

HsGpsDll Library User Manual (v1.6)

Date: 06/04/2012

Table of Contents

1	Introduction.....	2
1.1	About HsGpsDll.....	2
1.1.1	Overview.....	2
1.1.2	What components are included in HsGpsDll.....	2
1.1.3	Limitations in Evaluation Version.....	2
1.1.4	HsGpsDll Internal Architecture.....	3
1.1.5	Purchasing Information.....	3
2	HsGpsDll API Specification.....	4
2.1	Function Reference.....	4
2.1.1	HsGPSDllOpen.....	4
2.1.2	HsGPSDllOpenCs.....	5
2.1.3	HsGPSDllOpenVirtual.....	6
2.1.4	HsGPSDLLBufferRx.....	6
2.1.5	HsGPSDllClose.....	7
2.1.6	HsGPSDllCloseCs.....	7
2.1.7	HsGPSDLLGetPosition.....	7
2.1.8	HsGPSDLLGetPositionCs.....	9
2.1.9	HsGPSDLLGetPositionDegrees.....	11
2.1.10	HsGPSDLLGetPositionDegreesCs.....	12
2.1.11	HsGPSDLLGetAltitude.....	13
2.1.12	HsGPSDLLGetAltitudeCs.....	14
2.1.13	HsGPSDLLGetSpeed.....	14
2.1.14	HsGPSDLLGetSpeedCs.....	15
2.2	Error Codes and Other Constants.....	15
2.2.1	HsGpsDll return codes.....	15
2.2.2	Other Constants.....	16

1 Introduction

1.1 About HsGpsDll

1.1.1 Overview

HsGpsDll is a Windows Dynamic Link Library which provides access to any NMEA-183 compliant GPS receiver via a serial communications port.

HsGpsDll is designed for use from Visual C, Visual C Sharp .NET, Visual Basic or other programming languages, capable of calling DLL functions.

HsGpsDll allows a user application to read from a connected GPS device:

- current GPS geographical position fix (latitude and longitude) in degrees, and decimal seconds (for example 53' 20.6107" N, 6' 23.4648" W)
- current GPS geographical position fix (latitude and longitude) in decimal degree format (for example 53.343513 N, 6.391080 W)
- current speed (velocity over ground)
- number of satellites in view
- fix quality (valid or not)
- current altitude (against mean sea level)
- UTC date and time

To obtain the above information, HsGpsDll decodes the following sentences of NMEA-183 (National Marine Electronics Association, Interface Standard 0183):

- \$GPGGA - geographical position fix data
- \$GPRMC – minimum recommended fix data
- \$GPVTG - velocity over ground

Speed in kilometers per hour is obtained either from GPVTG message (if it is available) or from GPRMC message (in this case it is converted by the HsGpsDll from knots to kilometers per hour)

1.1.2 What components are included in HsGpsDll

HsGpsDll DLL License (without source code):

- HsGpsDll.dll - Windows DLL (Dynamic Link Library) for use from programs in Visual C, Visual Basic or other languages
- HsGpsDllCs.dll – Windows DLL (Dynamic Link Library) wrapper to enable C Sharp .NET programs to use the library
- HsGpsDemoVC - Visual C test application with source code using HsGpsDll.dll
- HsGpsDemoVB6 - Visual Basic test application with source code using HsGpsDll.dll
- HsGpsDemoCS – C Sharp .NET Demo application with source code using HsGpsDllCs.dll and HsGpsDll.dll

HsGpsDll Company Source Code License and Single developer source code license:

Includes everything in Standard edition plus FULL SOURCE CODE in C (plain C style), including core GPS NMEA message processing engine. The project files are in Microsoft Visual C format.

