MarshallSoft GPS Component

Reference Library

(MGC_REF)

Version 2.2

June 8, 2011.

This software is provided as-is.
There are no warranties, expressed or implied.

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815 USA

email : info@marshallsoft.com
web : www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

The **MarshallSoft GPS Component (MGC**) is a dynamic link library (DLL) (32-bit and 64-bit) which reads and decodes standard GPS (Global Positioning System) NMEA (National Marine Electronics Association) 183 sentences from an RS232 serial port as well as computes great circle distances and bearings.

The **MarshallSoft GPS Component** library can be called from any application capable of calling Windows API functions, including those written in C/C++, .NET, Visual C#, Delphi, Delphi .NET, Visual Basic, Visual Basic .NET, MS Access, MS Excel, MS Word, Fortran, COBOL, PowerBuilder, FoxPro, Power Basic, dBase, Xbase++, etc. We have released versions of the **MarshallSoft GPS Component SDK** for C/C++ (MGC4C) and Visual Basic (MGC4VB). All versions of MGC use the same DLLs (MGC32.DLL and MGC64.DLL).

The latest version of the **MarshallSoft GPS Component** software can be found online at:

http://www.marshallsoft.com/gps-communication-library.htm

The **MarshallSoft GPS Component Reference Manual** (MGC_REF) contains details on each individual **MGC** function.

1.1 General Remarks

All functions return an integer code. Negative values are always errors. See "MGC Error Codes" in Section_4.2. Only return values < MGC_ERROR_BASE (numerical –20000000) are errors.

Note that the **mgcErrorText** function is used to get the text message associated with any error code.

Each function argument is marked as:

- `(I) : 4-byte integer.`
- `(P) : 4-byte pointer.`

Refer to the declaration files (see Section 1.3 below) for the exact syntax of each **MGC** function. Also note that the example programs show exactly how **MGC** functions are called.

1.2 Documentation Set

The complete set of documentation consists of three manuals. This is the third manual (MGC_REF) in the set.

- MGC 4x Programmer's Manual (MGC_4x.PDF)
- MGC User's Manual (MGC_USR.PDF)
- MGC Reference Manual (MGC_REF.PDF)

1.3 Declaration Files

The exact syntax for calling **MGC** functions are specific to the host language (C/C++, Delphi, VB, etc.) and are defined for each language in the "MGC declaration files". Each MGC product comes with the appropriate declaration file for the supported language. For example,

```
MGC4C   C/C++                      MGC.H
MGC4VB  Visual Basic               MGC.BAS
        VBA (EXCEL,ACCESS,etc.)    MGC.BAS
MGC4D   Delphi                     MGC.PAS
```

All **MGC** functions are used in one or more example programs.

1.4 Language Notes

All language versions of **MGC** include the example program MGCVER. Refer to this program and the declaration file as defined in Section 1.3 above to see how **MGC** functions are called. The MGCVER program is also the first program that should be compiled and run.

MGC functions can be called from 32-bit and 64-bit application programs.

2 MGC Functions

2.1 **mgcAltitude** :: Gets the last altitude value.

SYNTAX

```
mgcAltitude()
```

REMARKS

The **mgcAltitude** function returns the last altitude value read provided that GPGGA sentence decoding has been selected. Altitude is returned in units of tenths of meters or tenths of feet. The default is meters.

For example, if the value 124 is returned and the units are meters, then the altitude is 12.4 meters.

EXAMPLE

C/C++ Example

```
// get altitude (in meters) above sea level
Altitude = 10.0 * (double)mgcAltitude();
```

BASIC Example

```
' get altitude (in meters) above sea level
Altitude = 10.0 * CDbl(mgcAltitude())
```

RETURNS

- `Return = 0 :` Altitude in selected units.
- `Return < 0 :` See section 4 for error return codes.

2.2 **mgcAttach** :: Initializes the MGC component.

SYNTAX

```
mgcAttach(KeyCode)

   KeyCode  : (I) Registration key code.
```

REMARKS

The **mgcAttach** function must be the first MGC call made: it establishes communications with the MGC component. The **mgcAttach** function should be called once during the initialization portion of the application code.

When **MGC** is purchased, you will receive a 'KeyCode' which matches the 'KeyCode' within the registered DLL. For the evaluation version, the keycode is 0. See file KEYCODE found in the APPS subdirectory.

EXAMPLE

C/C++ Example

```
unsigned int KeyCode;
KeyCode = 0;
// attach MGC component
Code = mgcAttach(KeyCode);
```

BASIC Example

```
Dim KeyCode As Long
KeyCode = 0
' attach MGC component
Code = mgcAttach(KeyCode)
```

RETURNS

- `Return = 0 :` Altitude in selected units.
- `Return < 0 :` See section 4 for error return codes.

2.3 **mgcBearing** :: Computes the bearing from one point to another.

