) into tracks(s) [W/R].

This option selects the destination type of this filter to be tracks. Choose this when you want to create tracks from a list of waypoints or routes. A single track will be created in the sequence they appear in the input.

Example 4.29. Converting a pile of waypoints to a GPX track

Say you you have a data file that came from CSV file that you want to convert to a GPX track that can be loaded into Basecamp. Use the following command:

```
gpsbabel -i csv -f blah.txt -x transform,trk=wpt -o gdb -F blah.gdb
```

rptdigits option

Number of digits in generated names.

This option lets you configure how many digits GPSBabel uses for numbering generated route point names.

When GPSBabel creates route points during the transformation process these points are sequentially numbered and named "RPTnnn" where nnn represent the number. By default GPSBabel uses 3 digits for these numbers. Rationale: This way a large number of route points can be uniquely named while the generated names are limited to 6 characters. This limitation is imposed by specific GPS-devices.

Using this option GPSBabel can be configured to use less or more digits for the generated names. This option is best used in conjunction with the rptname option.

Example 4.30. Convert a GPX track to a GPX route, deleting the original track, using 2 digits for the generated numbers.

```
gpsbabel -i gpx -f track.gpx -x transform,wpt=trk,del,rptdigits=2 -o
gpx -F route.gpx
```

rptname option

Use source name for route point names.

With this option you can decide to let GPSBabel name generated route points according to their source track name.

GPSBabel creates route points during the transformation process named "RPTnnn" where nnn is a numeric part.

Using this option GPSBabel can be configured to replace the "RPT" part of the generated names by the name of the source track during the transformation process. This is especially useful if several differently named tracks are contained in the source file which should each be transformed into routes.

Example 4.31. Convert a GPX track to a GPX route, deleting the original track, naming the generated points like the original track name.

```
gpsbabel -i gpx -f track.gpx -x transform,wpt=trk,del,rptname=y -o gpx
-F route.gpx
```

del option

Delete source data after transformation.

This option, when used in connection with the wpt, rte, or trk options, tells GPSBabel to delete the source data after conversion. This is most useful if you are trying to avoid duplicated data in the output.

Example 4.32. Convert a GPX track to GPX waypoints, tossing the original track

```
gpsbabel -i gpx -f blah.gpx -x transform,wpt=trk,del -o gpx -F convert-
ed.gpx
```

timeless option

Create transformed points without times.

This option tells GPSBabel to create points without creation times instead of copying the creation time from the source points.

Validate internal data structures (validate)

This filter can be used to check internal data structures for validity. The output of the filter is identical to the input, but if corruption is found a fatal error will be issued.

checkempty option

Check for empty input.

This option will cause a fatal error if there are no waypoints, no route waypoints and no track waypoints, i.e. the reader didn't produce anything.

debug option

Output debug messages instead of possibly issuing a fatal error.

This option will output verbose messages reporting the state of the internal data structures holding waypoints, routes and tracks. Detected problems will normally produce a fatal error, but with this option in effect no error will be thrown allowing continued processing.

Appendix A. Supported Datums

Some formats in GPSBabel support multiple datums. For example, the `datum` option to the garmin_txt format allows you to specify a datum for the output file.

The following is a list of the datums supported by GPSBabel.

Adindan
AFG
Ain-El-Abd
Alaska-NAD27
Alaska-Canada
Anna-1-Astro
ARC 1950 Mean
ARC 1960 Mean
Asc Island 58
Astro B4
Astro Beacon E
Astro pos 71/4
Astro stn 52
Australia Geo 1984
Bahamas NAD27
Bellevue IGN
Bermuda 1957
Bukit Rimpah
Camp_Area_Astro
Campo_Inchauspe
Canada_Mean(NAD27)
Canal_Zone_(NAD27)
Canton_Island_1966
Cape
Cape_Canaveral_mean
Carribean NAD27
Carthage
Cent America NAD27
Chatham 1971
Chua Astro
Corrego Alegre
Cuba NAD27
Cyprus
Djakarta(Batavia)
DOS 1968
Easter lsland 1967
Egypt
European 1950
European 1950 mean
European 1979 mean
Finnish Nautical
Gandajika Base
Geodetic Datum 49
Ghana
Greenland NAD27
Guam 1963
Gunung Segara
Gunung Serindung 1962
GUX1 Astro
Herat North
Hjorsey 1955
Hong Kong 1963
Hu-Tzu-Shan
Indian
Iran
Ireland 1965
ISTS 073 Astro 69
Johnston Island 61
Kandawala
Kerguelen Island
Kertau 48
L.C. 5 Astro
La Reunion
Liberia 1964
Luzon
Mahe 1971
Marco Astro
Masirah Is. Nahrwan
Massawa
Merchich
Mexico NAD27
Midway Astro 61
Mindanao
Minna
Montjong Lowe
Nahrwan
Naparima BWI
North America 83
N. America 1927 mean
Observatorio 1966
Old Egyptian
Old Hawaiian_mean
Old Hawaiian Kauai
Old Hawaiian Maui
Old Hawaiian Oahu
Oman
OSGB36
Pico De Las Nieves
Pitcairn Astro 67
S. Am. 1956 mean(P)
S. Chilean 1963 (P)
Puerto Rico
Pulkovo 1942
Qornoq
Quatar National
Rome 1940
S-42(Pulkovo1942)
S.E.Asia_(Indian)
SAD-69/Brazil
Santa Braz
Santo (DOS)
Sapper Hill 43
Schwarzeck
Sicily
Sierra Leone 1960
S. Am. 1969 mean
South Asia
Southeast Base
Southwest Base
Tananarive Obs 25
Thai/Viet (Indian)
Timbalai 1948
Tokyo mean
Tristan Astro 1968
United Arab Emirates
Viti Levu 1916
Wake Eniwetok 60
WGS 72
WGS 84
Yacare
Zanderij
Sweden

Appendix B. Garmin Icons

Following is a list of the valid values for the garmin `deficon` option. These values are also used internally by the GDB format.

