MarshallSoft GPS Component

for C/C++

Programmer's Manual

(MGC4C)

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

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

TABLE OF CONTENTS

1 Introduction

The **MarshallSoft GPS Component for C/C++ (MGC4C)** library is a toolkit that allows software developers to quickly develop GPS applications in C/C++, Visual C++, Visual C# and .NET.

The **MarshallSoft GPS Component (MGC)** is a library of functions providing direct and simple control of GPS data. A straightforward interface provides the functionality to read and decode standard GPS (Global Positioning System) NMEA 183 sentences from the RS232 serial port as well as compute great circle distances and bearings.

This **MarshallSoft GPS Component Programmer's Manual** provides information needed to compile and run programs in a C/C++ environment.

The **MarshallSoft GPS Component for C/C++** toolkit supports and has been tested with C/C++, Microsoft Visual C++, Visual Studio .NET Framework (Visual C++ .NET, C# ,NET), Visual C++ Express, Borland C/C++, Borland Turbo C++ for Windows, Borland C++ Builder, Watcom C/C++, MinGW C++ and LCC-Win32 C compilers. MGC4C can also be used with most other C/C++ Windows compilers. Registered users can statically link **MGC** with their application rather than using the MGC DLLs.

MGC4C runs under all versions of Windows (Windows 95, Windows 98, Windows ME, Windows 2000, Windows 2003, Windows NT, Windows XP, Windows 7, Vista and x64). The MarshallSoft GPS Component SDK DLLs (MGC32.DLL and MGC64.DLL) can also be used from any language (Visual Basic, VB.NET, Codegear Delphi, Visual FoxPro, COBOL, Xbase++, dBase, Microsoft Office, etc.) capable of calling the Windows API. Win32 and Win64 DLLs are provided.

MGC4C includes more than 15 C/C++ example programs with full source code demonstrating MGC functions to read and decode GPS data. Microsoft Foundation Class (MFC) and Borland C++ Builder (BCB) examples are also included.

Our goal is to provide a robust serial communication component library that you and your customers can depend upon. A fully functional evaluation version is available. Contact us if you have any questions.

When comparing the **MarshallSoft GPS Component Library** against our competition, note that:

1. MGC4C is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. WIN32 and WIN64 DLLs are included.
3. MGC4C does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
4. MGC is fully threadable.
5. The MGC functions can be called from applications not capable of using controls.

We also provide a version of the **MarshallSoft GPS Component Library** for Visual Basic (MGC4VB. Each version of MGC uses the same DLLs (MGC32.DLL and MGC64.DLL). However, the examples provided for each version are written for the specified computer language. We also have declaration files for other programming languages. MarshallSoft provides a separate GPS component for WIN/CE for eVC (MGC4eVC).

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

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

1.1 Features

Some of the many features of the **MarshallSoft GPS Component (MGC)** library toolkit are:

- Supports both 32-bit and 64-bit Windows.
- Uses the Windows API for serial port input. No special Windows driver is required.
- Runs as a background thread unattended. MGC is fully thread safe.
- The most current GPS data is always available on demand.
- Includes support functions such as **mgcGreatCircle.**
- Can read any GPS NMEA 183 sentence, extracting each field
- Can read and decode GPGGA, GPRMC, GPGLL,GPGSA, GPVTG, GPBOD, GPWPL and GPGSV navigation sentences.
- Can graphically display latitude and longitude.
- Can compute distances and bearings.
- Provides ability to convert units.
- Works with Bluetooth serial.
- Work with USB ports that have a "USB to Serial Port" converter cable.

- License includes one year of technical support and downloadable updates.
- Royalty free distribution with your compiled application.
- Evaluation versions are fully functional. No unlock code is required.
- Can be used from GUI mode or console mode programs.
- Supports Windows 95/98/Me/NT/2000/2003/XP/Vista/7.
- **MGC4C** is implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Works with all versions of Microsoft Visual C++ (v4.0 through Visual Studio 2010.)
- Works with Borland C/C++ (v5.0, v5.5) and Borland C++ Builder.
- Works with Microsoft Foundation Class, Watcom v11, MinGW and LCC-WIN32.
- Can be called from any program that is capable of calling the Windows API.
- **MGC** functions can be called directly from both ANSI C programs and from C++ programs.
- The license can be used with all supported computer programming languages.
- Will run on machines with or without .NET installed
- Does **not** depend on support libraries. Makes calls to core Windows API functions only.
- Can be used with any application program (in any language) capable of calling Windows API functions including C/C++, Visual Basic, PowerBASIC, Delphi, Visual FoxPro, Visual dBase, Xbase++, Fortran, VBA, etc.
- Can be purchased with (or without) source code.
- Documentation online as well as in printable format.