SYNTAX

```
mgcBearing(Lat1,Lon1,Lat2,Lon2)

   Lat1 : (I) Latitude of first position (Coded Integer form).
   Lon1 : (I) Longitude of first position (Coded Integer form).
   Lat2 : (I) Latitude of second position (Coded Integer form).
   Lon2 : (I) Longitude of second position (Coded Integer form).
```

REMARKS

The **mgcBearing** function returns the bearing from (Lat1, Lon1) to (Lat2, Lon2) in units of hundred-thousandths of a degree. For example, if the value 6589204 is returned, then the bearing is 65.89204 degrees (true).

Refer to the Section_5.1, "Coordinates Used in MGC", in the MarshallSoft GPS Component User's Manual for more details on Coded Integer form.

Note that 0 degrees is due North, 90 degrees is due East, 180 degrees is due South, and 270 degrees is due West.

EXAMPLE

C/C++ Example

```
// Lat1 = Latitude (Coded Integer form)
// Lon1 = Longitude (Coded Integer form)
Angle = mgcBearing(Lat1, -Lon1, ..., ...) / 60000.0;
```

BASIC Example

```
' Lat1 = Latitude (Coded Integer form)
' Lon1 = Longitude (Coded Integer form)
Angle = mgcBearing(Lat1, -Lon1, ..., ...) / 60000.0
```

RETURNS

- `Return > MGC_ERROR_BASE :` Bearing in coded integer form.
- `Return < MGC_ERROR_BASE :` Error. Refer to Section 4 for error return codes.

2.4 **mgcCallback** :: Specifies the callback function for new data.

SYNTAX

```
Code = mgcCallback( fn_addr )

   Fn_addr : (P) Function address.
```

REMARKS

Once registered, the callback function, which is created in the user's program code, will be called by **MGC** whenever there is a new sentence ready to be read.

The **mgcCallback** function is for use by C/C++ programs only. Refer to the CALLBACK.C example program.

EXAMPLE

C/C++ Example

```
void Callback(int SentenceType)
{// lock data buffer
 mgcLockData(1);

 //process sentence here . . .

 // unlock data buffer
 mgcLockData(0);
}

// register callback function
Code = mgcCallback( &Callback );
```

2.5 **mgcClose** :: Closes MGC.

SYNTAX

```
mgcClose()
```

REMARKS

The **mgcClose** function closes the serial port, after which no further processing can occur. **mgcClose** should be called before exiting your application.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
mgcClose()
```

BASIC Example

```
mgcClose()
```

ALSO SEE

`mgcOpen`

2.6 **mgcDecodeDeg** :: Decodes integral degrees from coded integer.

SYNTAX

```
mgcDecodeDeg(iCoded)

   iCoded : (I) Coded degrees >= 0
```

REMARKS

The **mgcDecodeDeg** function extracts integral degrees from a coded integer value. Refer to Section_5.1, "Coordinates Used in MGC" in the MarshallSoft GPS Component User's Manual for more details on Coded Integer format.

Degrees returned will always be positive regardless of the sign of the coded integer.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded);
```

BASIC Example

```
' iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded);
```

ALSO SEE

`mgcDecodeMin, mgcDecodeSec, mgcDecodeTho.`

2.7 **mgcDecodeMin** :: Decodes integral minutes from coded integer..

SYNTAX

```
mgcDecodeMin(iCoded, iDeg)

   iCoded : (I) Coded degrees >= 0
   iDeg   : (I) Integral degrees as returned by mgcDecodeDeg()
```

REMARKS

The **mgcDecodeMin** function extracts integral minutes from a coded integer value. Refer to Section_5.1, "Coordinates Used in MGC" in the MarshallSoft GPS Component User's Manual for more details on Coded Integer format.

Minutes returned will always be positive regardless of the sign of the coded integer.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded);
iMin = mgcDecodeMin(iCoded, iDeg);
```

BASIC Example

```
' iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded)
iMin = mgcDecodeMin(iCoded, iDeg)
```

ALSO SEE

`mgcDecodeDeg, mgcDecodeSec, mgcDecodeTho.`

2.8 **mgcDecodeSec** :: Decodes integral seconds from coded integer.

SYNTAX

```
mgcDecodeSec(iCoded, iDeg)

   iCoded : (I) Coded degrees >= 0
   iDeg   : (I) Integral degrees as returned by mgcDecodeDeg()
   iMin   : (I) Integral minutes as returned by mgcDecodeMin()
```

REMARKS

The **mgcDecodeSec** function extracts integral seconds from a coded integer value. Refer to Section_5.1, "Coordinates Used in MGC" in the MarshallSoft GPS Component User's Manual for more details on Coded Integer format.

Seconds returned will always be positive regardless of the sign of the coded integer.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded);
iMin = mgcDecodeMin(iCoded, iDeg);
iSec = mgcDecodeSec(iCoded, iDeg, iMin);
```

BASIC Example

```
' iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded)
iMin = mgcDecodeMin(iCoded, iDeg)
iSec = mgcDecodeSec(iCoded, iDeg, iMin)
```

ALSO SEE

`mgcDecodeDeg, mgcDecodeMin, mgcDecodeTho.`

2.9 **mgcDecodeTho** :: Decodes integral decimal minutes (*1000).