ATV	Contact, Glasses	Hunting Area	Number 0, Green	Scales
Airport	Contact, Goatee	Ice Skating	Number 0, Red	Scenic Area
Amusement Park	Contact, Kung-Fu	Information	Number 1, Blue	School
Anchor	Contact, Panda	Intersection	Number 1, Green	Seafood
Anchor Prohibited	Contact, Pig	Intl freeway hwy	Number 1, Red	Seaplane Base
Animal Tracks	Contact, Pirate	Intl national hwy	Number 2, Blue	Shipwreck
Asian Food	Contact, Ranger	Italian food	Number 2, Green	Shopping Center
Bait and Tackle	Contact, Smiley	Large Ramp intersection	Number 2, Red	Short Tower
Ball Park	Contact, Spike	Large exit without services	Number 3, Blue	Shower
Bank	Contact, Sumo	Letter A, Blue	Number 3, Green	Ski Resort
Bar	Controlled Area	Letter A, Green	Number 3, Red	Skiing Area
Beach	Convenience Store	Letter A, Red	Number 4, Blue	Skull and Crossbones
Beacon	Cover	Letter B, Blue	Number 4, Green	Small City
Bell	Covey	Letter B, Green	Number 4, Red	Small Game
Big Game	Crossing	Letter B, Red	Number 5, Blue	Soft Field
Bike Trail	Dam	Letter C, Blue	Number 5, Green	Square, Blue
Blind	Danger Area	Letter C, Green	Number 5, Red	Square, Green
Block, Blue	Deli	Letter C, Red	Number 6, Blue	Square, Red
Block, Green	Department Store	Letter D, Blue	Number 6, Green	Stadium
Block, Red	Diamond, Blue	Letter D, Green	Number 6, Red	State Hwy
Blood Trail	Diamond, Green	Letter D, Red	Number 7, Blue	Steak
Boat Ramp	Diamond, Red	Letterbox Cache	Number 7, Green	Street Intersection
Border Crossing (Port Of Entry)	Diver Down Flag 1	Levee	Number 7, Red	Stump
Bottom Conditions	Diver Down Flag 2	Library	Number 8, Blue	Summit
Bowling	Dock	Light	Number 8, Green	Swimming Area
Bridge	Dot, White	Live Theater	Number 8, Red	TACAN
Building	Drinking Water	Localizer Outer Marker	Number 9, Blue	Tall Tower
Buoy, White	Dropoff	Locationless (Reverse) Cache	Number 9, Green	Telephone
Campground	Elevation point	Lodge	Number 9, Red	Tide/Current Prediction Station
Car	Event Cache	Lodging	Oil Field	Toll Booth
Car Rental	Exit	Man Overboard	Open 24 Hours	TracBack Point
Car Repair	Exit without services	Marina	Oval, Blue	Trail Head
Cemetery	Fast Food	Medical Facility	Oval, Green	Tree Stand
Church	First approach fix	Micro-Cache	Oval, Red	Treed Quarry
Circle with X	Fishing Area	Mile Marker	Parachute Area	Triangle, Blue
Circle, Blue	Fishing Hot Spot Facility	Military	Park	Triangle, Green
Circle, Green	Fitness Center	Mine	Parking Area	Triangle, Red
Circle, Red	Flag	Missed approach point	Pharmacy	Truck
City (Capitol)	Flag, Blue	Movie Theater	Picnic Area	Truck Stop
City (Large)	Flag, Green	Multi-Cache	Pin, Blue	Tunnel
City (Medium)	Flag, Red	Multi-Cache	Pin, Green	U Marina
City (Small)	Food Source	Museum	Pin, Red	U stump

City Hall	Forest	Navaid, Amber	Pizza	US hwy
Civil	Furbearer	Navaid, Black	Police Station	Ultralight Area
Coast Guard	Gambling/casino	Navaid, Blue	Post Office	Unknown Cache
Contact, Afro	Gas Station	Navaid, Green	Post Office	Upland Game
Contact, Alien	Geocache	Navaid, Green/Red	Private Field	VHF Omni-range
Contact, Ball Cap	Geocache Found	Navaid, Green/White	Puzzle Cache	VOR-DME
Contact, Big Ears	Geographic place name, Man-made	Navaid, Orange	RV Park	VOR/TACAN
Contact, Biker	Geographic place name, land	Navaid, Red	Radio Beacon	Virtual cache
Contact, Blonde	Geographic place name, water	Navaid, Red/Green	Ramp intersection	Water Hydrant
Contact, Bug	Ghost Town	Navaid, Red/White	Rectangle, Blue	Water Source
Contact, Cat	Glider Area	Navaid, Violet	Rectangle, Green	Waterfowl
Contact, Clown	Golf Course	Navaid, White	Rectangle, Red	Waypoint
Contact, Dog	Ground Transportation	Navaid, White/Green	Reef	Webcam Cache
Contact, Dreadlocks	Heliport	Navaid, White/Red	Residence	Weed Bed
Contact, Female1	Horn	Non-directional beacon	Restaurant	Winery
Contact, Female2	Hotel	Null	Restricted Area	Wrecker
Contact, Female3	House	Number 0, Blue	Restroom	Zoo

Appendix C. GPSBabel XCSV Style Files

Introduction to GPSBabel Styles

Often it is desirable to add a new file format for "one-off" work (perhaps you want to export something to a spreadsheet or graphing program) or to read a format that GPSBabel does not yet support. For suitably simple formats, this can be done by a user with no programming experience by providing a GPSBabel style file.

