Version 0.2

TomTom Watch Interface

Version 0.2

TOMTOM WATCH INTERFACE

INTRODUCTION

The TomTom Watch family (Multisport, Spark, Runner, Adventurer) are GPS watches with a multitude of fitness tracking functions. TomTom still adds new functions, though (07-2017) it appears that TomTom will move out of wearables. The watches are used in combination with the TomTom Mysports cloud account. A local application on PC or Mobile/Tablet is used to communicate to the watch and sync between the watch and the cloud.

In order to create a PC application I continued reverse engineering the watch. The start of this was formed by the excellent application ttwatch of Ryan Binns. It has been applied in the TomTomWatch Application (http://blog.studioblueplanet.net/?page_id=566).

'not checked' indicates I copied the information from the Ryan Binns application, but did not check myself.

INTERFACE

The watch has two physical interfaces:

- Bluetooth
- USB

This document is limited to the USB interface. Using this interface a client can issue commands to the watch and read, write and delete files. First the file system and formats are described, then the USB interface.

FILE SYSTEM

OVERVIEW

The watch interface functions are based on a file system that is accessible via Bluetooth and the USB interface. Note that another file system is present that is accessible via the PC via USB as regular removable storage and used for transfer of music. This file system is mapped on the PC file system (like an USB stick) when the watch is connected.

This section describes the former filesystem. Files are addressed by an 32 bit integer ID. In this document the file IDs are given in an 8 character hexadecimal, like **0x00910000**.

Version 0.2

Following files have been identified:

File ID	Description
0x00000012	Bluetooth Low Energy (BLE) firmware. Used for firmware upgrades. (NOT CHECKED)
0x000000F0	System firmware. Used for firmware upgrades
0x00010100	GPS Quickfix data, used for obtaining a quick GPS lock. Is written each time the watch is connected to TomTom MySports. For the adventurer, the file is downloaded from https://gpsquickfix.services.tomtom.com/fitness/sifgps.f2p3enc.ee
0x00010200	GPS Firmware. not checked
0x00010301	Some version numbers??
0x00013000	Stacktrace. Text file
0x00013001	BLE firmware Update log not checked
0x00013002	Sysem firmware Update log. Text file
0x00013100	System log. Text file
0x0071xxnn	Races. xx defines the activity. nn is the race number. Proprietary format.
0x0072nnnn	Race history. not checked
0x0073xxnn	History data. For each activity type (xx) for the last 10 activities (0xnn) such a file is generated
0x0081nnnn	Language files
0x008300xx	Activity summary. Each file contains the last 10 activities. This file is used by the watch to show the last 10 activities for each activity type xx.
0x0085000n	Manifest files. Contain the settings. 0x00850000 is the current list of settings. 0x00850001 and 0x00850002 are backups
0x00880000	Textual representation of the playlists (music watches only)
0x0091nnnn	Activity files. Correspond with the ttbin files. Proprietary file format containing records.
0x00B100nn	Tracked activity (steps, calories, heart rate, sleep, fitness points, etc) for upload to TomTom MySports. Up to 20 files are generated. A new file is generated each time the watch is connected to the PC and disconnected. The files are uploaded and deleted when the watch is connected to TomTom MySports. Protobuf format
0x00B20000	Tracked activity, temporary file which is used during the time the watch is connected to the PC. When disconnected, this file is 'renamed' to the next 0x00B100nn file (if there are 20 0x00B100nn files, tracked activity keeps being logged to this file; rename after the the 0x00B100nn files have been deleted; CHECK!). Same protobuf format as 0x00B100nn
0x00B20001	Unclear. 4 bytes.
0x00B3000n	Tracked activity of the last 7 days. n=8-f. Same protobuf format as 0x00b100nn, however heartrates are not stored.
0x00B8000n	Routes (track planning). Each file contains a route. Protobuf format. The watch accepts 15 route files, hence 0x0≤n≤0xE
0x00B9nnnn	
0x00BEnnnn	The personalized workouts, added since firmware version 1.7.53 (Adventurer). Protobuf format.
0x00F20000	Preferences file. XML format. Contains the watch name and other preferences for connecting to TomTom MySports

PREFERENCES FILE, 0X00F20000

The preferences file is an XML file containing the preferences used when connecting to TomTom MySports. After factory reset the watch does not have such a preference file. When connecting the watch to the 'TomTom Sports Connect' it is created as part of the registration process. A.o. a token and secret are generated and coupled to a TomTom Mysports cloud account. When connecting the Watch to 'TomTomWatch' a default preference file is written without token and secret (file will be overwritten by 'TomTom Sports Connect').

Apart from the token and secret it contains the watch name, configuration service URL and some additional configuration. The file is also copied on a Windows PC to:

`c:\users\<user>\TomTom Sports\<device serial>\preferences.dat`

Changing the watch name can be done by changing the name between the `<watchName></watchName>` tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<preferences version="1" modified="Sun Oct 29 17:12:36 2017">
 <ephemerisModified>0</ephemerisModified>
 <watchName>GPS Watch Jorgen</watchName>
 <ConfigURL>https://mysports.tomtom.com/service/config/config.json</ConfigURL>
 <exporters>
 <online>
 <export id="MySports" autoOpen="1"/>
   <MySportsAuthToken>...</MySportsAuthToken>
   <MySportsTokenSecret>...</MySportsTokenSecret>
 </online>
 </exporters>
</preferences>
```

FIRMWARE FILES (0X000000F0, 0X00000012, 0X00010200)

Firmware update steps are:

1. Fetch the configuration service as defined in the preferences file, `<ConfigURL>` tags.
   E.g.
   `<ConfigURL>https://mysports.tomtom.com/service/config/config.json</ConfigURL>`

2. Get the “service:firmware” url from the resulting JSON:
   E.g.
   `https://sports.tomtom-static.com/downloads/firmware/{PRODUCT_ID}/FirmwareVersionConfigV2.xml`

3. Fill in the {PRODUCT_ID} as resulted from the product ID USB function.
   E.g. for the adventrurer:
   `https://sports.tomtom-static.com/downloads/firmware/E0070000/FirmwareVersionConfigV2.xml`

4. Retrieve the xml file. This xml file defines the latest firmware and corresponding firmware files