SYNTAX

```
mgcDecodeTho(iCode, iDeg, iMin)

   iCoded : (I) Coded degrees >= 0
   iDeg   : (I) Integral degrees as returned by mgcDecodeDeg()
   iMin   : (I) Integral minutes as returned by mgcDecodeMin()
```

REMARKS

The **mgcDecodeTho** function extracts integral thousandths of a minute from a coded integer value. Refer to Section_5.1, "Coordinates Used in MGC" in the MarshallSoft GPS Component User's Manual for more details on Coded Integer format.

Thousandths returned will always be positive regardless of the sign of the coded integer.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded);
iMin = mgcDecodeMin(iCoded, iDeg);
iTho = mgcDecodeTho(iCoded, iDeg, iMin);
```

BASIC Example

```
' iCoded = integer coded value
iDeg = mgcDecodeDeg(iCoded);
iMin = mgcDecodeMin(iCoded, iDeg);
iTho = mgcDecodeTho(iCoded, iDeg, iMin);
```

ALSO SEE

`mgcDecodeDeg, mgcDecodeMin, mgcDecodeSec.`

2.10 **mgcEncodeDMS** :: Encodes (Deg, Min, Sec) to Coded Integer

SYNTAX

```
mgcEncodeDMS(Deg, Min, Sec)

   iDeg : (I) Integral degrees >= 0
   iMin : (I) Integral minutes >= 0
   iTho : (I) Integral thousandths >= 0
```

REMARKS

The **mgcEncodeDMS** function encodes degrees, minutes, and seconds to a coded integer value. Refer to Section_5.1, "Coordinates Used in MGC" in the MarshallSoft GPS Component User's Manual for more details on Coded Integer format.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// encode (deg, min, sec)
iCoded = mgcEncodeDMS(iDeg,iMin,iSec);
```

BASIC Example

```
' encode (deg, min, sec)
iCoded = mgcEncodeDMS(iDeg,iMin,iSec)
```

ALSO SEE

`mgcEncodeDMT`

2.11 **mgcEncodeDMT** :: Encodes (Deg, Min, Tho) to Coded Integer

SYNTAX

```
mgcEncodeDMT(Deg, Min, Tho)

   iDeg : (I) Integral degrees >= 0
   iMin : (I) Integral minutes >= 0
   iTho : (I) Integral thousandths >= 0
```

REMARKS

The **mgcEncodeDMT** function encodes degrees, minutes, and thousandths to a coded integer value. Refer to Section_5.1, "Coordinates Used in MGC" in the MarshallSoft GPS Component User's Manual for more details on Coded Integer format.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// encode (deg, min, tho)
iCoded = mgcEncodeDMS(iDeg,iMin,iTho);
```

BASIC Example

```
' encode (deg, min, tho)
iCoded = mgcEncodeDMS(iDeg,iMin,iTho)
```

ALSO SEE

`mgcEncodeDMS`

2.12 **mgcErrorText** :: Get text of error message.

SYNTAX

```
mgcErrorText(ErrCode, Buffer)

   ErrCode : (I) Error code.
    Buffer : (P) Buffer for error message.
```

REMARKS

The **mgcErrorText** function formats the error message for error 'ErrCode' in 'Buffer'.

All MGC functions return an integer value. If this return value is less than MGC_ERROR_BASE (numerical –20000000), then an error has occurred.

Call the mgcErrorText function when an error is returned from a **MGC** function so that the text of the error message can be displayed or logged.

RETURNS

- `Return > 0 :` Number of characters in 'Buffer'.
- `Return < MGC_ERROR_BASE :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
int ErrCode;
char ErrBuffer[128];
mgcErrorText(ErrCode, (char *)ErrBuffer);
printf("%s\n", ErrBuffer);
```

BASIC Example

```
Dim ErrCode As Long
Dim ErrBuffer As String * 128
Code = mgcErrorText(ErrCode, ErrBuffer)
PRINT ErrBuffer
```

ALSO SEE

`mgcOpen`

2.13 **mgcGetData** ::Gets data sentence field value.

SYNTAX

```
mgcGetData(DataID, Buffer)

   DataID : (I) Field name.
   Buffer : (P) Buffer for field text data.
```

REMARKS

The **`mgcGetData`** function returns the value of the field specified by DataID as specified in Section 6 of the MGC User's Manual (MGC_USR). Only GPGSA, GPGGA, GPRMC, GPGLL, GPGSV, GPBOD, GPWPL and GPVTG sentences have DataID fields defined.

Note that for any sentence, individual fields are numbered beginning with 0. Therefore, to download the data for the third field of the current data record, specify DataID = 2.