For a format to be described by a style file, it must be predictable and generally readable by a human. Formats with binary or unreadable content are not good fits for this scheme. It should have:

A fixed header at the beginning, if it has any at all. This is called a 'prologue'.
Waypoints that are grouped by fixed separators, often a newline. In style file parlance, this is called a 'record'.
Traits of that waypoint described in that record. In the style files, these are called 'fields' and examples may include longitude or a name.
Fields that are grouped by fixed separators, often a comma or a tab. In the style files, this is called the field separator. Fields may be enclosed by characters, such as a double quote.
A fixed footer at the end, if it has any at all. This is called the 'epilogue'.

Once you have created a style file that describes the file format you have or want, you must tell GPSBabel to use the xcsv format and have the xcsv format use that file. If you created a new style file called "mystyle.style" and you want to write the waypoints from a GPX file named "mine.gpx" to it, you would issue a command like:

```
gpsbabel -i gpx -f mine.gpx -o xcsv,style=mystyle.style -F mine.new
```

You might then examine `mine.new` to see if it met your expectations. If not, you could continue to tweak `mystyle.style` until it did, rerunning the above command each time. If 'mystyle' is a format that describes a popular program or is likely to be of use to others, you can then share `mystyle.style` with other GPSBabel users. Send it along with a coherent description to the GPSBabel-Misc mailing list for consideration to be included in a future version.

Style file overview

The first and foremost important step is understanding how the style file is laid out itself. The format is:

`DIRECTIVE<whitespace>VALUE`

Where <whitespace> is one or more spaces or tabs. There should be no spaces or tabs at the beginning of the line; all directives start at the left edge in column zero.

An example style format is shown here:

```
# Format: MS S&T 2002/2003
# Author: Alex Mottram
#   Date: 12/09/2002
```

```
#

DESCRIPTION       Microsoft Streets and Trips 2002-2006
EXTENSION             txt

#
# FILE LAYOUT DEFINITIIONS:
#
FIELD_DELIMITER     TAB
RECORD_DELIMITER  NEWLINE
BADCHARS    ,"

PROLOGUE Name  Latitude Longitude   Description URL   Type  Container   Diff  Terr

#
# INDIVIDUAL DATA FIELDS, IN ORDER OF APPEARANCE:
# NOTE: MS S&T ONLY IMPORTS DATA, IT DOESN'T
#       EXPORT THIS ANYWHERE SO WE CAN HAVE OUR
#       WAY WITH THE FORMATTING.
#
IFIELD   SHORTNAME, "", "%s"      # Name
IFIELD   LAT_DECIMAL, "", "%f"    # Latitude
IFIELD   LON_DECIMAL, "", "%f"    # Longitude
IFIELD   DESCRIPTION, "", "%s"    # Name 2 (Big Description)
IFIELD   URL, "", "%s"         # URL
IFIELD   GEOCACHE_TYPE, "", "%s"     # Geocache Type
IFIELD   GEOCACHE_CONTAINER, "", "%s"   # Geocache Type
IFIELD   GEOCACHE_DIFF, "", "%3.1f"  # Geocache Type
IFIELD   GEOCACHE_TERR, "", "%3.1f"  # Geocache Type
```

Each of these lines will be explained in the following sections.

Internal Constants

A few internal constants are defined in the XCSV parser to make the style file simpler. They may or may not be used and are optional in most cases. Note that only certain style file directives map these constants.

Style Constant: COMMA
Maps to Char(s): ,
Style Constant: COMMASPACE
Maps to Char(s): ,<space>
Style Constant: SINGLEQUOTE
Maps to Char(s): '
Style Constant: DOUBLEQUOTE
Maps to Char(s): "
Style Constant: COLON
Maps to Char(s): :
Style Constant: SEMICOLON
Maps to Char(s): ;
Style Constant: NEWLINE
Maps to Char(s): \n
Style Constant: CR
Maps to Char(s): \r
Style Constant: CRNEWLINE

Maps to Char(s): \r\n
Style Constant: TAB
Maps to Char(s): \t
Style Constant: SPACE
Maps to Char(s): <space>
Style Constant: HASH
Maps to Char(s): #
Style Constant: PIPE
Maps to Char(s): |
Style Constant: WHITESPACE
Maps to Char(s): see below

WHITESPACE

The WHITESPACE constant has special properties. When reading data, WHITESPACE refers to sequential runs of SPACES and/or TABS. When writing data, WHITESPACE is always a single SPACE.

For example, the following line:

```
SOME_NAME       30.1208 -91.1365    SOME OTHER NAME
```

Parses into the following data fields:

```
SOME_NAME,30.1208,-91.1365,SOME,OTHER,NAME
```

COMMENTS

Anything after a hash (#) on a line is not parsed. For example:

```
#THIS ENTIRE LINE IS A COMMENT.
#FIELD   LAT_DECIMAL, "", "%f"   THIS ENTIRE LINE IS A COMMENT
FIELD LAT_DECIMAL, "", "%f"  # ONLY THIS SENTENCE IS A COMMENT.
```

Global Properties of the File

There are a few available directives to describe general traits of the file being described and not specific data within the file itself.

DESCRIPTION

This is the description of the file format being described. This text appears in the help screens and in menus used by the various GUI wrappers.

EXTENSION

This directive gives the filename extension generally associated with this file.

ENCODING

Describes the character set used by this format. The value given must be one listed by 'gpsbabel -l'. example:

```
ENCODING          UTF-8 # Use UTF-8 for input and output.
```

DATUM

This value specifies the GPS datum to be used on read or write. Valid values for this option are listed in Appendix A, *Supported Datums* .

```
DATUM             European 1950
```

DATATYPE

Specifies the kind of data we have to read or write.

