
BASIC-Tiger[®] and GPS

Jens-Erich Lange, Gunther Zielosko

1. GPS – having the total overview

GPS (Global Positioning System) has probably been the most important development concerning orientation on earth, since compass and sextant were invented. GPS enables even nautical laymen to determine their position on the sea, on land or in the air. In the following sections we will learn what GPS is, how a typical GPS receiver works, which data it can show and output and how a combination with BASIC-Tiger[®] could look like.

Of course we will offer you a schematic and an example program. BASIC-Tiger[®] is supposed to receive data from a finished GPS receiver and to “archive” travel data similarly to a logbook. On demand it will be able to simulate a GPS receiver for PC applications which require one – even when GPS reception is not possible (e.g. in the inside of a building). Our simple logbook allows us to reconstruct a journey at a later time.

1.1. Basics of satellite navigation

GPS is a complex system using a number of satellites and ground stations. The system was originally set up for military purposes by the US government and was made available for civil purposes later – with the restriction of “artificial” imprecision (SA, Selective Availability). Several commercial applications arose, which are now common in modern traffic (navigation systems for ships, planes, cars etc.). On 2 May 2000 the Selective Availability was abolished by President Clinton, which improved the system’s precision to a 5 m (!) resolution. The principle of satellite navigation is as follows:

A number of specific satellites orbit earth in space. Some are always positioned to enable a GPS receiver to record their signals at a given location. These signals contain information about the individual satellite at a specific time and location. From this information the location of the GPS receiver is ascertained by complex computations. Basically the location survey takes place via the signal propagation delay of the single satellites, which simultaneously transmit their positions (or rather a timetable, because all happens very dynamically!). By measuring the signal propagation delay the distance from the particular satellite is ascertained considering speed of light. Taking only one satellite this will result in a sphere with a radius which equals the measured distance. Our location has to be somewhere on the imaginary sphere’s surface. If two satellites are surveyed, there are two spheres, so our location can only be on an intersection of the spheres (i.e. a circle). With three satellites only two possible intersections are left. Only with four satellites we are left with only one intersection, the exact location including altitude above sea level.

GPS is ingenious because it allows surveying all satellite data, including the orbital deviation, via ground stations. Those corrections are transmitted to the satellites and they send back data with their “updated” location. A GPS receiver contains a complex reception technology as well as a very complex arithmetic unit which converts data into useful coordinates, dates and

altitudes. The following information shows us how difficult the problem of data reception from space is:

24 satellites in total (including reserve satellites even 27) orbit earth with ca. 11,200 km/h about 20,200 km above sea level. Transmitting power is only 20 – 50 W at a transmission frequency of 1,575.42 MHz. At this frequency we have nearly optical conditions, i.e. every building, every vehicle's roof or foliage shields signals almost completely. Electrical interferences from numerous radio transmission services (e.g. mobile phones etc.) handicap reception in addition. Nevertheless modern GPS receivers receive signals from up to 12 satellites at the same time (!). For an exact location at least 3 – 4 satellites are required.

From the given information we learn that even the miraculous GPS has a limit and that you cannot be sure that location information is correct everywhere and always. The most frequent sources of error are:

- The abolished Selective Availability can go into effect again, especially in times of crisis. It is even possible that GPS is deactivated completely occasionally
- Satellites can be shielded by vehicle roofs, foliages etc.
- Changes in signal propagation delay can occur because of different conditions in the ionosphere
- Signal reflections on buildings or mountains can cause longer propagation delays
- Deviations in time (receiver – satellite)
- Deviations of satellite orbits (occur from time to time despite of corrections from the ground)
- Signals are received from too few satellites
- Adversarial satellite constellation (many directly above the receiver, few at the horizon)

1.2. GPS Receivers

The number of devices available has become vast. We will try to systematize them and thereby get to our actual application:

Complete installed navigation systems

You will know such devices from upper class cars. They are usually installed into the dashboard and display maps and the vehicle's position. Mostly there is an audio output which leads you to the destination entered beforehand. There is not much left to do for handicrafts. These devices are perfectly suitable for their application purpose, but also very expensive. The data needed for navigation are provided by a CD, which can be exchanged if required. Similar systems also exist for ships and small airplanes.

Navigation car radios

This is a similar system, only with a slightly cut back map function. Costs are more bearable, but it is no field of activity for handies.

Handheld navigation systems

A technology based on the first group of systems, but can be used outside the vehicle as well. Maps are usually loaded on small memory cards. A small TFT monitor is used as a display. Very comfortable, but also quite expensive.

GPS mobile phones with map functions

These are small, manageable GPS receivers looking like a mobile phone with similar performance characteristics like the previous group. Map functions exist, but mostly have a strongly reduced detail (e.g. often no colored display, low pixel number).

GPS mobile phones without map functions

Currently this is the cheapest version. These devices of course contain a GPS receiver as well as the appending computing technology. A map cannot be saved and therefore cannot be displayed. As some more complex devices they also have data interfaces, which are able to transmit GPS data to external devices (PC, handheld computers, but also BASIC-Tiger®!). It contains a simple display which depicts the device's state, satellite reception conditions, location, speed, altitude and driven or walked distances. You can save landmarks, give routes and record tracks (i.e. to save the trail of a hike). Those data can be recalled on the PC later and can be entered into a map. Such a simple device is for example the Garmin eTrex, which was used by the author and will be presented here.