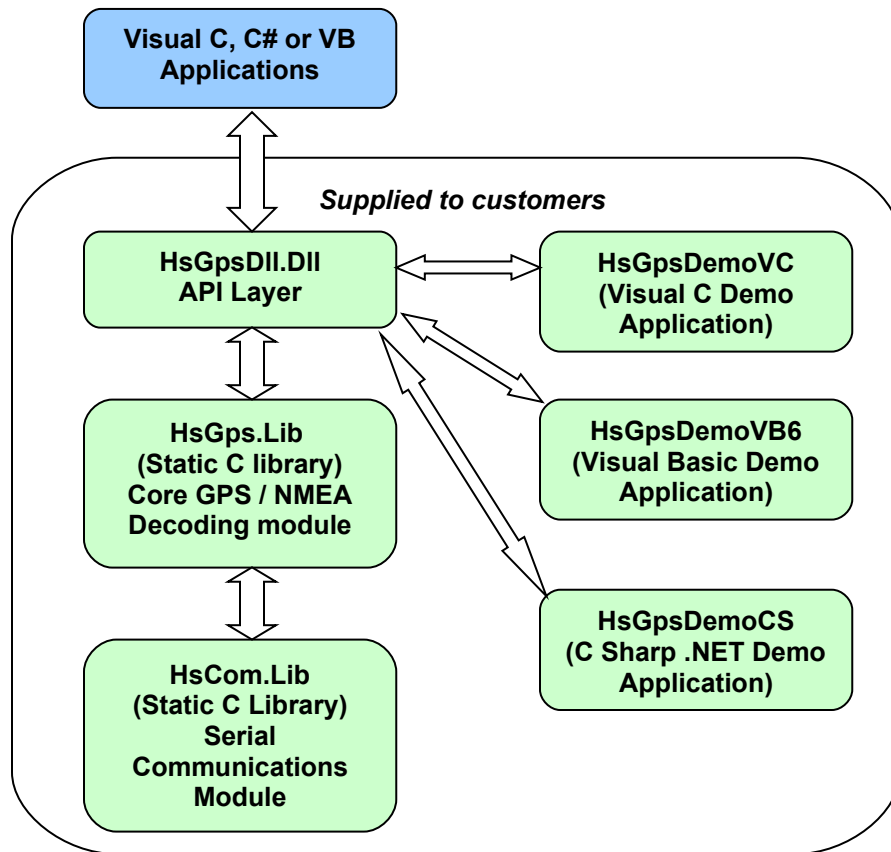
1.1.3 Limitations in Evaluation Version

There are several limitations in the evaluation version of HsGpsDll library, which are removed from the full version:

- Every time evaluation version of HsGpsDll library is loaded it displays a reminder message (nag screen), which requires a user to click Okay to continue.

- Evaluation version has a built in use count. After the use count is exceeded (the GPS position or speed retrieved several times), the library will fail every call and will return the corresponding error code (HS_GPSDLL_RC_EVAL_EXCEEDED= 0x06). The application will need to exit and restart to reset the use count.
- No source code is supplied with free evaluation version. The full source code is supplied with full version of HsGpsDll single developer and Company source code licenses.

1.1.4 HsGpsDll Internal Architecture



1.1.5 Purchasing Information

To purchase HsGpsDll library and for pricing information, please visit http://www.hillstone-software.com/hs_gpsdll_buy.htm

Alternatively contact us:

Hillstone Software,
<http://www.hillstone-software.com>
<mailto:info@hillstone-software.com>
 Tel. +353 1 626 8229
 M. +353 87 988 1568

2 HsGpsDII API Specification

2.1 Function Reference

2.1.1 HsGPSDIIOpen

Declaration:

```
extern long HS_GPSDLL_API HsGPSDIIOpen(  
    long portno, // baud rate 110 to 115200  
    long baudrate, // number of data bits  
    long databits, // parity  
    long parity, // parity  
    long stopbits, // number of stop bits  
    long options,  
    long *rc  
)
```

Summary:

Opens a GPS port and associated serial communications port and starts receiving NMEA data from a GPS receiver. Call this function before calling other functions to get current GPS information.