The buffer passed must be able to hold at least 12 bytes.

RETURNS

- `Return >= 0 :` The number of times this data record has been referenced.
- `Return < 0 :` Error. Refer to Section 4 for error return codes..

EXAMPLE

C/C++ Example

```
char DataBuffer[12];
RefCount = mgcGetData(GPRMC_DATE,(LPSTR)DataBuffer);
```

BASIC Example

```
Dim DataBuffer As String * 12
RefCount = mgcGetData(GPRMC_DATE, DataBuffer)
```

ALSO SEE

mgcGetString

2.14 **mgcGetInteger** :: Gets MGC integer parameter for GPS processing.

SYNTAX

```
mgcGetInteger(ParmName)

   ParmName : (I) Parameter name.
```

REMARKS

The **mgcGetInteger** function returns an integer parameter, depending on the parameter name passed, as follows:

```
[Parameter Name]            [Returns]
MGC_GET_VERSION              3 part (X.Y.Z) packed version number.
MGC_GET_BUILD                Build number
MGC_GET_VERSION_DIGIT1       First digit of version number.
MGC_GET_VERSION_DIGIT2       Second digit of version number.
MGC_GET_VERSION_DIGIT3       Third digit of version number.
MGC_GET_BYTES_RECEIVED       Total number of bytes received.
MGC_GET_LINES_RECEIVED       Total number of lines received.
MGC_GET_READY_BUFFER_COUNT   Number times current data record has been referenced.
MGC_GET_STRING_LENGTH        Number of bytes in 'ready' buffer.
MGC_GET_LINE_NUMBER          Current (sentence) line count.
```

RETURNS

- `Return > 0 :` Requested parameter defined above.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// get MGC build number
Build = mgcGetInteger(MGC_GET_BUILD);
```

BASIC Example

```
' get MGC build number
Build = mgcGetInteger(MGC_GET_BUILD)
```

ALSO SEE

mgcSetInteger

2.15 **mgcGetSentenceType** :: Gets the GPS NMEA sentence type.

SYNTAX

```
mgcGetSentenceType()
```

REMARKS

The **mgcGetSentenceType** function returns the **MGC** sentence type. This is necessary when multiple sentence types are specified as in the GGA_RMC example program.

Sentence types are defined [in MGC.H (C/C++), MGC32.BAS (VB), and MGC32.PAS (Delphi)] as:

```
MGC_SENTENCE_GPRMC = 101
MGC_SENTENCE_GPGGA = 102
MGC_SENTENCE_GPGLL = 103
MGC_SENTENCE_GPGSV = 104
MGC_SENTENCE_GPVTG = 105
MGC_SENTENCE_GPBOD = 106
MGC_SENTENCE_GPWPL = 107
MGC_SENTENCE_GPGSA = 109
```

RETURNS

- `Return > 0 :` Sentence type as defined above.
- `Return < 0 :` Error. No sentence type

EXAMPLE

C/C++ Example

```
int SentenceType
// get sentence type
SentenceType = mgcGetSentenceType();
```

BASIC Example

```
Dim SentenceType As Long
// get sentence type
SentenceType = mgcGetSentenceType()
```

ALSO SEE

mgcCallback

2.16 **mgcGetString** :: Gets MGC string parameter for GPS processing.

SYNTAX

```
mgcGetString(ParmName, Buffer, BufLen)

   ParmName : (I) Parameter.
   Buffer   : (P) Buffer pointer.
   BufLen   : (I) Length of above buffer.
```

REMARKS

The **mgcGetString** function returns the string corresponding to the parameter name specified, a follows:

```
[Parameter Name]       [String Returned]

MGC_GET_STRING         Gets (current) undecoded NMEA 183 sentence.
MGC_GET_REGISTRATION   Gets the registration string from MGC32.DLL.
```

Refer to the RAW example program.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
char DataBuffer[256];
// get undecoded GPS sentence
Code = mgcGetString(MGC_GET_STRING, (char *)DataBuffer, 255);
```

BASIC Example

```
Dim DataBuffer As String * 255
' get undecoded GPS sentence
Code = mgcGetString(MGC_GET_STRING, DataBuffer, 255)
```

ALSO SEE

mgcGetData

2.17 **mgcGreatCircle** :: Computes distance between two points.

SYNTAX

```
mgcGreatCircle(Lat1,Lon1,Lat2,Lon2)

   Lat1 : (I) Latitude of first position (Coded Integer form).
   Lon1 : (I) Longitude of first position (Coded Integer form).
   Lat2 : (I) Latitude of second position (Coded Integer form).
   Lon2 : (I) Longitude of second position (Coded Integer form).
```

REMARKS

The **mgcGreatCircle** function computes the great circle distance between two points of the surface of the earth. The latitude and longitude values are passed in units of hundred-thousandths of a degree.

Refer to the Section_5.1, "Coordinates Used in MGC", in the MarshallSoft GPS Component User's Manual (mgc_usr.pdf) for more details on Coded Integer form.