```
<FirmwareVersion>
  <latestVersion>
    <Major>1</Major>
    <Minor>7</Minor>
    <Build>53</Build>
  </latestVersion>
  <isCritical>yes</isCritical>
  <URL>1_7_53/0x000000F0</URL>
</FirmwareVersion>
```

5. Download the firmware files from the same location
   E.g.
   `https://sports.tomtom-static.com/downloads/firmware/E0070000/1_7_53/0x000000F0`

6. Upload the firmware files to the watch

7. Execute an USB reboot command to the watch.

Note:

- Firmware upgrades usually lead to an extension of the number of settings as stored in the manifest files (0x0085000n).

- I downgraded the firmware, by applying 1.3.255 0x000000F0 file to my watch running 1.6.26. This succeeded, however it resulted in a full reset and a required a reconfiguration. Not clear if downgrading is generally supported.

GPS QUICKFIX FILE (0X00010100)

Procedure for uploading quickfix data:

1. Fetch the configuration service as defined in the preferences file, <ConfigURL> tags.
   E.g.
   `<ConfigURL>https://mysports.tomtom.com/service/config/config.json</ConfigURL>`

2. Get the "service:ephemeris" url from the resulting JSON:
   E.g.
   `https://gpsquickfix.services.tomtom.com/fitness/sifgps.f2p{DAYS}enc.ee`

3. Replace the {DAYS} by the number of days ahead. 3 and 7 seem the only possible values…
   E.g.
   `https://gpsquickfix.services.tomtom.com/fitness/sifgps.f2p3enc.ee`

4. Download the file.

5. Upload the file to the watch using file ID 0x00010100.

MANIFEST FILES (0X0085000N)

These files contain the watch settings as key-value pairs. The number of settings is defined by the Length field. The list is settings is extended usually at software updates.

Field	Description	Bytes	Format
File type	0x0085	2	Integer
Length	Number of tag-value pairs	2	Integer
Array:			
Tag	Tag. Seems to be an increasing number from 0 to (Length-1)	2	Integer
Value	Value	4	Integer

Version 0.2

ACTIVITY FILES (0X0091NNNN)

HANDLING

When using the TomTom Sports Connect application, the 0x0091NNNN files are downloaded and stored on disk. Under windows the path is:

```
c:\users\<user>\TomTom Sports\<watch name>\<YYYY-MM-DD>\<sport>_<hh-mm-ss>.ttbin
```

For example:

```
c:\users\<user>\TomTom Sports\GPS Watch Jorgen\2017-01-01\Freestyle_20-35-58.ttbin
```

When using the Android app the files are stored on the file store:

```
/TomTom_MySports/<serial>/workouts/uploaded/<fileId>_<YYYYMMDD>_<hhmmss>.ttbin
```

For example:

```
/TomTom_MySports/HL1456G01770/workouts/uploaded/00910000_20170101_203558.ttbin
```

(Writing a ttbin file to `/TomTom_MySports/<serial>/workouts/` and starting the TomTom App makes the App upload the ttbin file to the TomTom MySports cloud.

FILE FORMAT

Activity files contain the logged activities. After deleting all **0x0091nnnn** files, the next activity is logged with nnnn=0, subsequent files are logged by increasing nnnn by 1.

Format:

Record 0 – Header
Record 1
Record 2
…
Record N

The ttbin file consist of a series of record. Each record starts with a tag followed by a number of values. The tag identifies the record type and defines the values to follow. A value can consist of 1 or more bytes encoding an integer or float value.

Tag	Value 1	Value 2	Value 3	…	…	Value M

All integers are little endian (LSB first)

The first record in the file is the header record (tag=0x20). This is a special record a.o. defining the records in the file.

HEADER RECORD

It is the first record in the ttbin file. The header defines the ttbin file. Amongst others it defined the record types that occur in the file with their lengths.

Length: 117 (version<=0x09) or 120 bytes (version>=0x0a), excluding the array with tags and lengths.

Field	Description	Bytes	Format
Tag	0x20	1	integer
version	Version of the ttbin file format. Versions of 0x07, 0x09 and 0x0A have been checked.	1	integer
firmware version	Versions of watch firmware, consisting of major, medium, minor, like 1.3.255. In ttbin “version” <=0x09 for each part 1 byte is reserved, “version” >=0x0A two bytes, little endian.	Version <=0x09: 3 >=0x0A: 6	integer
product ID	ID of the product. For Adventurer: 0xE0070000, Runner 3 Music 0xD1070000	2	integer
Start time	Start time of the activity, as epoch seconds, e.g. 0x5A0EF328 which corresponds to GMT: Friday, November 17, 2017 14:33:12	4	integer
Software version	On the Adventurer: all 0x00	16	byte array
GPS firmware version	On the Adventurer: all 0x00	80	byte array
Watch time	Watch time as epoch seconds. On the adventurer the same as “Start time”	4	integer
Local time offset	Time offset between local time and GMT. For Amsterdam this is 3600 seconds in winter, 7200 seconds in summer.	4	integer
Reserved		1	
Length records	The next section of the header defines the record tags that appear in the file, with the corresponding record length. This field defines the number of tag-length pairs in the array.	1	integer
Array:			
Tag	Record tag, defining the type of record	1	integer
Length	Length of the record in bytes. For a number of records (e.g. 0x4B) a length of 0xFFFF is defined, meaning variable length. In that case the length is defined in the record itself.	2	integer

STATUS RECORD

The record indicates status changes. READY -> ACTIVE <-> PAUSED -> STOPPED. READY is the state when the activity is chosen. ACTIVE is when 'get going' is selected. PAUSED when the activity is PAUSED by pressing the left button on the watch. STOPPED is when the left button is pressed another time and the activity is finished.

Length: 7 bytes

Field	Description	Bytes	Format
Tag	0x21	1	Integer
Status	New status: READY – 0, ACTIVE – 1, PAUSED – 2, STOPPED – 3	1	Integer
Activity	Activity code	1	Integer
Timestamp	GMT Timestamp in epoch seconds	4	Integer

GPS RECORD

This record is added each second when the watch is in the ACTIVE state.

Length: 28 bytes

Field	Description	Bytes	Format
Tag	0x22	1	Integer
Latitude	Latitude * 1E7 degrees, -180E7 – 180E7 degrees	4	Integer
Longitude	Longitude * 1E7 degrees, -180E7 – 180E7	4	Integer
Heading	Heading * 1E2 degrees, 0-360E2	2	Integer
Speed	Speed in 1E2 m/s	2	Integer
Timestamp	GMT Timestamp in epoch seconds	4	Integer
Calories	Cumulative calories burned (cal)	2	Integer
Filtered speed	Some filtered speed value in m/s	4	Float
Distance	Cumulative distance in m	4	Float
Cycles	The cycles per second. For running 2-3.	1	Integer

EXTENDED GPS RECORD

Additional information regarding the GPS tracking.

Length: 20 (version<=0x09) or 24 bytes (version>=0x0a).

Field	Description	Bytes	Format
Tag	0x23	1	Integer
EVPE	Estimated Vertical Precision Error in cm	2	Integer
EHPE	Estimated Horizonal Precision Error in cm	2	Integer
HDOP	Horizontal Dilution of Precision	1	Integer
Unknown		4	Int Array
Unknown		4	Int Array
Unknown		4	Int Array
Unknown		1	Integer
Unknown		1	Integer
TBD		4	

HEART RATE RECORD

This record is added each second when the watch is in the ACTIVE state and the HR sensor is active or an external HR sensor is connected.

Length: 6 bytes

Field	Description	Bytes	Format