Parameters:

portno

Serial communications port number from 0 to 99 to which a GPS receiver is connected. 0=COM1, 1=COM2...99=COM100

baudrate

COM port speed, the following values are valid speeds: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000. Most GPS receivers operate at 4800 baud, however some receivers may operate at a different fixed or configurable rate.

databits

COM port data size in number of bits. Valid values are 5,6,7,8

parity

COM port parity code as follows: 2=EVEN, 3=MARK ,0=NONE, 1=ODD, 4=SPACE

stopbits

COM port stop bits code as follows: 0=0, 1=1.5, 2=2

options

Operating options: USE_GPGAA = position coordinates will be obtained by HsGpsDII from \$GPGGA NMEA sentence; USE_GPRMC = position coordinates will be obtained by HsGpsDII from \$GPRMC NMEA sentence

rc

Pointer to long variable to receive function return code (See "Return values" below)

Return values:

HS_GPSDLL_RC_ALREADY_INIT	GPS library is already initialised
HS_GPSDLL_RC_FAIL	Failed to open serial communications port
HS_GPSDLL_RC_OK	Success, GPS port is open

2.1.2 HsGPSDIIOpenCs

Declaration:

```
extern int HS_GPSDLL_API HsGPSDIIOpenCs(
    int portno, // COM port number (zero based)
    int baudrate, // baud rate 110 to 115200
    int databits, // number of data bits
    int parity, // parity
    int stopbits, // number of stop bits
    int options // options
)
```

Summary:

This function is for use from C Sharp.NET applications.

Opens a GPS port and associated serial communications port and starts receiving NMEA data from a GPS receiver. Call this function before calling other functions to get current GPS information.

Parameters:

portno

Serial communications port number from 0 to 99 to which a GPS receiver is connected. 0=COM1, 1=COM2...99=COM100

baudrate

COM port speed, the following values are valid speeds: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, 256000. Most GPS receivers operate at 4800 baud, however some receivers may operate at a different fixed or configurable rate.

databits

COM port data size in number of bits. Valid values are 5,6,7,8

parity

COM port parity code as follows: 2=EVEN, 3=MARK ,0=NONE, 1=ODD, 4=SPACE

stopbits

COM port stop bits code as follows: 0=0, 1=1.5, 2=2

options

Operating options: USE_GPGAA = position coordinates will be obtained by HsGpsDII from \$GPGGA NMEA sentence; USE_GPRMC = position coordinates will be obtained by HsGpsDII from \$GPRMC NMEA sentence

Return values:

HS_GPSDLL_RC_ALREADY_INIT	GPS library is already initialised
HS_GPSDLL_RC_FAIL	Failed to open serial communications port
HS_GPSDLL_RC_OK	Success, GPS port is open

2.1.3 HsGPSDIIOpenVirtual

Declaration:

```
extern long HS_GPSDLL_API HsGPSDIIOpenVirtual(  
    long *rc // pointer to variable to receive return code  
)
```

Summary:

This function is for use from C/C++ applications.

Opens a GPS port without opening the underlying serial communications port. If the user calls this function instead of HsGPSDIIOpen, it is expected that GPS NMEA data is passed to the HsGpsDII by calling function HsGPSDLLBufferRx(). This function is useful for sending GPS NMEA data to the HsGpsDII from a text file instead of real GPS device attached to the serial port.

Parameters:

rc
Pointer to long variable to receive function return code (See "Return values" below)

Return values:

HS_GPSDLL_RC_ALREADY_INIT	GPS library is already initialised
HS_GPSDLL_RC_OK	Success, GPS port is open

2.1.4 HsGPSDLLBufferRx

Declaration:

```
extern long HS_GPSDLL_API HsGPSDLLBufferRx(char *buffer, int buflen, long *rc);
```

Summary:

This function is for use from C/C++ applications.

Send raw NMEA 183 GPS sentence data to HsGpsDII for decoding. The data buffer can be any length, may contain multiple NMEA sentences. The buffer is not required finish on the sentence border.

The decoded GPS data can be obtained as usual by calling one of HsGPSDLLGetXXX functions described in this manual. This function should be used if GPS interface has been open by calling HsGPSDIIOpenVirtual(). Do not use this function if NMEA data is received from directly attached GPS device via the COM port.