All groups of devices mentioned above are already quite useful without periphery and can be used only for navigation.

GPS mouses

There are also devices, which only contain antenna, receiver, computing system and interface. They are not useful all alone; they even do not have a simple display in most cases. Only in interaction with a PC or the BASIC-Tiger® they become useful.

GPS modules

These are fitted devices which can be integrated into self-made constructions. They are usually miniature models which require an external GPS antenna and which are connected to other components (PC, micro controllers et.) via a plug adapter. We have tested such a device and adapted hardware to it. The price of these GPS mouse or modules are similar to those of GPS mobile phones without map functions, they are usually even more expensive – a reason for buying a much more universal PGS mobile phone, which you can use without a periphery.

GPS receivers as PC cards

This special type of GPS receiver is available as an accessory for laptop computers or handheld PC. Used together with these devices they are very interesting but not suitable for our purposes because of complex interfaces.

1.3. The Garmin eTrex

Garmin is one of the big producers of GPS receivers of all kinds. Take a look on

<http://www.garmin.com>

An excellent overview of current devices can be seen on

<http://www.mobile-navigation.de/shop/enter.html>

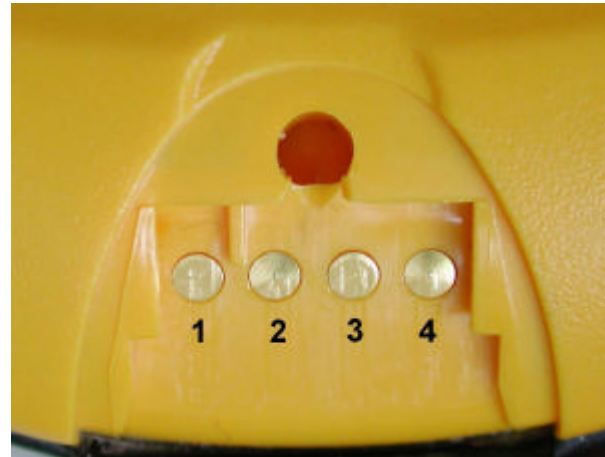
The eTrex family has a number of members, starting from the low-end model eTrex, via the eTrex Summit (additional barometric altimeter and electronic compass), the eTrex Venture (includes city data bank) and the eTrex Legend (includes map representation). We will be fine with the simplest model in its typical yellow case (figure 1), because it also contains the specific Garmin interface (figures 2 and 3).

We will not be able to depict all details of this small high-tech device within this application note. You can get further information on several websites and from a specialised dealer.

The hardware and software of the data interface will be important for us later on. The eTrex contains a RS232 (i.e. levels from -3 ... -12 V and +3 ... +12 V).



Fig. 1 Garmin eTrex



*Fig. 2 Garmin Connector:
1 = Power (+), 2 = Data In
3 = Data Out, 4 = Ground (-)*



Fig. 3 Garmin Connector, cable side

An optionally available data cable is the best way to transmit data from the eTrex to the PC respectively our tachograph and back. This saves us a lot of tinkering with the specific Garmin plug. Both solderable plug and finished eTrex/PC data cable are available on

<http://www.haid-services.de/pi1013036774.htm>

<http://www.haid-services.de/pd-1149209411.htm>

Our device was designed for this kind of connection; it has a 9 pole SUB D plug for directly connecting the data cable. Using the specific Garmin plug you can produce your own connection which also takes over the eTrex power supply in the vehicle. On the software side you can set different GPS protocols. We will use the NMEA protocol. This protocol is known by almost every GPS receiver as well as almost every periphery device respectively software.

1.4. The GN79N-NF (5V, BNC)

This fitted module made by Furuno requires an external antenna for operating. All connections (power, data, and buffer battery) take place via a miniature plug connector. The module and more information are available on:

<http://www.hy-line.de/communication/>

In our project there is a so-called GPS connector which contains the required connections in the right order.



*Fig. 4 GPS module Furuno GN79N
(Photo by Furuno)*



*Fig. 5 Appending GPS antenna AU-12-5B
(Photo by HY-LINE)*

2. Our project – an automatic itinerary

A modern GPS receiver as the eTrex is already able to memorise landmarks and tracks automatically, albeit with restrictions. So why do we need a BASIC-Tiger[®]? The answer is simple: We are challenged by DIY, the free access to programming and the comparatively enormous available memory of BASIC-Tigers[®]. For the first time we can decide which data our GPS device displays or saves, how they are outputted and what task we would like to deal with in general. Once we implement and understand the presented project, there is nothing to bar further applications using GPS and BASIC-Tiger[®]. Because of the comparatively high current consumption we will not keep secret that only applications in a vehicle seem to make sense. But this is the area where a tachograph is useful.

2.1. The circuitry

The effort can be kept within reasonable limits. We use the smallest and cheapest Tiger version, the ECONO-Tiger[®], a MAX232, a voltage regulator 7806 (yes – 6 V!), a specific Tokin GoldCap capacitor FE0H105Z and a few more bits and pieces. Figure 6 depicts a simply designed circuitry which can even be implemented on a breadboard universal PCB.

We added the layout of a simple single-sided PCB to this application note which can be copied if required and can be permuted photo technically to a finished PCB (figures 7 and 8).