Tag	0x25	1	Integer
Unknown	0xFF for external HR sensor, other value for internal sensor	1	Integer
Timestamp	Timestamp in epoch seconds. Oddly enough, this is local time, whereas the rest of the timestamps in is GMT.	4	Integer

SUMMARY RECORD

Summary of the activity. Logged when the activity is STOPPED.

Length: 14 (version<=0x09) or 18 bytes (version>=0x0a).

Field	Description	Bytes	Format
Tag	0x27	1	Integer
Activity	Activity code	1	Integer
Distance	Distance	4	Integer
Duration	Duration of the activity in seconds. Excluding pause.	4	Integer
Calories	Calories burned during the activity	2	Integer
Unknown	? 0x004A=74 – Starting heartrate??	2	Integer
Duration2	Seems to be the duration. When paused slightly longer (2-3 sec) than Duration…	4	Integer

POOL SIZE RECORD

Pool size. Used in Swimming activity.

Length: 5 bytes

Field	Description	Bytes	Format
Tag	0x2A	1	Integer
Pool size	Pool size in cm	4	Integer

WHEEL SIZE RECORD

Wheel size. Used in cycling.

Length: 5 bytes

Field	Description	Bytes	Format
Tag	0x2B	1	Integer
Wheel size	Wheel circumference in mm, as defined under Cycling.	4	Integer

TRAINING SETUP RECORD

Defines the training. Not used if no training set.

Length: 10 bytes

Field	Description	Bytes	Format
Tag	0x2D	1	Integer
Goal	Training goal 0 = goal distance, 1 = goal time, 2 = goal calories, 3 = zones pace, 4 = zones heart, 5 = zones cadence, 6 = race, 7 = laps time, 8 = laps distance, 9 = laps	1	Integer
Minimum	Minimum value: metres, seconds, calories, sec/km, km/h, bpm sec/km, km/h, bpm (only used for zones).	4	Float
Maximum	Maximum value. Used in combination with the miminum, e.g. to indicate a heartrate zone min and max value. If only one limit is needed, only minimum is used and Maximum is set to 0x00000000.	4	Float

LAP RECORD

Lap not checked

Length 11 bytes

Field	Description	Bytes	Format
Tag	0x2F	1	Integer
Time	Total time in seconds	4	Integer
Distance	Total distance in meters	4	Float
Calories	Total calories (cal)	2	Integer

0X30 RECORD

Occurs after the activity is selected.

Length 3 bytes

Field	Description	Bytes	Format
Tag	0x30	1	Integer
?	Values: 0x01-0x0f?		
?	Value: 0x00		

CYCLING CADENCE RECORD

Revolutions and time counters. Can be used to calculate the cadence. not checked

Length 11 bytes

Field	Description	Bytes	Format
Tag	0x31	1	Integer
Wheel revolutions	Counts the wheel revolutions	4	Integer
Wheel revolutions time	Counts the time in ms	2	Integer
Crank revolutions	Counts the crank revolutions	2	Integer
Crank revolutions time	Counts the time in ms	2	Integer

TREADMILL RECORD

Treadmill. not checked

Length: 17 bytes

Field	Description	Bytes	Format
Tag	0x32	1	Integer
Timestamp	Timestamp in epoch seconds, UTC	4	Integer
Distance	Total distance in m	4	Float
Calories	Calories burned	2	Integer
Steps	Number of steps since ??	4	Integer
Step length	Step length in cm	2	Integer

SWIM RECORD

Treadmill. not checked

Length: 21 bytes

Field	Description	Bytes	Format
Tag	0x34	1	Integer
Timestamp	Timestamp in epoch seconds, UTC	4	Integer
Distance	Total distance in m	4	Float
Frequency		1	Integer
Stroke type		1	Integer
Strokes	Strokes since the last record	4	Integer
Completed laps		4	Integer
Calories		2	Integer

0X37 RECORD

Occurs a few seconds after the watch is set to active (activity is started). not checked

Length: 2 bytes

Field	Description	Bytes	Format
Tag	0x37	1	Integer
?	Counter or status value?? Value: 1. No lap counter, no intervals counter.	1	

INTERVAL SETUP RECORD

Interval training setup as defined on the watch. not checked

Length: 22 bytes

Field	Description	Bytes	Format
Tag	0x39	1	Integer
Warm type	0 – Distance, 1 - Time	1	Integer
Warm	Warm up in meters or seconds	4	Integer
Work type	0 – Distance, 1 – Time	1	Integer
Work	Work in meters or seconds	4	Integer
Rest type	0 – Distance, 1 – Time	1	Integer
Rest	Rest in meters or seconds	4	Integer
Cool type	0 – Distance, 1 – Time	1	Integer
Cool	Cool down in meters or seconds	4	Integer
Sets	Number of sets	1	Integer

INTERVAL START RECORD

Start of the interval. not checked

Length: 2 bytes (version ≤ 0x09), 3 bytes (version ≥ 0x0a)

Field	Description	Bytes	Format
Tag	0x3A	1	Integer
Type	1 - warm up, 2 - work, 3 - rest, 4 - cool down, 5 - finished	1	Integer

INTERVAL FINISH RECORD

Interval finish report. not checked

Length: 12 bytes (version ≤ 0x09), 14 bytes (version ≥ 0x0A)

Field	Description	Bytes	Format
Tag	0x3B	1	Integer
Type	1 - warm up, 2 - work, 3 - rest, 4 - cool down, 5 - finished	1	Integer
Time	Duration of the interval in seconds	4	Float