A good selection of C/C++ example programs with full source code is included. Refer to Section 6 for more details on each of the example programs.

[PROGRAM]	[DESCRIPTION]
BCB_PRJ:	Borland C++ Builder version of LATLON.C
COMPUTE:	Computes distances and bearings.
CALLBACK:	Example program that uses mgcCallback.
CONVERT:	Converts between (deg,min,min/100), (deg,min,sec), etc.
CS_LATLON:	Microsoft Visual C# version of LATLON.C
CS_VERS:	Microsoft Visual C# .NET example program similar to MGCVER.
EXAMPLE:	Displays latitude & longitude in decimal degrees.
GGA_RMC:	Displays both GSA and RMC sentences.
GPBOD:	Program that decodes GPBOD sentences.
GPGGA:	Program that decodes GPGGA sentences.
GPGLL:	Program that decodes GPGLL sentences.
GPGSA:	Program that decodes GPGSA sentences.
GPGSV:	Program that decodes GPGSV sentences.
GPRMC:	Program that decodes GPRMC sentences.
GPVTG:	Program that decodes GPVTG sentences.
GPWPL:	Program that decodes GPWPL sentences.
HELLO:	Program similar to MGCVER but uses the MGC class fMGC.
LATLON:	Program that graphically displays latitude & longitude.
LOADLIB:	Loads MGC32.DLL or MGC64.DLL from specified directory.
MFC_PGM:	Microsoft Foundation Class version of LATLON.
MGCVER:	Program that displays the MGC4C version & build numbers.
RAW:	Program that decodes all sentences.
TRANS:	Translates between various forms of latitude & longitude.
VC_CGA:	Microsoft VC .NET example program similar to GPGGA.
VC_LATLON:	Microsoft VC .NET version of LATLON.C
VC_VERS:	Microsoft VC .NET example program similar to MGCVER.

1.2 Documentation Set

The complete set of documentation consists of three manuals. This is the first manual (MGC_4C) in the set.

- MGC 4C Programmer's Manual (MGC_4C.PDF)
- MGC User's Manual (MGC_USR.PDF)
- MGC Reference Manual (MGC_REF.PDF)

The **MGC4C Programmer's Manual** is the language specific (C/C++) manual and provides information needed to install and compile example programs in a C/C++ environment.

The **MarshallSoft GPS Component User's Manual** (MGC_USR) discusses GPS (Global Positioning System) fundamentals as well as language independent programming issues such as application notes, purchasing and licensing.

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

Online documentation can be accessed on the **MarshallSoft GPS Component for C/C++** product page at:

http://www.marshallsoft.com/mgc4c.htm

1.3 Example Program

The following example demonstrates the use of some of the **MarshallSoft GPS Component** library toolkit functions. It reads, decodes and prints out GPGGA navigation sentence data.

```
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "mgc.h"

static char DataBuffer[256];
static int Port;

int ShowMGCError(int Code)
{printf("MGC error %d\n",Code);
 return Code;
}

void main(int argc, char *argv[])
{int Code;
 double Latitude;
 double Longitude;
 /* process args */
 if(argc!=2)
   {printf("Usage: EXAMPLE <port>\n");
    return;
   }
 Code = mgcAttach(MGC_KEY_CODE);
 if(Code<0)
   {printf("Error %d: Cannot attach MGC32.DLL, check key code\n", Code);
    return;
   }
 Port = atoi(argv[1]) - 1;
 Code = mgcOpen(Port);
 if(Code<0)
   {ShowMGCError(Code);
    exit(1);
   }
 printf("EXAMPLE Program\n");
 // specify GPGGA sentences
 Code = mgcSetInteger(MGC_SET_SENTENCE_TYPE, MGC_SENTENCE_GPGGA);
 if(Code<0)
   {ShowMGCError(Code);
    exit(1);
   }
 while(1)
   {// lock data buffer
    mgcLockData(1);
    // get data quality
    Code = mgcGetData(GPGGA_QUALITY,(LPSTR)DataBuffer);
    if(Code<0) ShowMGCError(Code);
    if(Code==1)
      {// reference count = 1 (1st time seen this data set)
       printf("<%s> ", DataBuffer);
       // get latitude
       Latitude = (double)mgcGetLatitude() / 100000.0;
       printf("Lat=%0.4f ", Latitude);
       // get longitude
       Longitude = (double)mgcGetLongitude() / 100000.0;
       printf("Lon=%0.4f ", Longitude);
       printf("\n");
      }
    // unlock data buffer
    mgcLockData(0);
    Sleep(500);
   }
 mgcClose();
} /* end main */
```