Parameters:

buffer
pointer to buffer containing one or more NMEA 183 sentences. The buffer can contain any number of sentences which may be full or partial.

buflen
number of bytes in the buffer

rc
Pointer to long variable to receive function return code (See "Return values" below)

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised (HsGPSDllOpenVirtual was not called)
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_OK	Success, GPS port is open

2.1.5 HsGPSDllClose

Declaration:

```
extern long HS_GPSDLL_API HsGPSDllClose(long *rc);
```

Summary:

Closes HsGpsDll library interface and the serial port to which the GPS receiver is connected.

Parameters:

rc

Pointer to long variable to receive function return code (See "Return values" below)

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_FAIL	Failed to close serial communications port
HS_GPSDLL_RC_OK	Success, GPS port is closed

2.1.6 HsGPSDllCloseCs

Declaration:

```
extern int HS_GPSDLL_API HsGPSDllCloseCs(int unused_arg);
```

Summary:

This function is for use from C Sharp.NET applications.

Closes HsGpsDll library interface and the serial port to which the GPS receiver is connected.

Parameters:

unused_arg - Not used, set to 0

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_FAIL	Failed to close serial communications port
HS_GPSDLL_RC_OK	Success, GPS port is closed

2.1.7 HsGPSDLLGetPosition

Declaration:

```
extern long HS_GPSDLL_API HsGPSDLLGetPosition(
    long *lat_deg,    /* latitude degrees */
    long *lat_min,    /* latitude minutes */
    long *lat_dec,    /* latitude decimal part of minutes */
    long *lat_dir,    /* latitude 0 -North, 1 -South */

    long *lon_deg,    /* longitude degrees */
    long *lon_min,    /* longitude minutes */

```

```

    long  *lon_dec,      /* longitude decimal part of minutes */
    long  *lon_dir,      /* longitude 2 -East 3 -West */

    long  *fix_valid,    /* 1 - fix valid, 0 - fix not valid (As received from GPS device) */
    long  *nsat,         /* Number of sattelites used in position fix */

    long  *utc_hour,     /* UTC hour */
    long  *utc_min,     /* UTC minute */
    long  *utc_sec,     /* UTC second */
    long  *utc_msec,    /* UTC millisecond */

    long  *day,         /* day of fix */
    long  *month,       /* month of fix */
    long  *year,        /* year of fix */

    long  *rc           /* result return code */
);

```

Summary:

This function returns last known:

- Geographical position coordinates (latitude and longitude) in degrees and decimal minutes
- Fix quality (valid or not valid)
- Number of satellites in view
- UTC date and time

The latitude and longitude coordinates are returned in the form of integer degrees, integer part of minutes, decimal part of minutes (returned in an integer) and direction code of latitude and longitude.

Example reading:

```

lat_deg = 53 lat_min = 20 lat_dec = 6107 lat_dir = 0 → 53' 20.6107" North
lon_deg = 6 lon_min = 23 lon_dec = 4638 lon_dir = 3 → 6' 23.4638" West

```

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

lat_deg
Pointer to long variable which receives latitude degrees

lat_min
Pointer to long variable which receives latitude minutes (integer part)

lat_dec
Pointer to long variable which receives latitude minutes (decimal part)

lat_dir
Pointer to long variable which receives latitude direction code: 0 –North, 1 -South

lon_deg
Pointer to long variable which receives longitude degrees

lon_min
Pointer to long variable which receives longitude minutes (integer part)

lon_dec

Pointer to long variable which receives longitude minutes (decimal part)

lon_dir

Pointer to long variable which receives longitude direction code: 2 -East, 3 -West

utc_hour

Pointer to long variable which receives UTC hour

utc_min

Pointer to long variable which receives UTC minute

utc_sec

Pointer to long variable which receives UTC second

utc_msec

Pointer to long variable which receives UTC millisecond

day

Pointer to long variable which receives day of current date

month

Pointer to long variable which receives month of current date

year

Pointer to long variable which receives year of current date

rc

Pointer to long variable to receive function return code (See "Return values" below)

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory).
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.1.8 HsGPSDLLGetPositionCs