By default all data are seen as waypoint data. With DATATYPE you are now able to bind a specific type to this format. Possible values are WAYPOINT, ROUTE or TRACK.

```
DATATYPE          ROUTE # route-only format
```

GPSBabel Behavior Directives

There are a few available directives to control some of the internal processing functions of GPSBabel.

SHORTLEN

This sets the maximum allowed shortname length when using the internal shortname synthesizer.

example:

```
SHORTLEN 16 # shortnames will be at most 16 characters long.
```

SHORTWHITE

This tells the shortname synthesizer whether or not to allow whitespace in the synthesized shortnames. Allowed values are zero and one.

example:

```
SHORTWHITE  0  # Do not allow whitespace in shortname.
   SHORTWHITE   1 # Allow whitespace in shortname.
```

Defining the Layout of the File

The first few directives define the layout the physical file itself:

FIELD_DELIMITER

The field delimiter defines the character(s) that separate the fields in the rows of data inside the XCSV file. Common field delimiters are commas and tabs. (referred to as "comma separated values" and "tab separated values")

examples:

```
FIELD_DELIMITER    COMMA
   FIELD_DELIMITER    ~
```

The directive FIELD_DELIMITER is parsed for STYLE CONSTANTS as defined in the table above.

FIELD_ENCLOSER

The field encloser defines the character(s) that surround the field values. Common field enclosers are single and double quote marks. Many styles will leave this directive unset. If set, it will be applied to all fields.

examples:

```
FIELD_ENCLOSER    DOUBLEQUOTE
   FIELD_ENCLOSER    SINGLEQUOTE
```

The directive FIELD_ENCLOSER is parsed for STYLE CONSTANTS as defined in the table above.

RECORD_DELIMITER

The record delimiter defines that character(s) that separate ROWS of data (FIELDS) in the XCSV file. The most common record delimiters are NEWLINE and CR (carriage return).

examples:

```
RECORD_DELIMITER    NEWLINE
   RECORD_DELIMITER    |
```

The directive RECORD_DELIMITER is parsed for STYLE CONSTANTS as defined in the table above.

BADCHARS

Bad characters are things that should *never* be written into the XCSV file as data on output. GPSBabel automatically includes any non-blank FIELD_DELIMITER and FIELD_ENCLOSER and RECORD_DELIMITER characters as BADCHARS by default.

examples:

```
BADCHARS    COMMA
  BADCHARS    ~|
```

The directive BADCHARS is parsed for STYLE CONSTANTS as defined in the table above.

PROLOGUE

A prologue is basically constant data that is written to the output file BEFORE any waypoints are processed. PROLOGUE can be defined multiple times in the style file, once for each "line" before the data begins. This is commonly used in XCSV files as a "header" row.

examples:

```
PROLOGUE OziExplorer Waypoint File Version 1.1
  PROLOGUE  WGS 84
  PROLOGUE  Symbol,Name,Latitude,Longitude
```

EPILOGUE

An Epilogue is the same as a prologue, except this data is written at the END of the file. See the examples for PROLOGUE above.

Defining Fields Within the File

A field defines data. There are two different classifications of FIELDS, IFIELD (file input) and OFIELD (file output). In the absence of any OFIELDS, IFIELDS are use as both input and output. The existence of OFIELDS is primarily to allow more flexible mapping of GPSBabel data to output data (say, for instance, to map the internal GPSBabel "description" variable to two or more fields on output). For all practical purposes, IFIELDS and OFIELDS are defined the same way in the style file.

The following per-field options are defined:

- "no_delim_before" is supported on in OFIELD tags to specify that this field should be written without a field delimiter before it. It's useful for limited field concatenation.
- "absolute" is supported on OFIELD tags for lat and lon to indicate that only absolute values (never negative) are to be printed.
- "optional" is supported only OFIELD tags and indicates that the field may or may not be available in the source data. If the field is absent, no trailing field separator is written.

  This attribute is most useful when paired with "no_delim_before" as it allows you to concatenate fields without concern for whether those fields are actually populated or not.

There are several different types of fields that may be defined. Each field consists of three pieces of information: the FIELD TYPE, a DEFAULT VALUE, and a PRINTF CONVERSION (for output). In many cases, not all pieces are used, but all 3 pieces are required. Additionally, an fourth field is supported that modifies the behavior of the field being described.

FIELDS should be defined in the style file in the logical order that they appear in the data, from left to right. This is the order in which they are parsed from input and written to output.

The fields used by the XCSV parser are as follows:

IGNORE

IGNORE fields are, guess what, ignored on input. Internally, IGNORE fields are treated as CHARACTER data, and as such, require a printf conversion for a character array.

examples:

```
IFIELD IGNORE,"","%14.14s"   # (writes a 14 character blank field)
   IFIELD IGNORE,"","%s"        # (writes a blank field on output)
```

CONSTANT

CONSTANT fields are, of course, constant. They are ignored on input, however they write CONSTANT data on output. As such, they require a DEFAULT VALUE and a printf conversion for a character array.

examples:

```
IFIELD CONSTANT,"FFFFFF","%s"   # (writes "FFFFFF" in the field)
   IFIELD CONSTANT,"01/01/70","%s" # (a constant date field)
```

INDEX

An INDEX field is used ONLY on output. The INDEX constant defines a field that, at output, contains the sequence number of the waypoint being written, starting at 0. An index is managed internally as an INTEGER and requires an INTEGER printf conversion. An INDEX has one special property. The DEFAULT VALUE of the index is added to the index on each iteration (to allow indexes starting at 1, 100, etc..).