Distance	Distance covered during the interval in m	4	Integer
Calories	Calories burned	2	Integer
?			

RACE SETUP RECORD

Race definition file. not checked

Length: 41 bytes

Field	Description	Bytes	Format
Tag	0x3C	1	Integer
Race ID	Only used for web services race, otherwise 0	16	Byte array
Distance	Distance in m	4	Float
Duration	Duration in seconds	4	Integer
Name	Null terminated character string	16	Char array

RACE RESULT RECORD

Race results. not checked

Length: 11 bytes

Field	Description	Bytes	Format
Tag	0x3D	1	Integer
Distance	Distance in m	4	Float
Duration	Duration in seconds	4	Integer
Calories	Calories burned	2	Integer

ALTITUDE UPDATE RECORD

Altitude record. Since version ≥ 0x0A no longer present on the Adventurer. It has been replaced by the elevation record (tag=0x47) not checked

Length: 8 bytes

Field	Description	Bytes	Format
Tag	0x3E	1	Integer
Rel. Altitude	Relative altitude since start of the workout	2	Integer
Climb	Total climb	4	Float
Qualifier	Not defined yet	1	

HEART RATE RECOVERY RECORD

This record presents the heart rate recovery. The recovery is measured when the watch is set to pause. During 1 minute the decrease in heart rate is recorded.

Length: 9 bytes

Field	Description	Bytes	Format
Tag	0x3F	1	Integer
Score	Score: 0 – no recovery, 1 – poor recovery, 2 – fair recovery, 3 – good recovery, 4 – excellent recovery (≥40 bpm)	4	Integer
Recovery	Heart rate recovery in BPM per minute. A positive value means decrease, a negative value means an increase after the minute (no recovery).	4	Integer

INDOOR CYCLING RECORD

Indoor cycling not checked

Length: 12 bytes

Field	Description	Bytes	Format
Tag	0x40	1	Integer
Timestamp	Timestamp in epoch seconds, UTC	4	Integer
Distance	Distance in m	4	Integer
Calories	Calories burned	2	Integer
Cadence	Cadence	1?	Integer

GYM RECORD

Gym record not checked

Length: 11 bytes

Field	Description	Bytes	Format
Tag	0x41	1	Integer
Timestamp	Timestamp in epoch seconds, UTC	4	Integer
Calories	Calories burned	2	Integer
Cycles	Total number of cycles	4	Integer

MOVEMENT RECORD

Some status regarding movement TBD

Length: 2 bytes

Field	Description	Bytes	Format
Tag	0x42	1	Integer
Movement status	Not clear: 0 - standing still, 1 – moving slower, 2 – moving, 3 – moving??		

ROUTE DESCRIPTION RECORD

Description of the planned route.

Length: 101 bytes

Field	Description	Bytes	Format
Tag	0x43	1	Integer
??	0x00?	16	
??		4	
Route name	Null terminated string	80	Char array

ELEVATION RECORD (ADVENTURER)

Contains barometric elevation information

Length: 12 bytes

Field	Description	Bytes	Format
Tag	0x47	1	Integer
Unknown	Some status. Bit values.	1	Integer
Elevation 1	Absolute altitude, probably GPS altitude, in m	2	Integer
Elevation 2	Relative altitude, starting at 0, in m	2	Integer
Ascend	Total cumulative ascend in m	2	Integer
Descend	Total cumulative descend in m	2	Integer
Unknown	Seems to be a measure for the height increase (dz/dt)	2	Integer

BATTERY RECORD

Battery record. Occurs at 30 s (occasionally at 60 s) intervals.

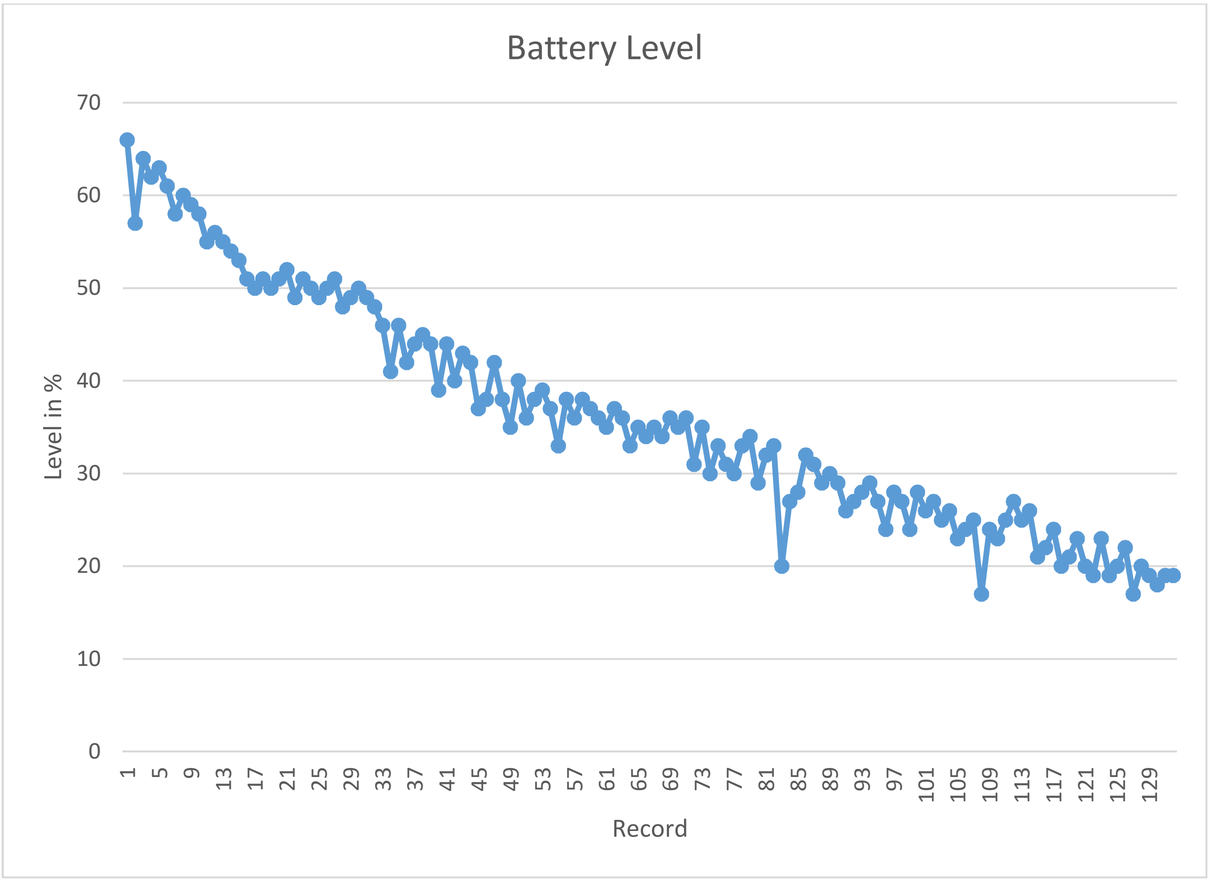


Figure 1 Typical battery level values (recorded during 1h20' run on Adventurer)

Length 5 bytes

Field	Description	Bytes	Format
Tag	0x49	1	Integer
Level	Battery level in % (0-100). Tends to decreasing during the exercise as one might expect, though subsequent values might fluctuate (down and up).	1	Integer
Unknown1	Typical value 127	1	Integer
Unknown2	Typical values 4, 5, 6	1	Integer
Unknown3	Typical value 0	1	Integer

FITNESSPOINTS RECORD

Contains the TomTom fitness points. Fitness points are an evaluation of the workout that depends on the heart rate.

Length 9 bytes

Field	Description	Bytes	Format
Tag	0x4A	1	Integer
Timestamp	Timestamp in epoch seconds	4	Integer
Fitnesspoints 1	Cumulative fitness points	2	Integer
Fitnesspoints 2	Appears to be same value as Fitnesspoints 1	2	Integer

WORKOUT RECORD 1

Has something to do with the workout.

Length: variable (0xFF)

Field	Description	Bytes	Format
Tag	0x4B	1	Integer
Length	Length of the remainder of the record in bytes	2	Integer
??			
??			

WORKOUT RECORD 2

Has something to do with the workout. Contains the messages. The record does not seem to be mentioned in the header (tag=0x20)

Length: variable (0xFF)

Field	Description	Bytes	Format
Tag	0x4C	1	Integer
Length	Length of the remainder of the record in bytes	2	Integer
??			
??			

ACTIVITY TRACKING FILES (0X00B1NNNN, 0X00B2NNNN, 0X00B3NNNN)

Daily activity is tracked in the activity tracking files. The format is protobuf format. Next protobuf definition shows the content of the files.

