

## Interface data structures

### **GPS reception module**

This module has a input queue (GPSrecQ) and writes output data in all the queues that the “connection” commands receive indicated.

All structures are defined in recGPS.h

### **Received messages**

Only MSG\_CONTROL messages (value = 1) can be received in this queue, message structure is:

```
typedef struct RGPSctlMsg
{
    long mtype; /* mtype is 1 for control messages */
    int iRGPSid; /* values, GPSREC_CON, GPSREC_DISC, GPSREC_ACK, GPSREC_PLTFILE
*/
    union RGPSctl
    {
        RGPS_con stcRGPS_con;
        RGPS_disc stcRGPS_disc;
        RGPS_PLTfile stcRGPS_PLTfile;
    } RGPSctl;
} RGPS_ctlMsg;
```

Where:

- mtype:** *message type. This is a mandatory variable as first element in msgbuf structure, indicating message type. For control messages mtype = 1, for data messages mtype = 2. This module will always receive mtype = MSG\_CONTROL (1) messages.*
- iRGPSid:** *GPS Reception message ID. Indicates which control message it is. Value GPSREC\_CON (=1) indicates connection message, GPSREC\_DISC (=2) indicates disconnection, GPSREC\_PLTFILE (=500) indicates new track file for readOzitrack simulator.  
(readOzitrack is a program that simulates the recGPS module, but reading data from an Oziexplorer® track file)*
- RGPSctl:** *GPS reception message info union. Is a union of three possible sub-structures: stcRGPS\_con for connection messages, stcRGPS\_disc for disconnect messages and stcRGPS\_PLTfile for new file used in readOzitrack program.*

**StcRGPS\_con structure**

This is the structure containing all data needed for the connection message to GPS reception module.

```
// Conection struct
typedef struct RGPS_con
{
    int iDatum;
    unsigned char btUnits;
    unsigned char btPosFormat;
    key_t ktQueue;
    int iQid; /* used internally by GPS receive module */
} RGPS_con;
```

It contains the following fields:

- iDatum:** Datum in which GPS rec module must send the position data to calling module. Values are the same as defined in table datum\_structure defined in library Jeeps 1.1.4.
- DT\_WGS84 (118), indicates data should be received in WGS84 datum.
  - DT\_EUR50 (38), datum is European 1950
  - DT\_EUR79 (39), datum is European 1979
  - DT\_PNIEVES (87), datum is “Pico de las Nieves” (Canary Islands)
- btUnits:** Units in which distance and speed data should be received, values are:
- UD\_ESTATUATE (0), units are Estatuete (miles, mph)
  - UD\_METRIC (1), units are metric (meters, K/h)
  - UD\_NAUTIC (2), units are Nautic (nautic miles, knots)
- btPosFormat:** Position format. Indicates the format in which position should be received, values are:
- PF\_HD (0), hddd.dddd° format
  - PF\_HD\_M (1), hddd° mm.mmm' format
  - PF\_HD\_M\_S (2), hddd° mm' ss.s” format
  - PF\_UTM (3), UPS UTM format (zone, Northing, Easting)
- ktQueue** *Queue key where GPS reception module will send the position messages. If queue not exist, connection is closed.*
- iQid** *Used internally by GPS receive module to store the Qid returned by iOpenQmsg function.*

**StcRGPS\_disc structure**

This structure contains all data needed for the disconnect message to the GPS reception module.

```
// Disconnect struct
typedef struct RGPS_disc
{
    key_t ktQueue;
} RGPS_disc;
```

Where:

*ktQueue*      Queue key of queue corresponding to connection to closed. Queue is not destroyed.

**stcRGPS\_PLTfile structure**

This structure contains the OziExplorer track file name to be used by readOzitrack simulator. This message is sent by readOzitrack\_file program, and used by readOzitrack, recGPS module will discard this message.

```
//PLTfile message struct
typedef struct RGPS_PLTfile
{
    char stFileName[128];
} RGPS_PLTfile;
```

Where:

*stFileName*    File path/name of the .plt Oziexplorer® track file (2.1 format)

## Sent messages

Every time the GPS reception module have a new position data (usually as consequence of a NMEA sentence received from GPS), will sent a position message to all modules that have a session connected.

The message to each destination will have the same data, but formats, datums and units will correspond to the negotiated during connection.