examples:

```
IFIELD INDEX,"0","%04d"     # (Starts counting at zero)
   IFIELD INDEX,"","%04d"      # (Starts counting at zero)
   IFIELD INDEX,"1","%04d"     # (Starts counting at one)
```

SHORTNAME

A SHORTNAME is generally the waypoint name of the data being processed. SHORTNAME maps directly to the GPSBabel variable ->shortname. A SHORTNAME is CHARACTER data and requires a character array printf conversion.

example:

```
IFIELD SHORTNAME,"","%s"
```

DESCRIPTION

A DESCRIPTION is generally a long description of the waypoint. A DESCRIPTION maps to the GPSBabel variable ->description and is otherwise handled exactly like a SHORTNAME.

examples:

```
IFIELD DESCRIPTION,"","%s"
```

NOTES

NOTES are generally everything else about a waypoints. NOTES map to the GPSBabel variable ->notes and is otherwise handled exactly like a SHORTNAME.

URL

URL is a URL for the waypoint. URL maps to the GPSBabel variable ->url and is otherwise handled exactly like a SHORTNAME.

example:

```
IFIELD URL,"","%s"
```

URL_LINK_TEXT

URL_LINK_TEXT is a textual description of where a URL points. URL_LINK_TEXT maps to the GPSBabel variable ->url_link_text and is otherwise handled exactly like a SHORTNAME.

example:

```
IFIELD URL_LINK_TEXT,"","%s"
```

ICON_DESCR

ICON_DESCR is a textual description of an icon type for a waypoint. ICON_DESCR maps to the GPSBabel variable ->icon_desc and is otherwise handled exactly like a SHORTNAME.

example:

```
IFIELD ICON_DESCR,"","%s"
```

LAT_DECIMAL

LAT_DECIMAL defines LATITUDE in DECIMAL format. Note that this is a PURE signed decimal format (i.e. -91.0000). This data is handled internally as a DOUBLE PRECISION FLOAT and requires a FLOATING POINT printf conversion.

example:

```
IFIELD LAT_DECIMAL,"","%f"
```

LON_DECIMAL

See LAT_DECIMAL, except LON_DECIMAL defines LONGITUDE.

LAT_INT32DEG

LAT_INT32DEG defines LATITUDE in what I call INT32DEGREES. This value is a signed LONG INTEGER and requires a LONG INTEGER printf conversion. (This format is only used by some DeLorme products.)

example:

```
IFIELD LAT_INT32DEG,"","%ld"
```

LON_INT32DEG

See LON_INT32DEG except LON_INT32DEG defines LONGITUDE.

LAT_DECIMALDIR / LAT_DIRDECIMAL

LAT_DECIMALDIR and LAT_DIRDECIMAL define LATITUDE in DECIMAL format with the added bonus of a 'N/S' or 'E/W' direction character. This data is handled internally as a DOUBLE PRECISION FLOAT and a single CHARACTER and requires a FLOATING POINT as well as a CHARACTER printf conversion. The only difference between the two is whether the directional character appears before (LAT_DIRDECIMAL) or after (LAT_DECIMALDIR) the decimal number.

examples:

```
IFIELD LAT_DECIMALDIR,"","%f %c"     #  (writes 31.333 N)
```

```
   IFIELD LAT_DIRDECIMAL,"","%c %f"     #  (writes N 31.333)
```

LON_DECIMALDIR / LON_DIRDECIMAL

Same as LAT_DECIMALDIR / LAT_DIRDECIMAL except LON_ defines LONGITUDE.

LAT_DIR / LON_DIR

LAT_DIR returns the single character 'N' or 'S' depending on the hemisphere of the latitude. LON_DIR returns 'E' or 'W' depending on the hemisphere of the longitude.

LAT_HUMAN_READABLE

LAT_HUMAN_READABLE defines LATITUDE in a human-readable format. This format is probably the most expressive format. It is similar to LAT_DECIMALDIR in that it requires multiple printf conversions, but it is far more flexible as to the contents of those conversions. On read, the printf conversions are ignored and GPSBabel attempts to determine the latitude and longitude based on what is in the file.

examples:

```
#  (writes N 31 40.000)
   IFIELD LAT_HUMAN_READABLE,"","%c %d %f"
   #  (writes "31 deg 40.000 min N")
   IFIELD LAT_HUMAN_READABLE,"","%d deg %f min %c"
   #  Note that this string will confuse the reading routine due
   #  to the letter "n" in "min" and the letter "e" in "deg."
   # (writes 31 40 00.000N)
   IFIELD LAT_HUMAN_READABLE,"","%d %d %f%c"
```

MAP_EN_BNG

MAP_EN_BNG converts coordinates from/to British National Grid (BNG).

The only supported order of the items is: Map,Easting,Northing. During output all coordinates have to be located within this limited area.

examples:

```
IFIELD MAP_EN_BNG,"","%s%5d %5d"   #  (writes i.e. "SJ00001 00001")
   IFIELD MAP_EN_BNG,"","%s %d %d"    #  (writes i.e. "TQ 888 999")
```

LON_HUMAN_READABLE

See LAT_HUMAN_READABLE except LON_HUMAN_READABLE defines LONGITUDE.