RETURNS

- `Return > 0 :` Distance along surface of Earth (default units are meters).
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
Lat1 = 3450583   // Coded Integer form (34.50582 deg N)
Lon1 = -8691777  // Coded Integer form (86.91777 deg W)
Distance = mgcGreatCircle(Lat1, Lon1, ..., ...);
```

BASIC Example

```
Lat1 = 3450583   ' Coded Integer form (34.50582 deg N)
Lon1 = -8691777  ' Coded Integer form (86.91777 deg W)
Distance = mgcGreatCircle(Lat1, Lon1, ..., ...)
```

2.18 **mgcLatitude** :: Gets last latitude value.

SYNTAX

```
mgcLatitude()
```

REMARKS

The **mgcLatitude** function returns the last latitude value read provided that GPGGA, GPRMC, or GPGLL sentence decoding has been selected.

Latitude is returned in “coded integer form”. Call the functions mgcDecodeDeg(),mgcDecodeMin(), and mgcDecodeSec() to extract degrees, minutes, and seconds from the coded integer returned from mgcGetLatitude.

Refer to the Section_5.1, "Coordinates Used in MGC" in the MarshallSoft GPS Component User's Manual for more details on Coded Integer form.

South latitude is negative.

RETURNS

- `Return > MGC_ERROR_BASE :` Latitude in coded integer form.
- `Return < MGC_ERROR_BASE :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// get current latitude
LatCode = mgcLatitude();
Degrees = mgcDecodeDeg(LatCode);
Minutes = mgcDecodeMin(LatCode, Degrees);
Seconds = mgcDecodeSec(LatCode, Degrees, Minutes);
```

BASIC Example

```
' get current latitude
LatCode = mgcLatitude()
Degrees = mgcDecodeDeg(LatCode)
Minutes = mgcDecodeMin(LatCode, Degrees)
Seconds = mgcDecodeSec(LatCode, Degrees, Minutes)
```

ALSO SEE

`mgcLongitude`

2.19 **mgcLockData**:: Locks current data record.

SYNTAX

```
mgcLockData(Flag)

   Flag : (I) Lock Flag (TRUE/FALSE).
```

REMARKS

The **mgcLockData** function locks (if Flag is not zero) or unlocks (if Flag is 0) the current data record so that the field data may be copied.

Normally, the data record should be locked just prior to reading individual fields, then unlocked afterwards.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// lock current sentence
mgcLockData(1);
// unlock current sentence
mgcLockData(0);
```

BASIC Example

```
' lock current sentence
mgcLockData(1)
' unlock current sentence
mgcLockData(0)
```

2.20 **mgcLongitude** :: Gets current longitude value..

SYNTAX

```
mgcLongitude()
```

REMARKS

The **mgcLongitude** function returns the last longitude value read provided that GPGGA, GPRMC, or GPGLL sentence decoding has been selected.

Longitude is returned in “coded integer form”. Call the functions mgcDecodeDeg(), mgcDecodeMin(), and mgcDecodeSec() to extract degrees, minutes, and seconds from the coded integer returned from mgcGetLongitude.

Refer to the Section_5.1, "Coordinates Used in MGC", in the MarshallSoft GPS Component User's Manual for more details on Coded Integer form.

West longitude is negative.

RETURNS

- `Return > MGC_ERROR_BASE :` Longitude in coded integer form.
- `Return < MGC_ERROR_BASE :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
     // get current Longitude
     LatCode = mgcLongitude();
     Degrees = mgcDecodeDeg(LatCode);
     Minutes = mgcDecodeMin(LatCode, Degrees);
     Seconds = mgcDecodeSec(LatCode, Degrees, Minutes);
```

BASIC Example

```
     // get current longitude
//$  LonCode = mgcLongitude()
     Degrees = mgcDecodeDeg(LonCode)
     Minutes = mgcDecodeMin(LonCode, Degrees)
     Seconds = mgcDecodeSec(LonCode, Degrees, Minutes)
```

ALSO SEE

`mgcLatitude`

2.21 **mgcNewLatitude** :: Computes new latitude.

SYNTAX

```
mgcNewLatitude(Lat, Lon, Course, Dist)

   Lat    : (I) initial latitude (Coded Integer form)
   Lon    : (I) initial longitude (Coded Integer form)
   Course : (I) Course to follow (Coded Integer form)
   Dist   : (I) Great circle distance (meters)
```

REMARKS

The **mgcNewLatitude** function returns the latitude of the position on the earth's surface that is computed from the initial latitude and longitude following the specified course and distance.

Latitude is specified in units of hundred-thousandths of a degree. For example, 34.50583 degrees is specified by the integer 3450583. Note that error codes are always less than MGC_ERROR_BASE (numerical –20000000).

Refer to the Section 5.1, "Coordinates Used in MGC", in the MarshallSoft GPS Component User's Manual for more details on Coded Integer form.