1.4 Installation

(1) Before installation of MGC4C, a Windows C/C++ compiler should already be installed on your system and tested. In particular, include command line tools when installing your compiler if you want to compile using command line makefiles. If you need help with makefiles, see MAKEFILE.TXT.

(2) Unzip MGC4C22.ZIP (evaluation version) or MGCxxxxx.ZIP (registered version where xxxxx is your Customer ID) using any Windows unzip program.

(3) Run the installation program, SETUP.EXE that will install all MGC4C files, including copying MGC32.DLL and MGC64.DLL to the Windows directory. Note that no DLL registration is required.

All recent WIN32 C/C++ compilers support the "declspec" keyword. Microsoft Visual C++ (version_4.0 and up), Borland (version_5.0 and up), Watcom (version_11.0 and up), and LCC-Win32 compilers support the "declspec" keyword.

1.5 Uninstalling

Uninstalling MGC4C is very easy. First, delete the MGC project directory created when installing MGC4C. Second, delete MGC32.DLL and MGC64.DLL from your Windows directory, typically C:\WINDOWS for Windows 95/98/Me/XP/2003/Vista/7 and C:\WINNT for Windows NT/2000. That's it!

1.6 Pricing

A developer license for the MarshallSoft GPS Component library can be purchased for $115 (or $195 with Ansi C source code to the MGC library DLL). Purchasing details can be found in Section 1.4, "How to Purchase", of the MGC User's Manual (MGC_USR) http://www.marshallsoft.com/mgc_usr.pdf.

Also see INVOICE.TXT or http://www.marshallsoft.com/order.htm

1.7 Updates

When a developer license is purchased for MGC4C, the developer will be sent new registered DLLs (MGC32.DLL and MGC32.DLL) plus a license file (MGCxxxx.LIC) that can be used to update the registered DLLs for a period of one year from purchase. Updates can be downloaded from

http://www.marshallsoft.com/update.htm

After one year, your license must be updated to be able to download updates. A developer license can be updated for:

- $30 if the update is ordered within one year of the original purchase (or previous update).
- $55 if the update is ordered between one and three years of the original purchase.
- $75 if the update is ordered after three years of the original purchase.

If source code was previously purchased, updates to the source code can be purchased for $40 along with the DLL update. A license updates includes an additional year of technical support. Note that the registered DLL's never expire.

2 Library Overview

2.1 Dynamic Link Libraries

The **MarshallSoft GPS Component Library (CSC)** is implemented as a Win32/Win64 dynamic link library (DLL). A DLL is characterized by the fact that it need not be loaded until required by an application program and that only one copy of the DLL is necessary regardless of the number of application programs that use it. Contrast this to the traditional static library that is bound to each and every application that uses it at link time.

An important advantage that DLL's have over other "popular" library formats such as VBX or OCX is that DLL's are callable by all Windows applications. Since DLL's are the building blocks of the Windows Operations System, they will not be replaced by a "newer technology".

2.2 Keycode

The MGC DLLs (MGC32.DLL and MGC64.DLL) have a KeyCode encoded within them. Your keycode is a 9 or 10 digit decimal number (unless it is 0), and will be found in the file KEYCODE.H. The keycode for the evaluation version is 0. You will receive a new key code when registering. During SETUP, the keycode is copied to the \APPS sub-directory.