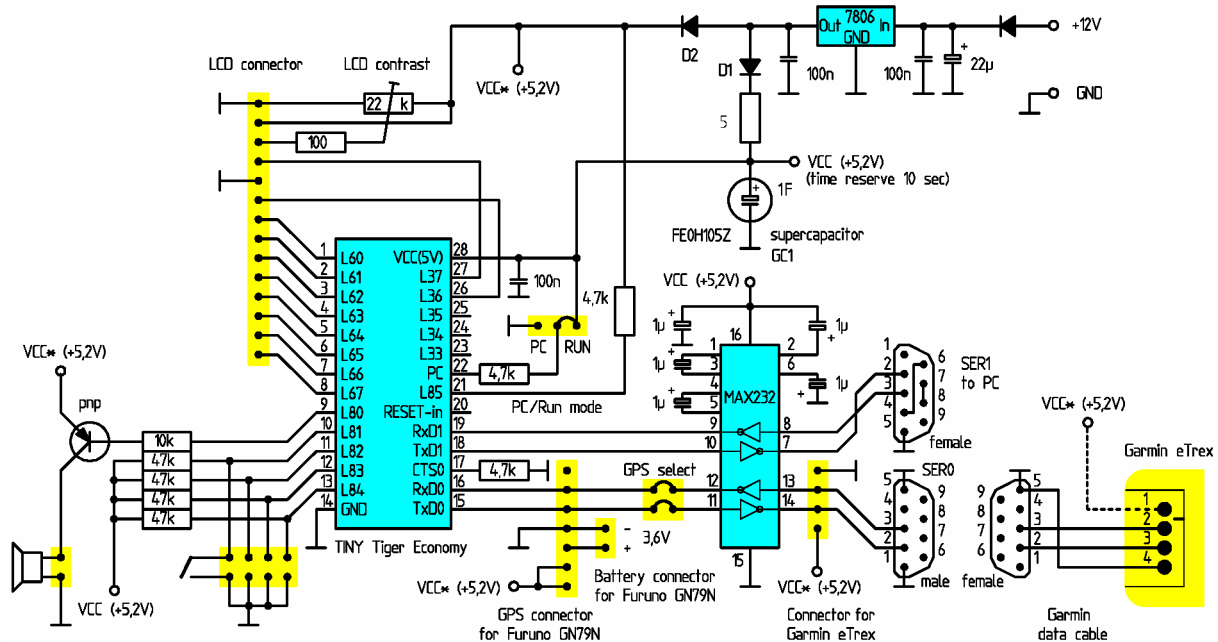


Fig. 6 GPS tachograph application circuitry

Function:

The circuitry requires +12 V and ground from the vehicle’s power supply, the plus pole has to be connected behind the ignition switch. The 7806 regulator is premium quality and delivers 6 V when ignition is activated. The circuitry is supplied with Vcc voltage via diode D1, the eTrex could also work with this voltage (see eTrex terminal assignment figures 2 and 3). But it runs on two mignon cells quite a long time, so the internal supply, e.g. with NiMH rechargeable batteries, works quite well. Diode D1 reduces this 6 V to typical 5.25 V, which suits the ECONO-Tiger[®] and its components. The GoldCap capacitor GC1 with 1 F (1 farad!) is intriguing. It is supposed to ensure that, after deactivating the ignition, the ECONO-Tiger[®] can continue working for some time (i.e. save data to flash). Please mind a low internal resistance of about 1 W when choosing this capacitor. Diode D2 delivers Vcc* which supplies the MAX 232 and, if necessary, the GPS receiver as well as the active beeper and the input jumpers. All components are deactivated as soon as the ignition is switched off, in order to save “reserve” current. In addition Vcc* delivers a logical high level to L85 of the ECONO-Tiger[®] which will by that know at once that the ignition is switched off. The MAX 232 is on the right side of the circuitry. It is responsible for the right RS232 level at both serial interfaces. The upper SER1 is used as usual for possibly programming the ECONO-Tigers[®] “on board” and for inquiring data from the tachograph. SER0 (beneath) is used as an interface to the eTrex (RS232) or to a device with TLL levels (GPS mouse or the like). When using eTrex both GPS-Select jumpers are placed in a way that both lines RxD0 and TxD0 are lead through the MAX232 to the 9 pole SUB-D plug. Then you can simply use the standard data

cable from the eTrex for connection. If your GPS receiver works with TTL levels, leave out the jumpers and connect the latter to the corresponding pins of the GPS connector.

ECONO-Tiger[®] port 6 as well as some control lines are used for an optional display. Here the user can watch some current data on his 4 line LC display (version and connection as in the Plug and Play system) using the sample program "GPS-REC.TIG". You see some more control elements at bottom left, such as a button, a special beeper and three jumpers, which can be used for settings. In the first instance you do not have to connect anything there.