RETURNS

- `Return > MGC_ERROR_BASE :` Latitude in coded integer form.
- `Return < MGC_ERROR_BASE :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// compute new latitude
Code = mgcNewLatitude(3450582, -8677000, 4500000, 1000);
```

BASIC Example

```
' compute new latitude
Code = mgcNewLatitude(3450582, -8677000, 4500000, 1000)
```

ALSO SEE

`mgcGetLatitude`

2.22 **mgcNewLongitude** :: Computes new longitude.

SYNTAX

```
mgcNewLongitude(Lat, Lon, Course, Dist)

   Lat    : (I) initial latitude (Coded Integer form)
   Lon    : (I) initial longitude (Coded Integer form)
   Course : (I) Course to follow (Coded Integer form)
   Dist   : (I) Great circle distance (meters)
```

REMARKS

The **mgcNewLongitude** function returns the longitude of the position on the earth's surface that is computed from the initial latitude and longitude following the specified course and distance.

Longitude is specified units of hundred-thousandths of a degree. For example, 34.50583 degrees is specified by the integer 3450583. Note that error codes are always less than MGC_ERROR_BASE (numerical –20000000).

Refer to the Section 5.1, "Coordinates Used in MGC", in the MarshallSoft GPS Component User's Manual (mgc_usr.pdf) for more details on Coded Integer form.

RETURNS

- `Return > MGC_ERROR_BASE :` Longitude in coded integer form.
- `Return < MGC_ERROR_BASE :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// compute new longitude
Code = mgcNewLongitude(3450582, -8677000, 4500000, 1000);
```

BASIC Example

```
' compute new longitude
Code = mgcNewLongitude(3450582, -8677000, 4500000, 1000)
```

ALSO SEE

`mgcGetLatitude`

2.23 **mgcOpen**:: Open MGC serial port.

SYNTAX

```
mgcOpen(Port)

   Port : (I) Port selected.
```

REMARKS

The **mgcOpen** function opens the specified serial port at 4800 baud in preparation for receiving MNEA sentences. **mgcAttach** must be called before **mgcOpen**. Ports COM1 through COM256 can be specified.

The default serial port parameters

```
mgcSetInteger(MGC_SET_BAUDRATE, 4800);
mgcSetInteger(MGC_SET_PARITY,   MGC_NoParity);
mgcSetInteger(MGC_SET_STOPBITS, MGC_OneStopBit);
mgcSetInteger(MGC_SET_DATABITS, MGC_DataBits8);
```

can be changed before opening the serial port.

Be sure to check the return code from **mgcOpen**().

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// open COM1 for NMEA sentence input
Code = mgcOpen(MGC_COM1);
```

BASIC Example

```
' open COM1 for NMEA sentence input
Code = mgcOpen(MGC_COM1)
```

ALSO SEE

`mgcClose`

2.24 **mgcRelease**:: Release MGC DLL.

SYNTAX

```
mgcRelease()
```

REMARKS

The **mgcRelease** function releases MGC32.DLL (or MGC64.DLL for 64-bit executables).

The MGC DLL (MGC32.DLL or MGC64.DLL) is normally released when your executable terminates.

The primary purpose of **mgcRelease** is to release MGC32.DLL (or MGC64.DLL for 64-bit executables) so that **mgcAttach** can be called again in the same program.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// release MGC32.DLL
Code = mgcRelease();
```

BASIC Example

```
' release MGC32.DLL
Code = mgcRelease()
```

ALSO SEE

`mgcAttach`

2.25 **mgcSeaLevel** :: Gets last sea level value.

SYNTAX

`mgcSeaLevel()`

REMARKS

The **mgcSeaLevel** function returns the last sea level value read provided that GPGGA sentence decoding has been selected.

Sea level altitude is returned in units of tenths of meters or tenths of feet. The default is meters.

RETURNS

- `Return = 0 :` Altitude above sea level (default tenths of meters).
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// get sea level in meters
Code = mgcSeaLevel();
SeaLevel = 10.0 * (double)Code;
```

BASIC Example

```
'get sea level in meters
Code = mgcSeaLevel();
SeaLevel = 10.0 * CDbl(Code)
```

ALSO SEE

mgcAltitude

2.26 **mgcSetInteger** :: Sets MGC integer parameter for GPS processing.

SYNTAX

```
mgcSetInteger(ParmName, ParmValue)

   ParmName   : (I) Parameter name
   ParmValue  : (I) Parameter value
```

REMARKS

The **mgcSetInteger** function sets a **MGC** parameter according to the following table.