If the error message (value -74) is received when calling **mgcAttach**, it means that the keycode in your application does not match the keycode in the DLL. After purchasing, it is best to remove the evaluation version of MGC32.DLL/MGC64.DLL from the Windows search path.

2.3 Using the Library

The **MarshallSoft GPS Component Library for C/C++ ++** has been tested on multiple computers running Windows 95/98/Me/2003/XP/Vista/7 and Windows NT/2000.

The MGC4C library has also been tested with several C/C++ compilers, including Microsoft Visual C++ (all versions including Visual Studio .NET and Visual Studio C# .NET), Borland C/C++, Borland C++ Builder, Turbo C/C++ for Windows, and Watcom C/C++.

The SETUP installation program will copy the Lib's and DLL to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the MGC4C files are copied to the directory specified (default \MGC4C). Three sub-directories are created, as follows:

```
DOCS – All documentation files
APPS – All example code
DLLS – All DLL's
```

2.4 GUI and Console Mode

MGC4C functions can be called from any Win32 application program (Win32 GUI mode or Win32 console mode). MGC4C functions cannot be called from Win16 programs.

A "GUI Mode" program is the common Windows program with the Graphical User Interface (GUI).

A "console mode" program is a Windows 95/98/NT/2000/Me/XP/2003/VISTA/7 WIN32 command line program running in a command window. Although console mode programs look like DOS programs, they are WIN32 programs that have access to the Win32 API and the entire Windows address space. Programming using console mode programs reduces the complexity of using GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code

2.5 Using Threads

MGC4C is thread safe, and may be called from threaded applications.

2.6 Win32 STDCALL and DECLSPEC

MGC is compiled using the _stdcall and _declspec keywords. This means that MGC4C uses the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. There are no leading underscores or trailing "@size" strings, added to function names.

Microsoft Visual C++ users can look at the MGC function names using the dumpbin.exe executable:

```
dumpbin /exports mgc32.dll
```

2.7 Calling MGC from C++

Like Windows itself, MGC functions are coded in ANSI C, but they can be called directly from both ANSI C programs and from C++ and C# programs.

MGC functions can also be called using the C++ class wrapper **fmgc**. See **fmgc.cpp** and **fmgc.h**. Refer to HELLO.CPP for an example.

2.8 Adding MGC to an Existing Program

In order to add **MGC** to an existing program, (1) add

```
#include "mgc.h"
```

to your source code, (2) link with MGC32.LIB (for MSVC), MGC32BCB.LIB (Borland C/C++ and C++ Builder), MGC32.LIB (Watcom), or MGC32LCC (Win32/LCC), and recompile from source.

For Win64 applications, link with MGC64.LIB rather than MGC32.LIB. Refer also to Section 4.0 below.

2.9 Error Display

The error message text associated with **MGC** error codes can be displayed by calling **mgcErrorText**.

```
void DisplayError(int ErrCode, char *MsgPtr)
{int Len;
 char ErrBuff[129];
 printf("ERROR %d: ", ErrCode);
 if(MsgPtr) printf("%s: ", MsgPtr);
 Len = mgcErrorText(ErrCode, (char *)ErrBuff, 128);
 if(Len>0) printf("%s\n", ErrBuff);
 else printf("\n");
}
```

2.10 Explicitly Loading MGC32.DLL/MGC64.DLL

When an application program runs that makes calls to MGC32.DLL (or MGC64.DLL), the Windows operating system will locate MGC32.DLL (or MGC64.DLL), by searching the directories as specified by the Windows search path. If the MGC32.DLL or MGC64.DLL is placed in the \WINDOWS directory (or \WINNT for Windows NT/2000), the DLL will always be found by Windows.

However, MGC32.DLL or MGC64.DLL can also be loaded from a specified directory by using the GetProcAddress Win32 API function. For an example, refer to the LoadLib.c program.

2.11 Targeting a 64-Bit CPU

If a compiler generates 32-bit application code and is running on a 64-bit version of Windows, then compiling and linking is the same as it is on a 32-bit Windows system. The 32-bit application code generated will be executed by the Windows **WOW64** (Windows on Windows 64-bit) component.