LATLON_HUMAN_READABLE

LATLON_HUMAN_READABLE is like LAT_HUMAN_READABLE and LON_HUMAN_READABLE except that it reads and writes both latitude and longitude as a single field. On write, the same format specifier is used for both coordinates. On read, GPSBabel does exactly the same thing it does for LAT_HUMAN_READABLE or LON_HUMAN_READABLE.

example:

```
IFIELD LATLON_HUMAN_READABLE,"","%c %d %f"
           # (writes "N 31 40.126 W 85 09.62" as a single field)
```

LAT_NMEA

Defines the latitude in the format used by the NMEA standard which is degrees multiplied by 100 plus decimal minutes.

example:

```
IFIELD  LAT_NMEA, "%f", "%08.3f"     # (writes  3558.322)
```

LAT_DDMMDIR

Derived from the LAT_NMEA latitude format, with degrees * 100 plus decimal minutes, but using an additional specifier to position the 'N' or 'S' instead of a leading minus sign (or absence thereof) to give direction from zero.

```
IFIELD LAT_DDMMDIR, "%f", "%08.3f%c" # (writes "5334.192S" giving
 -53.56987 degrees latitude)
```

LON_NMEA

Defines the longitude in the format used by the NMEA standard which is degrees multiplied by 100 plus decimal minutes.

Example:

```
IFIELD  LON_NMEA, "%f", "%010.3f"  # (writes -08708.082)
```

LON_DDMMDIR

Derived from the LON_NMEA longitude format, with degrees * 100 plus decimal minutes, but using an additional character format character to position the 'E' or 'W' instead of a leading minus sign (or absence thereof) to give direction from zero.

Example:

```
IFIELD LON_DDMMDIR, "%f", "%010.3f%c" # (writes "01232.745W" giving
 -12.54575 degrees
longitude)
```

LAT_10EX / LON_10EX

Defines the latitude or longitude in the format used i.e. by TomTom Navigator itinerary files. It is degrees multiplied by 10 power X. X have to be replaced with a valid decimal value. A factor of 10000 would be generated by LAT_10E5 as shown in the examples below.

examples:

```
IFIELD  LAT_10E5, "%f", "%.f"       # (writes  3558322)

IFIELD  LON_10E5, "%f", "%.f"       # (writes -8708082)
```

UTM

A location in UTM has several components: a zone, a northing, and an easting. The UTM format specifier is the most common representation of these.

example:

```
IFIELD UTM, "", "%s" # writes 6S 519045 3984035  -the easting is first
 by convention.
```

UTM_EASTING

This is the decimal component representing the easting

example:

```
IFIELD UTM_EASTING, "", "%.0f"  # outputs 519045
```

UTM_NORTHING

This is the decimal component representing the northing

example:

```
IFIELD UTM_NORTHING "", "%.0f"  # outputs 3984035
```

UTM_ZONE

The UTM zone.

example:

```
IFIELD UTM_ZONE "", "%d"  # outputs 6
```

UTM_ZONEC

The UTM Zone character.

example:

```
IFIELD UTM_ZONEC "", "%c"  # outputs S
```

The full UTM zone and latitude band.

example:

```
IFIELD UTM_ZONEF "", "%d%c"  # outputs 6S
```

ALT_FEET

ALT_FEET is the position's ALTITUDE in FEET. This value is treated as a SIGNED DOUBLE PRECISION FLOAT and requires a FLOATING POINT printf conversion.

example:

```
IFIELD ALT_FEET,"","%.0f"
```

ALT_METERS

ALT_METERS is identical to ALT_FEET with the exception that the altitude is in METERS.

HEART_RATE

Heart rate, measured in beats per minute. Only valid for units with heart rate monitor features (i.e. Garmin Forerunner 301).

example:

```
IFIELD HEART_RATE,"","%d"
```

CADENCE

Cadence in revolutions per minute. Only valid for units with heart rate monitor features (i.e. Garmin Edge 305).

example:

```
IFIELD CADENCE,"","%d"
```

POWER

Cycling power in Watts. Only valid for units with power meter features (i.e. Garmin Edge 305).

example:

```
IFIELD POWER,"","%.1f"
```

TEMPERATURE

Temperature in degrees Celsius.

example:

```
IFIELD TEMPERATURE,"","%.1f"
```

TEMPERATURE_F

Temperature in degrees Fahrenheit.

example:

```
IFIELD TEMPERATURE_F,"","%.1f"
```

EXCEL_TIME

EXCEL_TIME is the waypoint's creation time, if any. This is actually the decimal days since 1/1/1900 and is handled internally as a DOUBLE PRECISION FLOAT and requires a FLOATING POINT printf conversion.

example:

```
IFIELD EXCEL_TIME,"","%11.5f"
```

TIMET_TIME

TIMET_TIME is the waypoint's creation time, if any. This is actually the integer seconds since 1970-01-01T00:00:00 UTC. It is handled internally as a 64 bit integer and requires a LONG LONG INTEGER printf conversion.

example:

```
IFIELD TIMET_TIME,"","%lld"
```

TIMET_TIME_MS

TIMET_TIME_MS is the same as TIMET_TIME, but expressed in milliseconds. It too is handled internally as a 64 bit integer and requires a LONG LONG INTEGER printf conversion.

example:

```
IFIELD TIMET_TIME_MS,"","%lld"
```

YYYYMMDD_TIME

YYYYMMDD_TIME is the waypoint's creation time in UTC, if any. It's a single decimal field containing four digits of year, two digits of month, and two digits of date. Internally it is a LONG INTEGER and thus requires a LONG INTEGER printf conversion.

example:

```
IFIELD YYYYMMDD_TIME,"","%ld"
```

GMT_TIME

GMT_TIME is the waypoint's creation time, in UTC time zone. It uses the strptime conversion format tags. It can be used to parse/print a date and time in one column, or a date in one column and a time in another column. If used to parse a date and a time in separate columns the date and time can be in either order.

example with a a date followed by a time using a 12 hour clock and an AM/PM indication:

```
IFIELD GMT_TIME,"","%m/%d/%Y %I:%M:%S %p"
```

example with a a date followed by a time using a 24 hour clock:

```
IFIELD GMT_TIME,"","%Y/%m/%d"
IFIELD GMT_TIME,"","%H:%M:%S"
```

Search the web for 'strptime man page' for details strptime, but one such page can be found at http://www.die.net/doc/linux/man/man3/strptime.3.html

LOCAL_TIME

LOCAL_TIME is the waypoint's creation time, in the local time zone. It uses strptime conversion format tags. It can be used to parse/print a date and time in one column, or a date in one column and a time in another column. If used to parse a date and a time in separate columns the date and time can be in either order. See GMT_TIME for examples and a reference.