```
syntax = "proto2";

package tutorial;
//##################################################################################################
//
// Definition file for the TomTom activity tracking files (files with ID 0x00b1nnnn).
//
//##################################################################################################

<Language specific statements>

//##################################################################################################
// 1st level: root container
//##################################################################################################

message RootContainer
{
  optional Metadata             metadata          =7;
  optional DataContainer        dataContainer     =8;
}

//##################################################################################################
// 2nd level: metadata. Containing just two ints having the same value. Occurs often in one file
//##################################################################################################

message Metadata
{
  required fixed32              unknown1=1;  // always 0x1234DAEB
  required fixed32              unknown2=2;  // always 0x00010100
}

//##################################################################################################
// 2nd level: data container. Contains various types of data
//##################################################################################################

message DataContainer
{
  required SubDataContainer     subDataContainer  =1;
}

//##################################################################################################
// 3rd level: sub data container
//##################################################################################################


message SubDataContainer
{
  optional DeviceInfo           deviceInfo        =1;   // Device info
  optional TrackRecord          trackRecord       =2;   // activity tracking (calories, steps, hr,
sleep, etc)
  optional Record2              record2           =3;   // ?
  optional Record6              record6           =5;   // ?
  optional HeartRecord          heartRecord       =6;   // Heart rate
  optional Record4              record4           =7;   // ?
  optional FitnessRecord        fitnessRecord     =9;   // TomTom FitnessPoints counters
}