If a compiler generates 64-bit application code and is running on a 64-bit version of Windows, then the compiler must be reconfigured to generate 32-bit application code if the application will call 32-bit DLL's such as MGC32.DLL. The 32-bit application code generated will be executed by the Windows **WOW64** (Windows on Windows 64-bit) component.

2.11.1 Visual Studio C/C++: Versions 2005 through 2010

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click the "Configuration Manager" button in upper right corner.
Click the drop-down button below "Platform".
Click <New...>, then choose "x86" (Win32).

2.11.2 Visual Studio Visual Basic: Versions 2005 through 2010

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click "Build", then "Configuration Manager".
Click the drop-down button below "Active Solution Platform".
Click <New...>, then change "Any CPU" to "x86".

2.12 64-bit MGC

64-bit DLL's (such as MGC64.DLL) may only be used by 64-bit application programs running on 64-bit Windows computers. This means that 64-bit application programs must be linked with MGC64.LIB instead of MGC64.LIB.

However, if a compiler generates 32-bit code, the application must be linked with MGC32.DLL even though it may be running on a 64-bit machine.

There are several MGC4C 64-bit example programs. 64-bit Visual Studio 2008 and 2010 project files include the following in the /APPS folder:

```
vc_vers(VS2008)x64.vcproj
vc_gga(VS2008)x64.vcproj
vc_latlon(VS2008)x64.vcproj

vc_vers(VS2010)x64.vcxproj
vc_gga(VS2010)x64.vcxproj
vc_latlon(VS2010)x64.vcxproj
```

3 Compiler Issues

MarshallSoft GPS Component programs can be compiled using an IDE or command line compiler tools. The following sections provide general compiler information. In order to compile from the command line, command line compiler tools must be set up properly.

3.1 Compiling Using an IDE

All current windows compilers have an "Integrated Development Environment" (IDE) for building application programs in the Windows environment. Since there is no standard format for IDE project files, file names must be entered into the IDE from the keyboard.

Note that not only do IDE's vary between the different compiler manufacturers, but they also vary from version to version for the same compiler.

3.1.1 Compiling Example Programs with an IDE

Most of the example programs can be compiled from your compiler's IDE. For Visual C/C++, "project makefiles" are used since they can be used by all versions of Microsoft Visual C++ (v4.0, v5.0, and v6.0). When opening the workspace, select "makefiles(.mak)" for the file type.

Alternatively, for Visual C++ v6.0, select "projects (.dsp)" for the file type.

3.1.2 Compiling New Projects with an IDE

Creating a project makefile for the examples that have only command line makefiles is fairly straight forward. All of the IDE's use the concept of a file hierarchy. For example, the MGCVER example program file hierarchy in the IDE (for 32-bit) should look like:

```
MGCVER.EXE
   +++ MGCVER.C
   +++ MGC.LIB
```

Replace MGC.LIB above with MGCBCB.LIB if using Borland C++ Builder and with MGCLCC.LIB if using Lcc-Win32.

The order of the files is not significant. Also refer to the sections on individual IDE's that follow this section.

3.2 Command Line Tool Setup

Many software developers overlook the power of using command line compilers. There are a number of very significant advantages to using the command line version of your C/C++ compiler. Among these are:

- **Easy of Use**: Once set up, typing a single key can compile one or a thousand programs.
- **Power**: Using the command line allows the use of batch files, automating complicated builds.
- **Compatibility**: Command line makefiles (unlike IDE project files) are normally compatible from one version of your compiler to the next.

If you want to compile from the command line, your command line compiler tools must be set up properly. Note that you have an option of installing the command line tools (or not) when your compiler is first installed. Refer to your compiler manufacturer's manual for details.

If necessary, you can increase the size of your environment table space (to 1024 for example) by adding

```
SHELL=C:\COMMAND.COM /e:1024 /p
```

to CONFIG.SYS in C:\ and then rebooting. This works for all versions of Windows, including Windows NT, XP and Windows 7.