ISO_TIME

ISO_TIME is the waypoint's creation time, in ISO 8601 format, which include time zone information. It is expected to be in the format yyyy-mm-ddThh:mm:sszzzzz where zzzzzz is the local time offset or the character Z for UTC time. On output, UTC 'Z' time zone will always be used.

example:

```
IFIELD ISO_TIME,"","%s"
```

ISO_TIME_MS

ISO_TIME_MS is much like ISO_TIME, but expresses milliseconds at the end of the timestamp. It is thus in the format yyyy-mm-ddThh:mm:ss.SSSzzzzz where 'SSS' is milliseconds and zzzzzz is the local time offset or the character Z for UTC time. On output, UTC 'Z' time zone will always be used.

example:

```
IFIELD ISO_TIME_MS,"","%s"
```

NET_TIME

Microsoft dot net represents times in 100 nanosecond intervals since midnight Jan 1/0001 GMT, giving absurdly large numbers like 633943150010000000 for mid-November, 2009. NET_TIME is how to represent those in GPSBabel.

example:

```
IFIELD NET_TIME,"","%lld"
```

GEOCACHE_DIFF

GEOCACHE_DIFF is valid only for geocaches and represents a DOUBLE PRECISION FLOAT. This is the geocache "difficulty" rating as defined by Groundspeak. A "three and a half star" cache would therefore be "3.5"

example:

```
IFIELD GEOCACHE_DIFF,"","%3.1f"
```

GEOCACHE_TERR

GEOCACHE_TERR is valid only for geocaches and represents a DOUBLE PRECISION FLOAT. This is the geocache "terrain" rating as defined by Groundspeak. A "three and a half star" cache would therefore be "3.5"

example:

```
IFIELD GEOCACHE_TERR,"","%3.1f"
```

GEOCACHE_CONTAINER

GEOCACHE_CONTAINER is valid only for geocaches and is heavily influenced by the Groundspeak container types. Examples would include "Micro" and "Virtual".

example:

```
GEOCACHE_CONTAINER,"","%s"
```

GEOCACHE_TYPE

GEOCACHE_TYPE is valid only for geocaches and is heavily influenced by the Groundspeak cache types. Examples would include "Event cache" and "Multi-Cache".

example:

```
GEOCACHE_TYPE,"","%s"
```

GEOCACHE_PLACER

GEOCACHE_PLACER is a string containing the name of the placer of a geocache.

example:

```
GEOCACHE_PLACER,"","%s"
```

GEOCACHE_ISAVAILABLE

GEOCACHE_ISAVAILABLE is a string containing "True" or "False" indicating whether a geocache is currently available or not.

example:

```
GEOCACHE_ISAVAILABLE,"","%s"
```

GEOCACHE_ISARCHIVED

GEOCACHE_ISARCHIVED is a string containing "True" or "False" indicating whether a geocache has been archived.

example:

```
GEOCACHE_ISARCHIVED,"","%s"
```

GEOCACHE_LAST_FOUND

A long integer in format YYYYMMDD containing the last time this geocache was found.

example:

```
GEOCACHE_LAST_FOUND,"","%ld"
```

GEOCACHE_HINT

The hint for this geocache. No additional transformation (such as rot13) will be performed on this string.

example:

```
GEOCACHE_HINT,"","%s"
```

PATH_DISTANCE_MILES

PATH_DISTANCE_MILES outputs the total length of the route or track from the start point to the current point, in miles. This and the altitude could be used to create an elevation profile. PATH_DISTANCE_MILES is a DOUBLE PRECISION FLOAT.

PATH_DISTANCE_MILES is not valid as an input field.

PATH_DISTANCE_MILES is only meaningful if the data comes from a track or a route; waypoint data will generate essentially meaningless output.

example:

```
PATH_DISTANCE_MILES,"","%f"
```

PATH_DISTANCE_NAUTICAL_MILES

PATH_DISTANCE_NAUTICAL_MILES is like PATH_DISTANCE_MILES except it outputs the length in nautical miles.

PATH_DISTANCE_KM

PATH_DISTANCE_KM is like PATH_DISTANCE_MILES except it outputs the length in kilometers.

PATH_DISTANCE_METERS

PATH_DISTANCE_METERS is like PATH_DISTANCE_MILES except it outputs the length in meters.