//##################################################################################################
// 4th level: Device info
//##################################################################################################

message DeviceInfo
{
  required string               deviceName        =1;
  required fixed32              year              =2;
  required SoftwareVersion      softwareVersion   =3;
}

message SoftwareVersion
{
  required int32                majorVersion      =1;
  required int32                mediumVersion     =2;
  required int32                minorVersion      =3;
  required Unknown01            unknown01         =4;
  optional Unknown02            unknown02         =5; // 1.6.26
}


message Unknown01
```

```
{
  optional int32                dummy            =1;
}


message Unknown02
{
  optional int32                dummy            =1;
}

//###################################################################################################
// 4th level: Tracker Record
//###################################################################################################
message TrackRecord
{
  required int32                recordId         =1;
  required int32                time             =2;    // Epoch time in seconds, 900 s interval = 15
min
  required int32                timeZone         =3;    // Difference between time and UTC in seconds
(?)
  required int32                interval         =4;    // Interval of this measurement = 900s often
  required int32                steps            =5;    // steps
  required int32                active           =6;    // active time in seconds
  required int32                distance         =7;    // distance in m
  required int32                kcal             =8;    // kcal
  required int32                kcalRest         =9;    // kcal in rest? (~12 kcal/10 min)
  required int32                unknown1         =11;   // 0
  optional int32                sleepTime        =12;   // sleep time in seconds.
  optional int32                sleepMode        =13;   // 1: active, 2: charging? 3: 1st hour sleep 4:
sleep
                                                        // Total sleep time: sum sleep time if
sleepMode=3 or 4
                                                        // from 12:00:00 till 12:00:00
}

//###################################################################################################
// 4th level: Data Record 2
//###################################################################################################

message Record2
{
  required fixed32              time             =1;    // Epoch time in seconds, 900 s interval = 15
min
  required int32                interval         =2;    // Interval in seconds? 14400=4h
  repeated TagValueContainer    tagValue         =3;
  required int32                unknown1         =4;
  required int32                unknown2         =5;
  optional int32                unknown3         =6;
  optional fixed32              unknown4         =104;  // 1.6.26

}

message TagValueContainer
{
  required int32                tag              =1;
  required ValueContainer       valueContainer   =2;
}

message ValueContainer
{
  required int32                value            =3;
}

//###################################################################################################
// 4th level: Data Record 3 - heart rate
//###################################################################################################

message HeartRecord
{
  required fixed32              time             =1;    // Epoch time in seconds, 900 s interval = 15
min
  required int32                interval         =2;    // Interval in seconds?? 14400=4h
  required int32                heartRate        =3;    // Heartrate in bpm. This value is displayed by
TomTom MySports
  required int32                value01          =4;    // Some other heartrate value
  required int32                value02          =5;    // ?
}


//###################################################################################################
// 4th level: Data Record 4
//###################################################################################################

message Record4
{
  required fixed32              time             =1;    // Epoch time in seconds, 900 s interval = 15
min
```

```
  required int32                interval          =2;    // Interval in seconds?? 14400=4h or 
14280=3h58'???
  repeated int32                value             =3;    // 6 values in the message
}

//###############################################################################################
// 4th level: Data Record 5
//###############################################################################################

// Occurs every 120 seconds when logging activity

message UserData
{
  required string               version           =1;
  required int32                unknown1          =2;
  required int32                unknown2          =3;
  required int32                unknown3          =4;
  required int32                length            =5;  // Users heigth in cm
  required int32                weight            =6;  // User weight
  required int32                unknown4          =7;
  required int32                unknown5          =8;
  required int32                unknown6          =9;
  required int32                unknown7          =10;
  required int32                unknown8          =11;
  required int32                unknown9          =12;
  required int32                unknown10         =13;
  required int32                unknown13         =16;
  required int32                unknown14         =17;
  required int32                unknown15         =18;
  required int32                unknown16         =19;
}

message FitnessRecord
{
  required int32                time              =2;    // Epoch time in seconds, 120 s interval = 2 
min during activity
  required int32                interval          =3;    // Interval in seconds?? 14280=238 min = 
3h58'??
  optional UserData             userData          =5;    // User data
  optional int32                unknown1          =7;
  optional int32                unknown2          =8;
  optional int32                unknown3          =10;
  optional int32                unknown4          =11;
  optional int32                unknown5          =12;
  required int32                fitnessPoints1    =15;  // Cummulative TomTom activity points counter
  required int32                fitnessPoints2    =16;  // Cummulative TomTom activity points counter

}

//###############################################################################################
// 4th level: Data Record 6
//###############################################################################################

message Record6Sub
{
  required int32                unknown1          =1;
  required int32                unknown2          =2;
  required int32                unknown3          =3;
}

message Record6
{
  required int32                time              =1;    // Epoch time in seconds, 900 s interval = 15 
min
  required int32                interval          =2;    // Interval in seconds?? 14280=238 min = 
3h58'??
  required int32                unknown1          =3;
  repeated Record6Sub           sub               =4;    // Some data
}

//###############################################################################################
// Root message
//###############################################################################################


message Root
{
  repeated RootContainer        rootContainer =1;
}
```

ROUTEFILES (0X00B8NNNN)

Route files were added on the TomTom Adventurer for ahead track planning. The files are Google 'Protocol Buffers' encoded. Protocol buffers (https://developers.google.com/protocol-buffers/: *Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler*). The file format is defined in a .proto file that can be compiled into encoding/decoding code for various programming languages.

Below you find the .proto definition file for the route files.

```
//##################################################################################################
//
// Definition file for the TomTom routes (files with ID 0x00b8nnnn). A route is the result of
// trackplanning by converting and uploading GPX track files.
//
//##################################################################################################
syntax = "proto2";

<Language specific statements>

message MetaData
{
  required fixed32              unknown1            =1;  // always 0x1234DAEB?
  required fixed32              unknown2            =2;  // always 0x00010100?
}

// 1st level
message RootContainer
{
  optional MetaData             metaData            =7;
  optional TrackLevel1          level1              =8;
}

message LatLon
{
  required fixed32              value               =1;
}

// Container
message Coordinate
{
  required LatLon               lat                 =1;
  required LatLon               lon                 =2;
}

// Coordinate container
message CoordinateData
{
  required Coordinate           coordinate          =1;
}

// Start coordinate
message StartCoordinate
{
  required Coordinate           coordinate          =1;
  required int32                index               =2;
}
```

```
// Route segment
message Segment
{
  required int32                numberOfCoordinates =1;
  repeated CoordinateData       data                =2;
}

// Segment section
message SegmentData
{
  required int32                numberOfSegments    =1;
  repeated Segment              data                =2;
}