```
[Parameter Name]         [Parameter Value]             [Description]
MGC_SET_BAUDRATE         4800,9600, . . . ,115200.     Specify baud rate (default 4800).
MGC_SET_SENTENCE_TYPE    Sets the sentence type:
                              MGC_SENTENCE_RAW         All sentences.
                              MGC_SENTENCE_GPRMC       GPRMC sentences.
                              MGC_SENTENCE_GPGGA       GPGGA sentences.
                              MGC_SENTENCE_GPGLL       GPGLL sentences.
                              MGC_SENTENCE_GPGSV       GPGSV sentences.
                              MGC_SENTENCE_GPBOD       GPBOD sentences.
                              MGC_SENTENCE_GPWPL       GPWPL sentences.
MGC_SET_ALTITUDE_UNIT    Sets the altitude units:
                              MGC_ALTITUDE_IN_FEET     Feet
                              MGC_ALTITUDE_IN_METERS   Meters
MGC_SET_DISTANCE_UNIT    Sets the distance units
                              MGC_DISTANCE_IN_FEET     Feet
                              MGC_DISTANCE_IN_METERS   Meters
                              MGC_DISTANCE_IN_KM       Kilometers
                              MGC_DISTANCE_IN_SM       Statue miles
                              MGC_DISTANCE_IN_NM       Nautical miles
```

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes.

EXAMPLE

C/C++ Example

```
// specify RAW sentences
Code = mgcSetInteger(MGC_SET_SENTENCE_TYPE, MGC_SENTENCE_RAW);
```

BASIC Example

```
' specify RAW sentences
Code = mgcSetInteger(MGC_SET_SENTENCE_TYPE, MGC_SENTENCE_RAW)
```

ALSO SEE

mgcGetInteger

2.27 **mgcSleep** :: Sleeps specified time.

SYNTAX

```
mgcSleep(Time)

   Time   : (I) Time in milliseconds to sleep.
```

REMARKS

The **mgcSleep** function sleeps (approximately) for the number of milliseconds specified.

RETURNS

`TRUE (1)`.

EXAMPLE

C/C++ Example

```
/* sleep one second */
mgcSleep(1000);
```

BASIC Example

```
' sleep one second
N = mgcSleep(1000)
```

2.28 **mgcStatus** :: Return Data Status.

SYNTAX

```
mgcStatus()
```

REMARKS

The **mgcStatus** function returns TRUE if GPS serial data is being received and FALSE if serial data is not being received.

RETURNS

TRUE if GPS serial data is being received, else FALSE is returned.

EXAMPLE

C/C++ Example

```
/* are we still running ? */
if(!mgcStatus())
  {
   // not running
  }
```

BASIC Example

```
' are we still running ?
N = mgcStatus()
If N = 0 Then
   ' not running
End if
```

2.29 **mgcSysTime** :: Gets the system clock time.

SYNTAX

```
mgcSysTime()
```

REMARKS

The **mgcSysTime** function returns the number of milliseconds since bootup. This is the same value as returned by the **GetCurrentTime**() Win32 API function in Windows.

RETURNS

Number of milliseconds since bootup.

EXAMPLE

C/C++ Example

```
/* get system clock time */
Time = mgcSysTime();
```

BASIC Example

```
' get system clock time
Time = mgcSysTime ()
```

2.30 **mgcTimestamp** :: Gets the last timestamp value.

SYNTAX

```
mgcTimestamp()
```

REMARKS

The **mgcTimestamp** function returns the last timestamp value read provided that GPGGA or GPRMC sentence decoding has been selected.

RETURNS

- `Return = 0 :` No error.
- `Return < 0 :` Error. Refer to Section 4 for error return codes

EXAMPLE

C/C++ Example

```
unsigned long N;
// get system timestamp
N = mgcTimestamp();
```

BASIC Example

```
Dim N As Long
' get system timestamp
N = mgcTimestamp()
```

3 Sentence Field Codes

The numeric field values for all predefines sentence types follow:

3.1 GPRMC: Recommended minimum specific GPS/Transit data.

```
GPRMC_UTC_TIME          0
GPRMC_VALIDITY          1
GPRMC_LATITUDE          2
GPRMC_NORTH_SOUTH       3
GPRMC_LONGITUDE         4
GPRMC_EAST_WEST         5
GPRMC_SPEED             6
GPRMC_COURSE            7
GPRMC_DATE              8
GPRMC_VARIATION         9
GPRMC_VAR_EW           10
GPRMC_CHECKSUM         11
```

3.2 GPGGA: Global Positioning System Fix Data.

```
GPGGA_UTC_TIME          0
GPGGA_LATITUDE          1
GPGGA_NORTH_SOUTH       2
GPGGA_LONGITUDE         3
GPGGA_EAST_WEST         4
GPGGA_QUALITY           5
GPGGA_SATELLITES        6
GPGGA_HDOP              7
GPGGA_ALTITUDE          8
GPGGA_ALT_UNITS         9
GPGGA_GEOID            10
GPGGA_AGE              11
GPGGA_GEO_UNITS        12
GPGGA_GEO_REF_ID       13
GPGGA_CHECKSUM         14
```

3.3 GPGLL: Geographic position, latitude / longitude.