For all compilers, your path should point to the compiler's BIN directory. For example, to add "C:\BC50\BIN" to your existing path, use

```
PATH C:\BC50\BIN;%PATH%
```

3.2.1 Microsoft

Set LIB and INCLUDE environment variables. For example,

```
SET INCLUDE=C:\MSVC\INCLUDE
SET LIB=C:\MSVC\LIB
```

3.2.2 Borland

Check that TURBOC.CFG, BCC32.CFG, TLINK.CFG, and TLINK32.CFG all have the correct information in them, as they should have when your compiler was installed. For example, assuming your BC compiler is installed at C:\BC5, the INCLUDE `(-I)` and LIB `(-L)` paths are specified by:

```
-IC:\BC5\INCLUDE
-LC:\BC5\LIB
```

BRCC (the Borland Resource Compiler) doesn't use the *.CFG files. Set the INCLUDE environment variable or BRCC will not be able to find the INCLUDE files (such as WINDOWS.H). For example,

```
SET INCLUDE=C:\BC5\INCLUDE
```

Clear the LIB environment variable (so it is not present when SET is typed at the command line) with

```
SET LIB=
```

3.2.3 Watcom

Set the WATCOM environment variables to point to your compilers include (H) and BIN directories. For example,

```
SET INCLUDE=C:\WC11\H;C:\WC11\H\NT
SET WATCOM=C:\WC11
SET EDPATH=C:\WC11\EDDAT
SET WWINHELP=E:\BINW
```

3.2.4 LCC-Win32

The LCC environment variables are set like the others. For example,

```
SET INCLUDE=C:\LCC\INCLUDE
SET LIB=C:\LCC\LIB
```

After making the above changes for your compiler, type PATH at the command line prompt to verify the search path, and type SET at the command line prompt to verify the INCLUDE and LIB environment variables.

3.2.5 MinGW GCC

In order to use the MinGW compiler, add the path to the MinGW bin directory. For example,

```
PATH =  C:\COMPILER\MinGW\bin;%PATH%
```

3.3 Command Line Batch Files

If your compiler installation includes command line tools, then all of the example programs can be compiled directly from the command line. These same compiler commands can also be placed in a batch file.

See MGCVER$.BAT for an example of a console mode command line batch file and LATLON$.BAT for an example of a GUI mode command line batch file. Similarly, command line batch files can be created for all of the example programs.

3.4 Command Line Makefiles

Command line makefiles originated on UNIX systems. They are the standard way that C/C++ programs are constructed in command line environments. The advantage of makefiles (as compared to an integrated development environment) is that all compiler switches are coded within the makefile and the makefile can be run with a single keystroke.

Command line makefiles are provided for Microsoft, Borland, Watcom, and LCC-Win32 command line compilers. They can be found in the APPS sub-directory:

- Microsoft command line makefiles end with the extension _M_
- Borland 5.0 command line makefiles end with the extension _B_
- Borland 5.5 command line makefiles end with the extension _I_
- Watcom 11 command line makefiles end with the extension _W_

4.0 Supported Compilers

The **MarshallSoft GPS Component for C/C++** has been tested with and supports the following C/C++ compilers:

- Microsoft Visual C++ (all versions)
- Microsoft Visual Studio .NET
- Borland C/C++
- Borland C++ Builder
- Borland Turbo C++
- Watcom C/C++
- Lcc-Win32 C.
- MinGW

Other Windows C/C++ compilers may work as well. Refer also to Section 5.2 "Compiling Example Programs".

4.1 Microsoft Visual C++

Microsoft Visual C++ programs can be compiled from either the command line or from within the Microsoft development environment. All MS VC programs are Win32 only.

4.1.1 Microsoft Command Line Makefiles

Programs can be compiled using command line makefiles. All Microsoft (Win32) command line makefiles end with '._m_". To compile using a makefile, use the Microsoft NMAKE utility. For example,

```
NMAKE -f LATLON._M_
```

MGC can be used with Microsoft Foundation Class (MFC) programs. Use the MFC_PGM.MAK makefile to compile the MFC_PGM example program.

```
NMAKE -f MFC_PGM.MAK
```

4.1.2 **Microsoft Developer Studio** (VC v4.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK" , such as MGCVER.MAK

To create a new project in MSVC v4.0, choose "File", then "New", then "Project Workspace". Select "Application" or "Console Application" for "Type:" and the project name for "Name:". Choose Win32 for platform. Then select "Create". Select "Insert", then "Files into Project". Add all filenames including any resource file (.RC) and MGC32.LIB. Lastly, select "Build", then "Rebuild All".

Be sure to specify /YX rather than /Yu in the project settings [Build, Settings…, C/C++].