// Bounding box enclosing the route
message BoundingBox
{
  required LatLon               latDown             =1;
  required LatLon               lonLeft             =2;
  required LatLon               latUp               =3;
  required LatLon               lonRight            =4;
}

// Some information on the route
message TrackMetaData
{
  required string               name                =1;
  required BoundingBox          box                 =2;
  required bytes                time                =3;
}

// 3rd level
message TrackLevel2
{
  required TrackMetaData        metadata            =1;
  repeated StartCoordinate      coordinate          =2;
  required SegmentData          data                =3;
}

// 2nd level
message TrackLevel1
{
  required TrackLevel2          level2              =1;
}


// The Root
message Root
{
  repeated RootContainer        container           =1;
}
```

The highest level is a .proto file is `Root`. It contains three levels of containers: `RootContainer`, `TrackLevel1` and `TrackLevel2`. At `TrackLevel2` we find `TrackMetadata` (name, bounding box, time), the `StartCoordinate` and the `SegmentData`. `SegmentData` contains one or more `Segment`s, each segment contains the `Coordinate`s.

The total number of `Coordinate`s in all `Segment`s should not exceed 500. If a route to encode contains more than 500 points, skip some intermediate coordinates.

ACTIVITY TYPES

In several places a code is used to identify the activity. Next table shows the meaning:

Code	Sport
0x00	Running
0x01	Cycling
0x02	Swimming
0x07	Treadmill
0x08	Freestyle
0x09	Gym
0x0a	Hiking
0x0b	Indoor cycling
0x0e	Trail running
0x0f	Skiing
0x10	Snowboarding

USB INTERFACE

GENERIC MECHANISM

The communication to the watch at low level takes place by writing request packets to the write endpoint of the USB device and reading response packets from the read end point.

Request packet (TX):

1	1	1	1	N, 0≤N≤252
0x09	N+2	Counter	tx msg type	Payload

Response packet (RX):

1	1	1	1	M, 0≤M≤252
0x01	M+2	Counter	rx msg type	Payload

Header (green)

1. Start of message
   0x09 for TX, 0x01 for RX
2. Length of remaining part of the message
   Including the remaining header bytes
3. Counter
   Should be increased on each TX. RX reflects the value sent in the corresponding TX. Can be used to check if the response belongs to the request.
4. Message type
   Usually the msg type in the response is equal to the msg type in the request. The exception is the read data request (MSG_READ_FILE_DATA_REQUEST)

Version 0.2

Device	Vendor ID	Product ID	Read endpoint	Write endpoint	RX/TX Packet size
Multisports	0x1390	0x7474	0x84	0x05	64
Spark Music	0x1390	0x7475	0x81	0x02	256
Runner Music	0x1390	0x7475	0x81	0x02	256
Spark Cardio	0x1390	0x7475	0x81	0x02	256
Runner Cardio	0x1390	0x7477	0x81	0x02	256
Adventurer	0x1390	0x7477	0x81	0x02	256

In case the packet size is 256 and N<252, the remainder of the packet can be set to 0.

Identified message types:

Message type (cmd)	Description
0x02	MSG_OPEN_FILE_WRITE
0x03	MSG_DELETE_FILE
0x04	MSG_WRITE_FILE_DATA
0x05	MSG_GET_FILE_SIZE
0x06	MSG_OPEN_FILE_READ
0x07	MSG_READ_FILE_DATA_REQUEST
0x09	MSG_READ_FILE_DATA_RESPONSE
0x0A	~~MSG_FIND_CLOSE~~
0x0C	MSG_CLOSE_FILE
0x0D	MSG_UNKNOWN_0D
0x0E	MSG_FORMAT_WATCH
0x10	MSG_RESET_DEVICE
0x11	MSG_FIND_FIRST_FILE
0x12	MSG_FIND_NEXT_FILE
0x14	MSG_GET_CURRENT_TIME
0x1A	MSG_UNKNOWN_1A
0x1D	MSG_RESET_GPS_PROCESSOR
0x1F	MSG_UNKNOWN_1F
0x20	MSG_GET_PRODUCT_ID
0x21	MSG_GET_FIRMWARE_VERSION
0x22	MSG_UNKNOWN_22
0x23	MSG_UNKNOWN_23
0x28	MSG_GET_BLE_VERSION

HIGHER LEVEL FUNCTIONS

OPEN FILE/CLOSE FILE/DELETE FILE

TX:

1	1	1	1	4
0x01	6	cnt	**CMD**	File ID

RX:

1	1	1	1	4	4	4	4	4
0x01	22	cnt	**CMD**		File ID			Error

Variable	Description
CMD	Command byte (message type)
File ID	ID of the file. Appears to be 0x00000000 on the adventurer
Error	Error indication. 0 – no error, other value - error

Command (cmd)	Description
0x02	MSG_OPEN_FILE_WRITE
0x03	MSG_DELETE_FILE
0x06	MSG_OPEN_FILE_READ
0x0C	MSG_CLOSE_FILE

READING FILES

1. Open file for reading (MSG_OPEN_FILE_READ)
2. Request file size (=bytes to be read; MSG_GET_FILE_SIZE)
3. Repeat: read file data chunk (MSG_READ_FILE_DATA_REQUEST)
4. Close file (MSG_CLOSE_FILE)

Reading file data: File is read in chunks. The amount of bytes to read is defined in the TX message

TX:

1	1	1	1	4	4
0x01	10	cnt	**0x07**	File ID	Length

RX:

1	1	1	1	4	4	Read
0x01	Read+10	cnt	**0x09**	File ID	Read	File data

Variable	Description
File ID	File to read
Length	Bytes to read from the opened file. For the Multisports it is max. 50, for Spark, Runner, Adventurer it is max. 242 bytes.
Read	Bytes read. Should be equal to Length
File Data	Chuck of file data read.

WRITING FILES

1. Open file for writing (MSG_OPEN_FILE_WRITE)
2. Repeat: Write file data chunk (MSG_WRITE_FILE_DATA)
3. Close file (MSG_CLOSE_FILE)

Writing file data in chunks:
File is written in chunks. Therefore the file data has to be split up in chunks. The number of chunks is roundup(fileSize/maxChunkSize). The maxChunkSize depends on the watch type (see below).

TX:

1	1	1	1	4	Length
0x01	Length+6	cnt	**0x04**	File ID	Chunk data

RX:

1	1	1	1	4	4	4	4	4
0x01	22	cnt	**0x04**		File ID			

Variable	Description
File ID	File to write
Length	Bytes to write from the opened file. For the Multisports it is max. 54, for other Spark, Runner, Adventurer it is max. 246 bytes.
Read	Bytes read. Should be equal to Length
File Data	Chuck of file data read.

REQUEST FILE SIZE

This call returns the file size. On the Multisports model the file must be opened for reading first (MSG_OPEN_FILE_READ).

TX:

1	1	1	1	4
0x01	6	cnt	**0x05**	File ID

RX:

1	1	1	1	4	4	4	4	4
0x01	22	cnt	**0x05**		File ID		Size	Error

Variable	Description
File ID	ID of the file. Appears to be 0x00000000 on the Adventurer
Size	File size in bytes
Error	>0 when an error occurs: if the file does not exist or the file has not been opened for reading (Multisport model only). 0 If no error occurred.

LISTING/ENUMERATING FILES

This method can be used to enumerate files IDs and corresponding lengths.

1. List first file, resets the enumeration (MSG_FIND_FIRST_FILE)
2. Repeat: list subsequent files (MSG_FIND_NEXT_FILE)
3. ~~Close find (MSG_FIND_CLOSE)~~

List first file:

TX:

1	1	1	1	4	4
0x01	10	cnt	**0x11**	0	0

RX:

1	1	1	1	4	4	4	4	4
0x01	22	cnt	**0x11**		File ID		Size	End of list

List next file:

TX:

1	1	1	1
0x01	2	cnt	**0x12**

RX:

1	1	1	1	4	4	4	4	4
0x01	22	cnt	**0x12**		File ID		Size	End of list

Variable	Description
File ID	ID of the file.
Size	File size in bytes
End of list	0 if more files available, otherwise if not. If this value is unequal to 0, File ID and size have no meaning.

GET WATCH TIME

TX:

1	1	1	1
0x01	2	cnt	**0x14**

RX:

1	1	1	1	4	4	4	4	4
0x01	22	cnt	**0x14**	Time				

Variable	Description
Time	The current time in epoch seconds (UTC)

FIRMWARE VERSION (MSG_GET_FIRMWARE_VERSION)

TX:

1	1	1	1
0x01	2	cnt	**0x21**

RX:

1	1	1	1	length
0x01	2+length	cnt	**0x21**	Version

Variable	Description
Version	The version as string of *length* bytes, like '1.7.53'

BLUETOOTH LOW ENERGY VERSION (MSG_GET_BLE_VERSION)

TX:

1	1	1	1
0x01	2	cnt	**0x28**

RX:

1	1	1	1	4
0x01	6	cnt	**0x28**	Version

Variable	Description
Version	The version

PRODUCT ID (MSG_GET_PRODUCT_ID)

TX:

1	1	1	1
0x01	2	cnt	**0x20**

RX:

1	1	1	1	4
0x01	6	cnt	**0x20**	Product ID

Variable	Description
Product ID	Product ID, for Adventurer 0xe0070000.

RESET DEVICE (MSG_RESET_DEVICE)

TX:

1	1	1	1
0x01	2	cnt	**0x10**

The reset is used after uploading firmware files. The reboot installs the firmware. Since the watch resets, no response (RX) is sent.

RESET GPS PROCESSOR (MSG_RESET_GPS_PROCESSOR)

TX:

1	1	1	1
0x01	2	cnt	**0x1D**

RX:

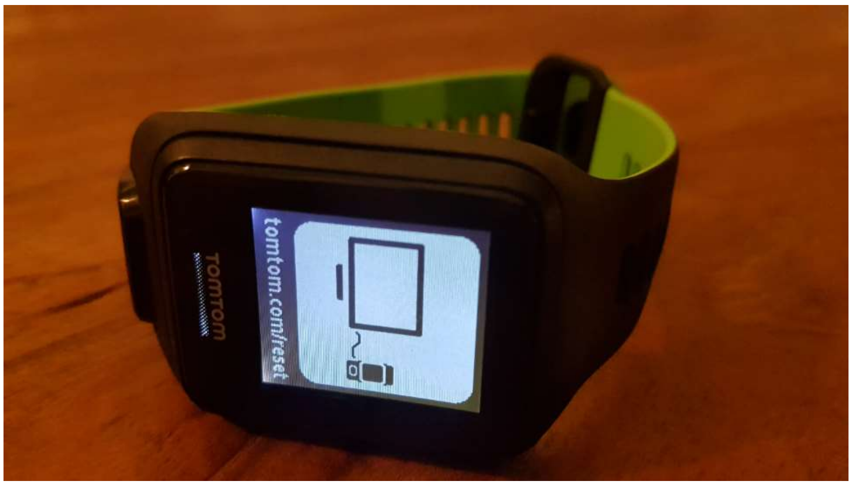
1	1	1	1	length
0x01	2+length	cnt	**0x1D**	Reboot message

Variable	Description
Reboot message	A message as string of *length* bytes, like ‘wait 1 minute before disconnecting USB’

FORMAT WATCH (MSG_FORMAT_WATCH)

This method formats the watch. After formatting, it is required to download and write the firmware as described in 'Firmware files (0x000000F0, 0x00000012, 0x00010200)'. Beware that the required preference file is also deleted during the format, so download this file first before formatting the watch.

Disconnecting the watch after format and before writing the firmware results in the watch asking to be reconnected again:



After the format and writing the firmware the watch can be registered again using TomTom Sports Connect.

TX:

1	1	1	1
0x01	2	cnt	**0x0E**

RX:

1	1	1	1	4	4	4	4	4
0x01	22	cnt	**0x0E**					Error

Variable	Description
Error	0 if no error, >0 if an error occurred