Getting hold of a Tokin Super Capacitor FE0H105Z is a bit difficult. Possible distributors are

Rutronik <http://www.rutronik.de/>
Gleichmann <http://www.msc-ge.com/>

2.2. Board layout

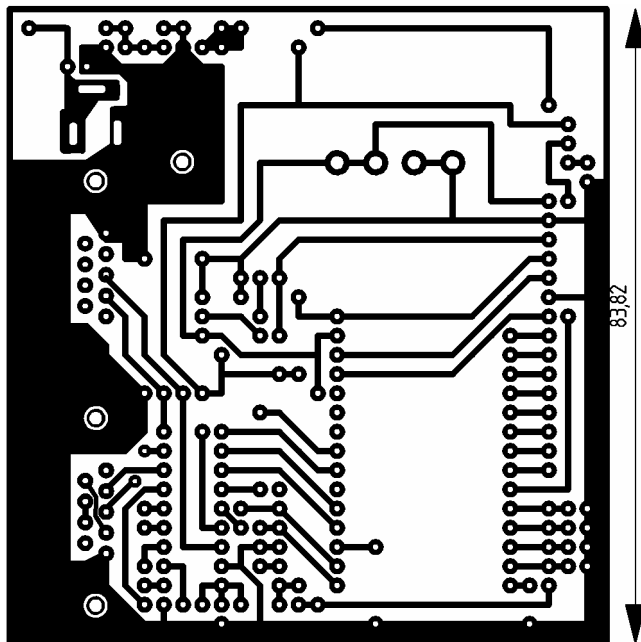


Fig. 7 Layout of the single-sided board

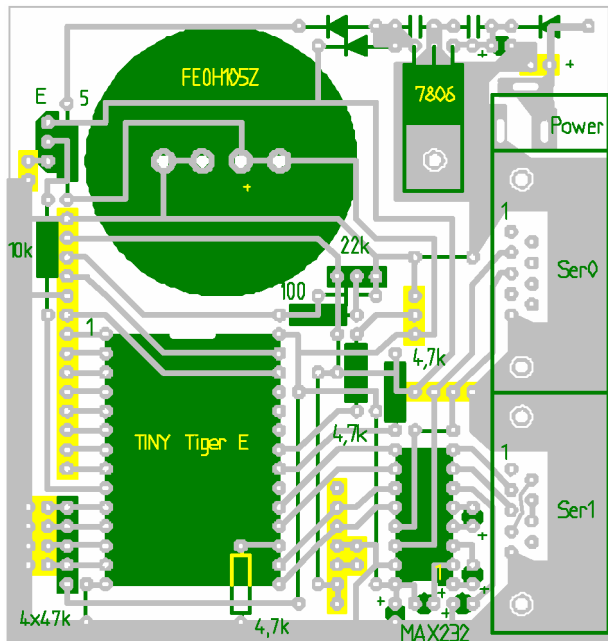


Fig. 8 Board's component side

3. NMEA protocol

For all of you who would like to take a closer look at the GPS world, we would like to present the NMEA 0183 protocol (National Marine Electronics Association). This complex standard does not only control the data traffic of GPS devices, but also of all important navigation devices in shipping (such as radar, weather data etc.). Further information is available on

<http://www.nmea.de/index.html>

Almost every GPS receiver with a communication interface knows at least this kind of data transfer. We will examine this data sequence instancing the eTrex.

Garmin eTrex operates with NMEA 0183 version 2.0 and transmits standard records: GPRMC, GPRMB, GPGGA, GPGSA, GPGSV, GPGLL, GPBOD and GPRTE. Our own Garmin records are added: PGRME (approximate error), PGRMZ (altitude). Every two seconds etrex delivers a data block with the following structure (location: Erfurt, Germany):

```
$GPRMC,125326,A,5057.1284,N,01102.9801,E,0.0,303.1,271002,1.0,E,A*16
$GPRMB,A,,,,,,,,,A,A*0B
$GPGGA,125326,5057.1284,N,01102.9801,E,1.04,1.9,262.5,M,46.6,M,,*42
$GPGSA,A,3,05,07,09,,,26,,,,,2.7,1.9,1.0*31
$GPGSV,2,1,08,05,28,234,50,07,43,085,33,09,63,289,45,18,25,273,00*76
$GPGSV,2,2,08,23,05,212,00,26,45,171,48,28,19,056,00,29,33,166,00*71
$GPGLL,5057.1284,N,01102.9801,E,125326,A,A*4C
$GPBOD,,T,,M,,*47
$PGRME,7.8,M,14.1,M,16.7,M*25
$PGRMZ,861,f,3*14
$GPRTE,1,1,c,*37
```

The next record arrives two seconds later:

```
$GPRMC,125328,A,5057.1284,N,01102.9801,E,0.0,303.1,271002,1.0,E,A*18
$GPRMB,A,,,,,,,,,,,,,A,A*0B
$GPGGA,125328,5057.1284,N,01102.9801,E,1,04,1.9,262.5,M,46.6,M,,*4C
$GPGSA,A,3,05,07,09,,,26,,,,,,,,,2.7,1.9,1.0*31
$GPGSV,2,1,08,05,28,234,50,07,43,085,33,09,63,289,45,18,25,273,00*76
$GPGSV,2,2,08,23,05,212,00,26,45,171,48,28,19,056,00,29,33,166,00*71
$GPGLL,5057.1284,N,01102.9801,E,125328,A,A*42
$GPBOD,,T,,M,,*47
$PGRME,7.8,M,14.1,M,16.7,M*25
$PGRMZ,861,f,3*14
$GPRTE,1,1,c,*37
```