Declaration:

```
extern int HS_GPSDLL_API HsGPSDLLGetPositionCs(  
    int    *lat_deg,      /* latitude degrees */  
    int    *lat_min,      /* latitude minutes */  
    int    *lat_dec,      /* latitude decimal part of minutes */  
    int    *lat_dir,      /* latitude 0 -North, 1 -South */  
  
    int    *lon_deg,      /* longitude degrees */  
    int    *lon_min,      /* longitude minutes */  
    int    *lon_dec,      /* longitude decimal part of minutes */  
    int    *lon_dir,      /* longitude 2 -East 3 -West */  
  
    int    *fix_valid,    /* 1 - fix valid, 0 - fix not valid (As received from GPS device) */  
    int    *nsat,         /* Number of satellites used in position fix */
```

```

    int    *utc_hour,    /* UTC hour */
    int    *utc_min,    /* UTC minute */
    int    *utc_sec,    /* UTC second */
    int    *utc_msec,   /* UTC millisecond */

    int    *day,        /* day of fix */
    int    *month,     /* month of fix */
    int    *year,       /* year of fix */
);

```

Summary:

This function is for use from C Sharp.NET applications.

This function returns last known:

- Geographical position coordinates (latitude and longitude) in degrees and decimal minutes
- Fix quality (valid or not valid)
- Number of satellites in view
- UTC date and time

The latitude and longitude coordinates are returned in the form of integer degrees, integer part of minutes, decimal part of minutes (returned in an integer) and direction code of latitude and longitude.

Example reading:

lat_deg = 53 lat_min = 20 lat_dec = 6107 lat_dir = 0 → 53' 20.6107" North

lon_deg = 6 lon_min = 23 lon_dec = 4638 lon_dir = 3 → 6' 23.4638" West

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

lat_deg

Pointer to int variable which receives latitude degrees

lat_min

Pointer to int variable which receives latitude minutes (integer part)

lat_dec

Pointer to int variable which receives latitude minutes (decimal part)

lat_dir

Pointer to int variable which receives latitude direction code: 0 –North, 1 –South

lon_deg

Pointer to int variable which receives longitude degrees

lon_min

Pointer to int variable which receives longitude minutes (integer part)

lon_dec

Pointer to int variable which receives longitude minutes (decimal part)

lon_dir

Pointer to int variable which receives longitude direction code: 2 –East, 3 –West

utc_hour
Pointer to int variable which receives UTC hour

utc_min
Pointer to int variable which receives UTC minute

utc_sec
Pointer to int variable which receives UTC second

utc_msec
Pointer to int variable which receives UTC millisecond

day
Pointer to int variable which receives day of current date

month
Pointer to int variable which receives month of current date

year
Pointer to int variable which receives year of current date

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory.
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.1.9 HsGPSDLLGetPositionDegrees

Declaration:

```
extern long HS_GPSDLL_API HsGPSDLLGetPositionDegrees(
    float *latitude,           // returns latitude in decimal degrees
    float *longitude,         // returns longitude in decimal degrees
    char *lat_direction,      // 'N' or 'S' (north, south)
    char *lon_direction,      // 'E' or 'W' (east, west)
    long *rc                   /* result return code */
);
```

Summary:

This function returns last known geographical position coordinates in decimal degree format, for example:

53'20.6096"N = 53.343493 N
6'23.4618"W = 6.39103 W

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

latitude

Pointer to a variable of type float that receives latitude coordinate of current position in decimal degree format

longitude

Pointer to a variable of type float that receives longitude coordinate of current position in decimal degree format

lat_direction

Pointer to char variable to receive direction of latitude, 'N' or 'S' for North or South

lon_direction

Pointer to char variable to receive direction of longitude, 'E' or 'W' for East or West

rc

Pointer to long variable to receive function return code (See "Return values" below)

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory.
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.1.10 HsGPSDLLGetPositionDegreesCs

Declaration:

```
extern int HS_GPSDLL_API HsGPSDLLGetPositionDegrees(  
    unsigned char *lat_str, // returns latitude in decimal degree format as a string. e. g. "53.34349333333335 N"  
    unsigned char *lon_str // returns longitude in decimal degree format as a string. e. g. "6.39103 W"  
);
```

Summary:

This function is for use from C Sharp .NET applications.