PATH_SPEED

Speed in meters per second. GPSBabel does NOT calculate this data by default; it is read from the input file if present. (If not present, it may be calculated with the track filter.)

example:

```
PATH_SPEED,"","%f"
```

PATH_SPEED_KPH

Like PATH_SPEED but means kilometers per hour.

example:

```
PATH_SPEED_KPH,"","%.1f"
```

PATH_SPEED_MPH

Like PATH_SPEED but means miles per hour.

example:

```
PATH_SPEED_MPH,"","%.1f"
```

PATH_SPEED_KNOTS

Like PATH_SPEED but means knots (nautical).

example:

```
PATH_SPEED_KNOTS,"","%.1f"
```

PATH_COURSE

Course in degrees. GPSBabel does not calculate this data by default; it is read from the input file if present. (If not present, it may be calculated with the track filter.)

example:

```
PATH_COURSE,"","%f"
```

GPS_HDOP / GPS_VDOP / GPS_PDOP

GPS horizontal / vertical / positional dilution of precision parameters. Needs float conversion.

example:

```
GPS_HDOP,"","%f"
```

GPS_SAT

Number of satellites used for determination of the position. Needs integer conversion.

example:

```
GPS_SAT,"","%d"
```

GPS_FIX

Type of fix (see GPX spec or track filter). Needs string conversion.

example:

```
GPS_FIX,"","%s"
```

TRACK_NEW

If '1', it indicates that this trackpoint is the first point of a new track. Needs integer conversion.

example:

```
IFIELD TRACK_NEW,"","%d"
```

TRACK_NAME

The name of the track currently being operated on. Needs string conversion.

example:

```
TRACK_NAME, "", "%s"
```

ROUTE_NAME

The name of the route currently being operated on. Needs string conversion.

example:

```
ROUTE_NAME, "", "%s"
```

STREET_ADDR

Street address including house number. Notice that this is not used for any geocoding, it's merely textual description associated with a position.

example:

```
STREET_ADDR, "", "%s"
```

CITY

The name of a city. Sometimes part of "Points of Interest". This is simple textual data associated with a position, no geocoding will be done..

example:

```
CITY, "", "%s"
```

COUNTRY

The name of a country associated with a position.

example:

```
COUNTRY, "", "%s"
```

EMAIL

An email address associated with a position.

example:

```
EMAIL, "", "%s"
```

FACILITY

The name of a facility to associate with a position.

example:

```
FACILITY, "", "%s"
```

PHONE_NR

A phone number associated with a position. This is just textual data attached for convenience.

example:

```
PHONE_NR, "", "%s"
```

POSTAL_CODE

A postal code to associate with a position. It is freeform text and is not used by GPSBabel for any geocoding or such.

example:

```
POSTAL_CODE, "", "%s"
```

FILENAME

The name of the input file from where the points were loaded. This field is available only on output.

example:

```
OFIELD FILENAME, "", "%s"
```

FORMAT

The name of the input format from where format the points came. This field is available only on output.

example:

```
OFIELD FORMAT, "", "%s"
```

Examples

Here is one example style file from the GPSBabel source.

```
# gpsbabel XCSV style file
#
# Format: Garmin POI
# Author: Robert Lipe
# Date: 10/07/2005
# Reference: http://forums.groundspeak.com/GC/index.php?showtopic=110641&st=0&#entry1752204
#
DESCRIPTION Garmin POI database
#
#
# FILE LAYOUT DEFINITIIONS:
#
FIELD_DELIMITER COMMA
RECORD_DELIMITER NEWLINE
BADCHARS COMMA
SHORTLEN 24
```

```
#
# INDIVIDUAL DATA FIELDS, IN ORDER OF APPEARANCE:
#
IFIELD LON_HUMAN_READABLE, "", "%08.5f"
IFIELD LAT_HUMAN_READABLE, "", "%08.5f"
IFIELD SHORTNAME, "", "%s"
IFIELD DESCRIPTION, "", "%s"

OFIELD LON_DECIMAL, "", "%08.5f"
OFIELD LAT_DECIMAL, "", "%08.5f"
OFIELD SHORTNAME, "", "%-.24s"
OFIELD GEOCACHE_TYPE, "", " %-.4s", "no_delim_before,optional"
OFIELD GEOCACHE_CONTAINER, "", "/%-.4s ", "no_delim_before,optional"
OFIELD GEOCACHE_DIFF, "", "(%3.1f", "no_delim_before,optional"
OFIELD GEOCACHE_TERR, "", "/%3.1f)", "no_delim_before,optional"
OFIELD DESCRIPTION, "", "%-.50s"
```

When used on a Groundspeak Pocket Query, it will output lines that look like:

```
-76.76234,38.39123,GC5370 Loca/Virt (1.0/1.0),Dude.. Wheres my Limo??
-90.42345,38.55234,GCC8B Trad/Regu (2.0/2.0),Sweet Reward
-90.81456,38.62456,GC3091 Trad/Regu (1.5/2.0),Matson Hill
```

that are suitable for Garmin's POI loader.

For additional examples, please see the `*.style` files in the `style/` subdirectory of the GPSBabel source tree or at the online source. [https://github.com/gpsbabel/gpsbabel/tree/master/style] .

Miscellaneous Notes

Default Values

Default values are supported for any output fields that contain pure character data output such as URL and NOTES. Default values are only written on output and are not used to supplement missing input. When using default values your mileage will vary greatly depending on the input formats used to populate waypoint data.

Glossary

Terms that are used in conjunction with GPSBabel.

G

Geocaching — GPS based "paper chase", see `http://en.wikipedia.org/wiki/Geocaching`

I

Itinerary — same as a Route (e.g. used by TomTom)

P

Points of Interest (POI) — a collection of gas stations, post boxes, shops and like.

R

Route — a list of geopoints (often with names) connected in a specific order. Usually a collection of geopoints defining the route you want to pass while traveling, created by PC software, or generated inside a GPS device. They can be composed of existing waypoints, or new "routepoints" might be generated.

T

Track — a collection of geopoints recorded by your GPS device while traveling -- "breadcrumb trails". The order of trackpoints within the track is important. Usually a trackpoint doesn't have a name or comment, but a timestamp. This distinguishes a trackpoint from a waypoint.

W

Waypoints — are geopoints that are not necessarily connected to other points, and their order is unimportant. They can be entered before, while or after you actually visit the place and might have tags like name, comment and the like. Usually used to mark special locations as your home, a hotel or a geocache.