We realise that all “real” NMEA lines begin with \$GP, which is a general identification for GPS devices. This is followed by three more characters, which characterise the type of message. Then real data follow, separated by a comma. The two last lines but one (\$PG...) are Garmin specific and differ concerning their identification. Let’s examine the above record:

```
$GPRMC,125326,A,5057.1284,N,01102.9801,E,0.0,303.1,271002,1.0,E,A*16
```

RMC - Recommended Minimum Navigation Information

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

```
$--RMC,hhmmss.ss,A,llll.ll,a,yyyy.yy,a,x.x,x.x,xxxx,x.x,a*hh<CR><LF>
```

- Field Number:
- 1) UTC Time (12:53:26)
 - 2) Status, V = Navigation receiver warning, A = o.k. (A)
 - 3) Latitude (5057.1284)
 - 4) N or S (N)
 - 5) Longitude (01102.9801)
 - 6) E or W (E)
 - 7) Speed over ground, (0.0) knots
 - 8) Track made good, degrees true (303.1)
 - 9) Date, ddmmyy (27.10.02)
 - 10) Magnetic Variation, degrees (1.0)
 - 11) E or W (E)
 - 12) Checksum (16)

```
$GPRMB,A,,,,,,,,,,,,,A,A*0B
```

RMB - Recommended Minimum Navigation Information

	1 2		3 4		5		6		7 8		9 10		11		12		13		14

```
$--RMB,A,x.x,a,c--c,c--c,llll.ll,a,yyyy.yy,a,x.x,x.x,x.x,A*hh<CR><LF>
```

- Field Number:
- 1) Status, V = Navigation receiver warning
 - 2) Cross Track error - nautical miles
 - 3) Direction to Steer, Left or Right
 - 4) TO Waypoint ID
 - 5) FROM Waypoint ID
 - 6) Destination Waypoint Latitude
 - 7) N or S
 - 8) Destination Waypoint Longitude
 - 9) E or W
 - 10) Range to destination in nautical miles
 - 11) Bearing to destination in degrees True
 - 12) Destination closing velocity in knots
 - 13) Arrival Status, A = Arrival Circle Entered
 - 14) Checksum (0B)

\$GPGGA,125326,5057.1284,N,01102.9801,E,1,04,1.9,262.5,M,46.6,M,,*42

GGA - Global Positioning System Fix Data, Time, Position and fix related data for a GPS receiver.

```

      1         2         3 4         5 6 7 8         9 10 | 11 12 13 14 15
      |         |         | |         | | | |         | | | | | | | |
$--GGA,hhmmss.ss,llll.ll,a,yyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh<CR><LF>
  
```

Field Number:

- 1) Universal Time Coordinated (UTC) (12:53:26)
- 2) Latitude (5057.1284)
- 3) N or S (North or South) (N)
- 4) Longitude (01102.9801)
- 5) E or W (East or West) (E)
- 6) GPS Quality Indicator, (1)
 - 0 - fix not available,
 - 1 - GPS fix,
 - 2 - Differential GPS fix
- 7) Number of satellites in view, 00 - 12 (04)
- 8) Horizontal Dilution of precision (1.9)
- 9) Antenna Altitude above/below mean-sea-level (geoid) (262.5)
- 10) Units of antenna altitude, meters (m)
- 11) Geoidal separation, the difference between the WGS-84 earth (46.6) ellipsoid and mean-sea-level (geoid), "-" means mean-sea-level below ellipsoid
- 12) Units of geoidal separation, meters (m)
- 13) Age of differential GPS data, time in seconds since last SC104 type 1 or 9 update, null field when DGPS is not used
- 14) Differential reference station ID, 0000-1023
- 15) Checksum (42)

\$GPGSA,A,3,05,07,09,,,26,,,,,2.7,1.9,1.0*31

GSA - GPS DOP and active satellites

```

      1 2 3         14 15 16 17 18
      | | |         | | | | |
$--GSA,a,a,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x,x*hh<CR><LF>
  
```

Field Number:

- 1) Selection mode
- 2) Mode
- 3) ID of 1st satellite used for fix (05)
- 4) ID of 2nd satellite used for fix (07)
- ...
- 14) ID of 12th satellite used for fix
- 15) PDOP in meters (2.7)
- 16) HDOP in meters (1.9)
- 17) VDOP in meters (1.0)
- 18) checksum (31)

\$GPGSV,2,1,08,05,28,234,50,07,43,085,33,09,63,289,45,18,25,273,00*76

\$GPGSV,2,2,08,23,05,212,00,26,45,171,48,28,19,056,00,29,33,166,00*71

GSV - Satellites in view

```

      1 2 3 4 5 6 7         n
      | | | | | |         |
$--GSV,x,x,x,x,x,x,x,...*hh<CR><LF>
  
```

Field Number:

- 1) total number of messages (2)
- 2) message number (1 or 2)
- 3) satellites in view (08)
- 4) satellite number (05) 1st satellite
- 5) elevation in degrees (28)
- 6) azimuth in degrees to true (234)
- 7) SNR in dB (50)
- ...

more satellite infos like 4)-7) (07,43,085,33 etc.)
...
n) checksum (76 or 71)

`$GPGLL,5057.1284,N,01102.9801,E,125328,A,A*42`

GLL - Geographic Position - Latitude/Longitude

1	2	3	4	5	6	7

`$--GLL,llll.ll,a,yyyy.yy,a,hmmss.ss,A*hh<CR><LF>`

Field Number:

- 1) Latitude (5057.1284)
- 2) N or S (North or South) (N)
- 3) Longitude (01102.9801)
- 4) E or W (East or West) (E)
- 5) Universal Time Coordinated (UTC) (12:53:28)
- 6) Status A - Data Valid, V - Data Invalid
- 7) Checksum (42)

`$GPBOD,,T,,M,,*47`

BOD - Bearing - Waypoint to Waypoint

1	2	3	4	5	6	7

`$--BOD,x.x,T,x.x,M,c--c,c--c*hh<CR><LF>`

Field Number:

- 1) Bearing Degrees, TRUE
- 2) T = True (T)
- 3) Bearing Degrees, Magnetic
- 4) M = Magnetic (M)
- 5) TO Waypoint
- 6) FROM Waypoint
- 7) Checksum (47)

`$PGRME,7.8,M,14.1,M,16.7,M*25`

RME - Estimated Position Error

Field Number:

- 1) Estimated horizontal position error (HPE)(7.8)
- 2) Unit, meters (m)
- 3) Estimated vertical position error (HVE) (14.1)
- 4) Unit, meters (m)
- 5) Overall spherical equivalent position error (16.7)
- 6) Unit, meters (m)
- 7) Checksum (25)

`$PGRMZ,861,f,3*14`

RMZ - Altitude information

Field Number:

- 1) Altitude (861)
- 2) Unit (f, feet)
- 3) Position fix dimensions: 2=user / 3=GPS
- 7) Checksum (14)

`$GPRTE,1,1,c,*37`

RTE - Routes

1	2	3	4	5	x	n

`$--RTE,x.x,x.x,a,c--c,c--c,.....c--c*hh<CR><LF>`

Field Number:

- 1) Total number of messages being transmitted (1)
- 2) Message Number (1)

- 3) Message mode (c)
 - c = complete route, all waypoints
 - w = working route, the waypoint you just left, the waypoint you're heading to then all the rest
- 4) Waypoint ID
- x) More Waypoints
- n) Checksum (37)

4. eTrex text mode

If you already own a Garmin eTrex, the much simpler text mode for data transmission and evaluation might be interesting for you. Here the amount of data is reduced clearly. All data sequences have the same length and every record has the same structure. This is perfect for evaluating using a control computer such as BASIC-Tiger® (very simple string modification). Here's another example:

```
@000607204655N6012249E01107556S015+00130E0021N0018U0000
```

```
@yyymmddhhmmss Latitude Longitude error Altitude EWSpd NSSpd VSpd
```

Every entry has a fixed length, which is only identified by its position in the string. Remarkably speed in x, y, and z direction is outputted in addition to location (x, y, z) – balloon drivers will appreciate this! The record is interpreted as follows:

```
@000607204655N6012249E01107556S015+00130E0021N0018U0000
```

@		every record begins with @
000607		UTC date (07 June 2000)
204655		Time (20:46:55)
N6012249		Latitude (N6012.249)
E01107556		Longitude (E01107.556)
S		Location state* (S)
015		Horiz. location error (015)
+		Altitude sign (+)
00130		Altitude in m (00130)
E		Speed direction East (E)
0021		Amount/10 (m/s)(002.1)
N		Speed direction North (N)
0018		Amount/10 (m/s) (001.8)
U		Speed direction up (U)
0000		Amount/100 (m/s)(00.00)

Position latitude:
Always decimal symbol after 4 digits
Position longitude:
Always decimal symbol after 5 digits
Possible characters location and speed:
N, S, E, W (North, South, East, West)
Possible characters altitude:
+, - (above/under sea level)
Possible characters vertical movement:
U, D (up, down)
Position state:
d (2D differential GPS position)
D (3D differential GPS position)
g (2D GPS position)
G (3D GPS position)
S (simulated position)
- (invalid position)

All values automatically get leading zeros until the defined length is reached.

5. The program's functionality

The BASIC-Tiger[®] program "GPS-REC.TIG" contains three tasks: Main task "Main", which initialises the drivers first and then scans port 8 in a loop, and two further tasks which combine the received characters from both serial interfaces SER0 and SER1 to strings.