This function returns last known geographical position coordinates in decimal degree format as a string. For example:

53'20.6096"N = "53.343493 N"

6'23.4618"W = "6.39103 W"

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

lat_str

Pointer to a string buffer that receives latitude coordinate of current position in decimal degree format as a string. Coordinate is followed by a space (hex 0x20) and a character specifying direction "N" or "S". The string is zero terminated.

lon_str

Pointer to a string buffer that receives longitude coordinate of current position in decimal degree format as a string. Coordinate is followed by a space (hex 0x20) and a character specifying direction “W” or “E”. The string is zero terminated.

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory).
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.1.11 HsGPSDLLGetAltitude

Declaration:

```
extern long HS_GPSDLL_API HsGPSDLLGetAltitude(
    float *altitude,          /* altitude mean sea level in meters */
    char *unit_of_altitude,  /* unit of altitude (e.g. 'M' meters) */
    long *rc                 /* result return code */
);
```

Summary:

This function returns last known altitude (mean sea level)

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

- altitude
Pointer to float variable which receives altitude measured against mean sea level
- unit_of_altitude
Pointer to character to receive measurement unit of antidote. 'M' for meters.
- rc
Pointer to long variable to receive function return code (See “Return values” below)

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory).
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.1.12 HsGPSDLLGetAltitudeCs

Declaration:

```
extern int HS_GPSDLL_API HsGPSDLLGetAltitudeCs(  
    unsigned char *alt_str /* buffer to receive string representing altitude mean sea level level */  
);
```

Summary:

This function is for use from C Sharp .NET applications.
This function returns last known altitude (mean sea level)

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

alt_str

Pointer to string buffer which receives altitude measured against mean sea level as a string. The altitude string is followed by space (hex 0x20) followed by character representing units of altitude (e.g. "M")

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory.
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.1.13 HsGPSDLLGetSpeed

Declaration:

```
extern long HS_GPSDLL_API HsGPSDLLGetSpeed(long *speed_kph, long *rc);
```

Summary:

This function returns last known speed in kilometres per hour

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

speed_kph

Pointer to long variable to receive current last known speed in kilometres per hour.

rc

Pointer to long variable to receive function return code (See "Return values" below)

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters

HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory).
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.1.14 HsGPSDLLGetSpeedCs

Declaration:

```
extern int HS_GPSDLL_API HsGPSDLLGetSpeedCs(long *speed_kph);
```

Summary:

This function is for use from C Sharp .NET applications.

This function returns last known speed in kilometres per hour

GPS receiver normally outputs NMEA messages containing current position data once per second. You can call this function based on a timer event or callback to continuously obtain last known GPS data.

Parameters:

speed_kph

Pointer to int variable to receive current last known speed in kilometres per hour.

Return values:

HS_GPSDLL_RC_NOT_INIT	GPS library not initialised
HS_GPSDLL_RC_INV_PAR	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory).
HS_GPSDLL_RC_NOT_AVAIL	GPS data not yet available
HS_GPSDLL_RC_OK	Success

2.2 Error Codes and Other Constants

2.2.1 HsGpsDll return codes

Name	Value	Comment
HS_GPSDLL_RC_OK	0	Success
HS_GPSDLL_RC_ALREADY_INIT	1	Library already initialised
HS_GPSDLL_RC_FAIL	2	Failure
HS_GPSDLL_RC_NOT_INIT	3	Library not initialised
HS_GPSDLL_RC_NOT_AVAIL	4	GPS data not available
HS_GPSDLL_RC_INV_PAR	5	Invalid parameters
HS_GPSDLL_RC_EVAL_EXCEEDED	6	Evaluation version only. Usage count exceeded. You must restart your application to continue evaluation (this unloads and re-loads HsGpsDll into memory).

2.2.2 Other Constants

Name	Value	Comment
USE_GPGAA	1	Extract coordinates from \$GPGAA message
USE_GPRMC	2	Extract coordinates from \$GPRMC message