Only MSG\_DATA messages are sent by this module. The message sent, will have this general structure.

```
typedef struct RGPS_dataMsg
{
    long mtype; /* mtype is 2 for data messages */
    int iRGPSid; /* value: always 1; position message */
    int iDatum;
    unsigned char btUnits;
    unsigned char btPosFormat;
    short shNumSat; /* num visible satellites */
    double dDist; /* distance in m/miles from previous point */
    float fSpeed; /* Speed */
    double dBearing; /* Bearing of movement */
    double dAltitude; /* Altitude above datum ellipsoid */
    time_t tTime; /* Time of adquisition */
    union RGPS_POS
    {
        RGPS_POS00 stcD;
        RGPS_POS01 stcDM;
        RGPS_POS02 stcDMS;
        RGPS_POS03 stcUTM;
    } RGPSpos;
} RGPS_dataMsg;
```

Where:

- mtype*: message type. This is a mandatory variable as first element in msgbuf structure, indicating message type. For control messages *mtype* = 1, for data messages *mtype* = 2. This module send always data message, *mtype* = MSG\_DATA (2)
- iRGPSid*: GPS Send message ID. Indicates which control message it is. Value is always GPSEND\_POS (=1) indicates position message.
- iDatum*: Datum of position data in message, must be the same as datum requested in connection message. Values are the same as defined in table datum\_structure defined in library Jeeps 1.1.4.
- btUnits*: Units in which distance and speed data are. Must be the same as *btUnits* variable in connection message.
- btPosFormat*: Position format. Indicates the format in which position data is, value must be the same as in connection message.
- RGPSctl*: GPS reception message info union. Is a union of two possible sub-structures: *stcRGPS\_con* for connection messages and *stcRGPS\_disc* for disconnect messages.
- shNumSat*: Number of satellites currently received by GPS.
- dDist*: Distance in selected units since previous received data.
- fSpeed*: Speed (reveived from GPS in NMEA sentence).
- dBearing*: Bearing of movement
- dAltitude*: Altitude of current position over sea level.
- Ttime*: Time of data adquisition in Unix format.
- RGPSpos*: Send message info union. Is a union of four structures, each contain the data of one of the four possible position formats.

**RGPS\_pos00 structure**

This is the structure containing the position data for position format = PF\_HD (hddd.dddd°).

```
// pos 00 format (hddd.ddd°)
typedef struct RGPS_POS00
{
    double dNorth;
    double dEast;
} RGPS_POS00;
```

Where:

*dNorth*: Latitude position in hexagesimal degrees.  
*dEast*: Longitude position in hexagesimal degrees

**RGPS\_pos01 structure**

Contains the position variables when position format is PF\_HD\_M (hddd° mm.mmm').

```
/pos 01 format (hddd° mm.mmm')
typedef struct RGPS_POS01
{
    int iDNorth;
    double dMNorth;
    int iDEast;
    double dMEast;
} RGPS_POS01;
```

Where:

*iNorth*: Latitude position integer degrees  
*dMNorth*: Latitude position decimal minutes  
*iEast*: Longitude position integer degrees  
*dMEast*: Longitude position decimal minutes

**RGPS\_pos02 structure**

Contains position when position format is PF\_HD\_M\_S (hddd° mm' ss.sss")

```
// pos 02 format (hddd° mm' ss.s")
typedef struct RGPS_POS02
{
    int iDNorth;
    int iMNorth;
    float fSNorth;
    int iDEast;
    int iMEast;
    float fSEast;
} RGPS_POS02;
```

Where:

*iNorth*: Latitude position integer degrees  
*iMNorth*: Latitude position integer minutes  
*fSNorth*: Latitude position portion in decimal seconds.  
*iEast*: Longitude position integer degrees  
*iMEast*: Longitude position integer minutes  
*fSEast*: Longitude position in decimal seconds

**RGPS\_pos03**

Contains position when position format is PF\_UTM (UPS UTM)

```
// pos 03 format (UPS UTM)
typedef struct RGPS_POS03
{
    long lZone;
    char cZC;
    double dNorth;
    double dEast;
} RGPS_POS03;
```

Where:

lZone	UPS UTM zone
cZC	Zone letter
dNorth	UPS UTM Northing
dEast	UPS UTM Easting