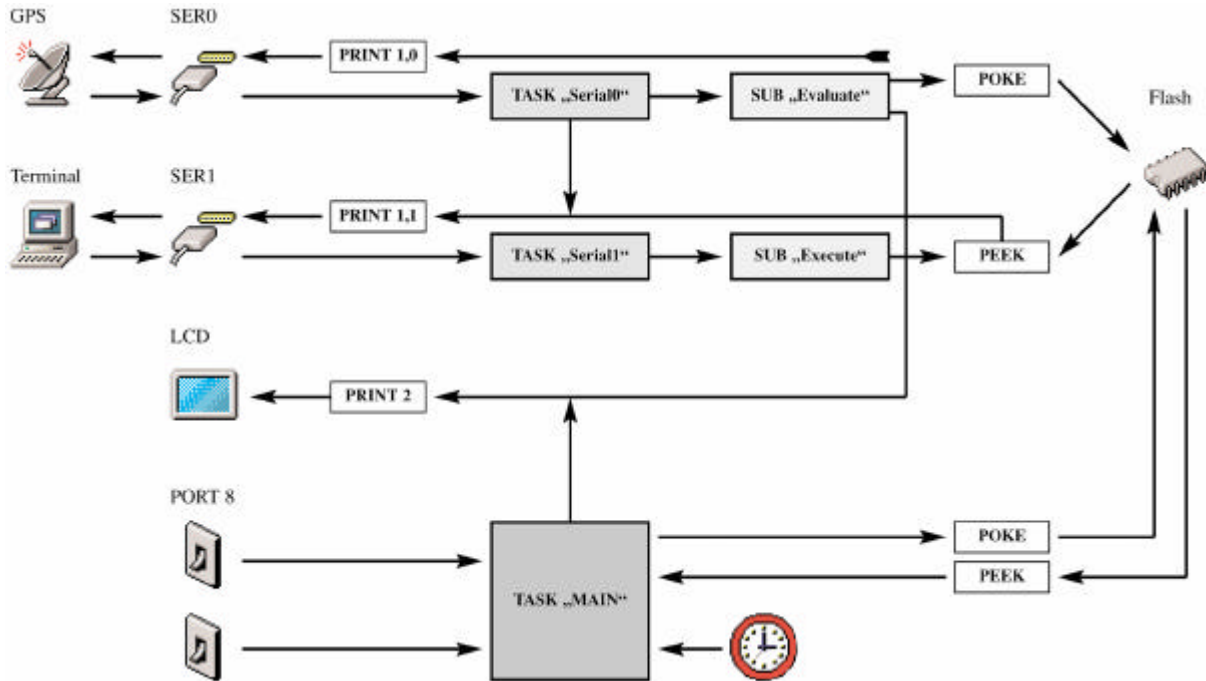


Fig. 9 Schematic program flow chart

When task "Serial0" receives a control character, it is recognised as a record's ending. The string which has been saved so far contains a NMEA record from the GPS receiver, which is processed in subroutine "Evaluate". In addition the received telegram is sent again via the other serial interface, if the program is in the "NMEA GPS" mode. While being in this mode a terminal connected to the interface "SER1" can receive the data seemingly coming directly from the receiver.

Task "Serial1" works similarly. Here characters, which are transmitted from the connected terminal at "SER1", are gathered. When a record is complete it is processed as a command by subroutine "Execute". For every valid command an answer is sent via the same interface. To avoid an unfavourable intermixture with NMEA telegrams from the GPS receiver, there are two operating modes: In the "NMEA GPS" mode, as described above, telegrams from the receiver are handed over. In the "command" mode commands coming from the terminal are processed and answered.

Switching operating modes takes place by opening and closing the switch at pin 10 (Port 8.1). In addition the end of a trip (deactivation of ignition) can be detected if a 1F Super-Cap is installed into the circuit. In this case BASIC-Tiger[®] evaluates the signal at pin 21 (Port 8.5). If the capacitor is missing, the departure time is memorised instead of arrival time. It is the

BASIC-Tiger® program's main task to file the departure and arrival and as many intermediate steps as possible in the Tiger's flash memory. If the program is set to command mode, these data can be read out via the serial interface later.

5.1. Memory stacks

Writing to and reading from the flash memory takes place in records, which are divided into a header and a body. The header always consists of one single byte and is used to identify the following data.

5.2. Data types

To determine the data type only the first 3 bits from the header are evaluated:

Type	Meaning	Bytes following
0	departure	18 bytes
1	arrival	14 bytes
2	compressed step	1 byte
3	short step	3 bytes
4	long step	6 bytes
5	full step	12 bytes
6	DEBUG	4 bytes
7	not defined	---

If bit 7 is set in the header (i.e. any value > 127), it is an invalid value. This way the program recognises the unused flash memory, since all bytes are set to 11111111b when erasing. The following bytes in the body can have arbitrary values.

5.3. Data content

Locations, date/time, driving time and distance are saved as data content. A position consists of 2 LONG, which depict latitude and longitude as a decimal degree times 10,000. So specifications contain the value in degree as well as 4 post decimal positions.

Date and time are given as seconds since 1 January 1980 0:00 in a LONG. This format is common and has some advantages when computing. To convert this value to e.g. the Microsoft Windows floating point number for time display, you can use the following formula: (seconds / 86400) + 29221.

The driving time is saved as WORD and contains the seconds gone by since departure. So time is not allowed to exceed 18 hours. Distance is saved as WORD and contains the driven hectometres (kilometres*10). The maximum value amounts 6553 km for each ride.

Data types 0 and 1 additionally save the so-called TTFF value (Time to First Fix). This value shows how long the reception of the first valid coordinate has taken after departure.

Data types 0, 1 and 5 contain the absolute value while types 2, 3 and 4 contain the difference to the previous value, in order to be saved as briefly as possible. Remarkable is data type 2. It allows saving a change of coordinate including longitude and latitude as well as the seconds driven in one single byte!