```
GPGLL_LATITUDE          0
GPGLL_NORTH_SOUTH       1
GPGLL_LONGITUDE         2
GPGLL_EAST_WEST         3
GPGLL_FIX               4
GPGLL_VALIDITY          5
```

3.4 GPGSV: GPS Satellites in view.

```
GPGSV_NUMBER_SATS      0
GPGSV_SENTENCE_NBR     1
GPGSV_SATS_IN_VIEW     2
GPGSV_SAT_NUMBER_1     3
GPGSV_ELEVATION_1      4
GPGSV_AZIMUTH_1        5
GPGSV_SIGNAL_1         6
GPGSV_SAT_NUMBER_2     7
GPGSV_ELEVATION_2      8
GPGSV_AZIMUTH_2        9
GPGSV_SIGNAL_2        10
GPGSV_SAT_NUMBER_3    11
GPGSV_ELEVATION_3     12
GPGSV_AZIMUTH_3       13
GPGSV_SIGNAL_3        14
GPGSV_SAT_NUMBER_4    15
GPGSV_ELEVATION_4     16
GPGSV_AZIMUTH_4       17
GPGSV_SIGNAL_4        18
```

3.5: GPVTG: Track made good and ground speed.

```
GPVTG_COURSE             0
GPVTG_TRUE_OR_MAGNETIC   1
GPVTG_2_NOT_USED         2
GPVTG_3_NOT_USED         3
GPVTG_GROUND_SPEED_KNOTS 4
GPVTG_EXPECT_N           5
GPVTG_GROUND_SPEED_KPH   6
GPVTG_EXPECT_K           7
```

3.6 $GPBOD: Bearing, origin to destination.

```
GPBOD_BEARING_TRUE       0
GPBOD_BEARING_T          1
GPBOD_BEARING_MAGNETIC   2
GPBOD_BEARING_M          3
GPBOD_DEST_ID            4
GPBOD_START_ID           5
```

3.7 GPWPL: Waypoint location

```
GPWPL_LATITUDE          0
GPWPL_NORTH_SOUTH       1
GPWPL_LONGITUDE         2
GPWPL_EAST_WEST         3
GPWPL_NAME              4
```

3.8 GPGSA: Overall Satellite Data

```
GPGSA_MODE1     0
GPGSA_MODE2     1
GPGSA_ID1       2
GPGSA_ID2       3
GPGSA_ID3       4
GPGSA_ID4       5
GPGSA_ID5       6
GPGSA_ID6       7
GPGSA_ID7       8
GPGSA_ID8       9
GPGSA_ID9      10
GPGSA_ID10     11
GPGSA_ID11     12
GPGSA_ID12     13
GPGSA_PDOP     14
GPGSA_HDOP     15
GPGSA_VDOP     16
```

4 Error Codes

4.1 Serial Port Error Codes

[NAME]	[VALUE]	[FUNCTION]
WSC_NO_DATA	-20000200	No data available.
WSC_RANGE	-20000200	Parameter not in legal range.
WSC_ABORTED	-20000202	Internal checksum error.
WSC_WIN32ERR	-20000203	Win32 API error.
WSC_BUFFERS	-20000205	Cannot allocate dynamic memory.
WSC_THREAD	-20000206	Cannot start Win32 thread.
WSC_IE_BADID	-1	Bad port ID.
WSC_IE_OPEN	-2	Port already open.
WSC_IE_NOPEN	-3	Port not open.
WSC_IE_MEMORY	-4	Dynamic memory error.
WSC_IE_DEFAULT	-5	Error in default parameters.
WSC_IE_HARDWARE	-10	Hardware error.
WSC_IE_BYTESIZE	-11	Byte size not supported.
WSC_IE_BAUDRATE	-12	Baud rate not supported.

4.2 MGC Error Codes

[NAME]	[VALUE]	[FUNCTION]
MGC_PORT_ALREADY_OPEN	-20000101	Port is already open.
MGC_PORT_NOT_OPEN	-20000102	Port is not open.
MGC_CANNOT_START_THREAD	-20000103	Error starting thread.
MGC_NO_SUCH_FIELD	-20000104	No such field.
MGC_NO_SUCH_PARAMETER	-20000105	No such parameter.
MGC_NO_SUCH_SENTENCE	-20000106	No such sentence type.
MGC_BAD_DIGIT	-20000107	Bad digit found.
MGC_NO_DATA	-20000108	No data is available.
MGC_NO_SUCH_UNIT	-20000109	No such unit.
MGC_ABORTED	-20000110	MGC aborted.
MGC_BUFFER_TOO_SMALL	-20000111	Buffer too small for data.
MGC_EXPIRED	-20000112	Evaluation version expired.
MGC_CANNOT_ATTACH	-20000113	Cannot attach MGC.
MGC_ALREADY_ATTACHED	-20000114	MGC already attached.
MGC_NOT_ATTACHED	-20000115	MGC not attached.
MGC_KEYCODE	-20000116	Bad key code.

[END]