5.4. Data type 2

Saving as data type 2 requires the following: The changes of coordinates must not be higher than +/- 1 (longitude and latitude) and seconds gone by must not amount to more than 31. Saving a change of coordinates actually requires only 3 bits; the remaining 5 bits are available for saving the time. The change of coordinates is coded as follows: The difference to the previous coordinate can always be depicted in a 3 by 3 matrix, the previous coordinate always being the centre of the matrix. Since there are now 8 out of 9 fields remaining for possible changes of direction, these 8 possible changes of directions can be coded in 3 bits. Changes of direction from the previous to the current coordinate are defined as follows:

0	North
1	North-East
2	East
3	South-East
4	South
5	South-West
6	West
7	North-West

5.5. Data type 3

Saving as data type 3 requires the following: Change of coordinates does not amount to more than +/- 255 (longitude and latitude) and the seconds gone by do not amount to more than 63. In the first byte the absolute amount of difference in latitude is saved. In the second byte the absolute amount of difference in longitude is saved. In the third byte the difference in time is saved in bits 0-5. In bit 7 the difference in latitude sign is saved, bit 6 saves the sign of the difference in longitude.

5.6. Data type 4

Saving as data type 4 has the following requirements: The change of coordinates does not amount to more than +/- 32767 in latitude and longitude. In the first word the difference in latitude plus 32767 is saved. In the second word the difference in longitude plus 32767 is saved. In the third word the difference in time is saved.

5.7. Commands

For implementing a simple command structure, the subroutine “Execute” always evaluates only the first character and executes the corresponding command. All answers to these commands begin with an at-sign (@), so that the evaluation software easily knows if an answer was given.

A star (*) followed by a checksum in hexadecimal code is appended to the answers to commands “?”, “.”, “:” and “;”. The calculation of the checksum corresponds to the procedure according to NMES, so that we can use the same check handler here.

If the check fails the command “?” will repeat the same request. The commands “.”, “:” and “;” will repeat their request with the command “;”. These commands output the data’s position in memory as the last parameter before the check sum. If this information is used with the commands “.” or “:” downloading can be continued step by step.

5.7.1. Version inquiry ?

Inquiry of version, amount of records, amount of journeys, memory address, total memory. The reading address for commands “.” or “:” is reset.

[@3.1,573,7,2219,393216*2E](#)

Single parameters between “@” and “*” are separated by commas. The following parameters are outputted:

- Version firmware
- Saved records
- Thereof saved journeys
- Used flash memory
- Total flash memory

5.7.2. Read record .{address}

The next record is read. In case there is a number given after the point, the record at this address is read (optionally).

[@2,21,91*39](#)

The single parameters between “@” and “*” are separated by commas. Depending on the record type the following data are outputted:

- | | |
|---|--|
| 0 | Longitude, latitude, time, distance, driving time, TTFFF |
| 1 | Longitude, latitude, time, TTFF |
| 2 | Delta latitude + delta longitude + delta driving time |
| 3 | Delta latitude, delta longitude, delta driving time |
| 4 | Delta latitude, delta longitude, delta driving time |

- 5 Latitude, longitude, time
- 6 Arbitrary LONG value

When no further data exist the following output is given:

@EOD*4E

5.7.3. Read record **:{address}**

The next record is read. In contrast to the command “.” only the next end of journey is transmitted. If a number is given after the colon, the record at this address is read (optionally).

@0,539414,103073,739738056,184,1370,8,1131*24

The single parameters between “@” and “*” are separated by commas. Depending on the record type the following data are outputted:

- 0 Latitude, longitude, time, distance, driving time, TTFF
- 1 Latitude, longitude, time, TTFF

When no further data exist the following output is given:

@EOD*4E

5.7.4. Repeat inquiry **;**

Repeat last point command (“.” or “.“). Makes sense when the checksum is wrong.

@0,539414,103073,739738056,184,1370,8,1131*24

5.7.5. Save record **,**

Save record. A previously saved record can be written back. This happens by indicating the record as it was received, leaving out the “@” and adding one comma in front and two commas behind.

@Next Flash address: xxxx

5.7.6. Transmit to receiver **\$**

NMEA record is transmitted to the receiver. The record is transmitted unaltered to the receiver.

5.7.7. Delete memory **!**

Delete Flash memory. For every deleted memory bank there's a feedback:

... @FLASH ERASED!

5.7.8. Inquire state of device -

Using this command you can check if the device is connected and if it is in command mode.

@OK

5.8. Details concerning the BASIC-Tiger® program

The source code "GPS-REC.TIG" contains the following highlights:

- Temporally independent reception of strings at a serial interface in a separate task --> "TASK Serial0" and "TASK Serial1"
- Changing the baud rate at runtime --> "SUB CheckInputs"
- Calculate distance of two coordinates --> "SUB GetDistance"
- Efficient conversion of date/time to seconds since 1 January 1980 --> "SUB GpsTime". The routine presented in this application note is more compact than the original Wilke example in file "Timecvt.tig"
- Calculating a checksum following the NMEA standard --> "SUB AddNmea"
- Testing a checksum following the NMEA standard --> "SUB NmeaCheck"
- Coding a change of coordinates using only 3 bits --> "SUB WriteFlash"

We hope we gave you an understanding of the interesting GPS world. This simple application presented here can be modified and extended in many ways. A "black box" version of this logbook can be found on

<http://www.dynamo-software.de>

By saving the requested data on a SmartMedia Card instead using the Tiger Flash, a much higher amount of data can be recorded. Besides evaluation becomes easier – take out the SM card after the ride and read it via a PC card reader. You will be able to examine all details of the ride, hike or balloon ride at once. Think of further applications – your vehicle's GPS receiver transmits data to BASIC-Tiger® when your car is used unauthorised. BASIC-Tiger® then sends a text message to your mobile phone which informs you or a security service about the current location of your car.