4.1.3 **Microsoft Developer Studio** (VC v5.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK" , such as MGCVER.MAK.

To create a new project in MSVC v5.0, choose "File", then "New", then "Win32 Application" or "Win32 Console Application" and the project name. Check "Create new workspace". Select "Project", then "Add to Project". Add all filenames including any resource file (.RC) and MGC32.LIB. Lastly, select "Rebuild All".

If the compiler complains that it cannot find "_main", "Console Application" was chosen but the program being compiled is a GUI application. If the compiler complains that it cannot find "WinMain", "Application" was chosen but the program being compiled is a Console Mode application. Be sure to specify /YX rather than /Yu in the project settings [Build, Settings…, C/C++].

4.1.4 **Microsoft Developer Studio** (VC v6.0)

To open an existing project, follow the same directions as for MSVC v5.0, except that a DSP project file may be used instead of the MAK project makefile.

To create a new project in MSVC v6.0, follow the same directions as for MSVC v5.0 above.

4.2 Microsoft Visual Studio (C++ .NET)

All Visual Studio VC .Net projects end with extension "`.vcproj`". For example,

```
vc_vers.vcproj
```

In order to open an existing Visual C++ .Net project, choose "File", "Open", and then "Project" from the Microsoft Visual Studio menu. Specify the directory containing the Visual C++ .Net project files (for example, C:\MGC4C\APPS).

Visual Studio can also open projects ending with ".DSP", creating a Visual Studio ".VCPROJ" project file.

In order to call MGC functions from C++ .Net programs, do the following to your existing C++ .Net project:

(1) Add

```
#include "mgc.h"
#include "keycode.h"
```

after your existing #include statements.

(2) Open the project properties window under "Project" on the main menu. If your project is named "MyProject", then select "MyProject properties".

(3) Select "Configuration Properties", "Linker", "Input", "Additional Dependencies", and then type in "MGC32.LIB" into the edit box. Note that MGC32.LIB must be in your project directory along with all of your source files.

(4) Rebuild. NOTE: If using pre-compiled headers, the include statement

```
#include "stdafx.h"
```

must be the first include statement in your program.

4.2.1 Microsoft Visual Studio 2003, 2005, 2008 and 2010 (VC++ 7.0, VC++ 8.0, VC++ 9.0 and VC++ 10.0)

Open the VC project file (files ending in ".vcproj" or ".vcxproj" [for Visual Studio 2010]), build and run, as in previous versions of Visual Studio.

Also see section 4.3 below.

4.2.2 Microsoft C++ Express Edition (VC++ 10.0)

The "Express Edition" of Microsoft C++ 10.0 is available as a free download at
http://www.microsoft.com/express/download/

Open the VC project file (files ending in ".vcxproj", ".vcproj" or ".dsp"), build and run, as in previous versions of Visual Studio. Also see section 4.3 below.

4.3 Microsoft Visual Studio C# .NET

MarshallSoft GPS Component functions can be called from Microsoft Visual C# (C-sharp) .NET in the same manner as Win32 API functions.

All Visual C# .NET projects end with extension ".csproj". For example,

```
cs_vers.csproj
```

In order to open an existing C# .NET project, choose "File", "Open", and then "Project" from the Microsoft C# Development Environment. Specify the directory containing the C# .NET project files (for example, C:\MGC4C\APPS).

In order to call **MGC** functions from an existing C# .NET program, do the following to the C# .NET source code:

(1) Add the contents of file `mgc_funs.cs` to the source code after

```
public class mgc : System.Windows.Forms.Form
```

(2) Add the constants from `mgc_cons.cs` to the program as they are needed.
Look at the `cs_vers` program in the APPS sub-directory for an example.
(3) Set "unsafe" compilation. Verify that AllowUnsafeBlocks is set to true in the project file (.vcproj).

4.4 Borland C++

Borland C++ version 5.0 programs can be compiled from either the command line (using makefiles ending with "._b_") or from within the Borland development environment using Borland v5.0 or above.

Borland C++ version 5.5 (available from http://www.codegear.com/downloads/free/cppbuilder) is a free Win32 console mode compiler (no IDE). Makefiles for BC v5.5 end with "._i_", and (like Borland C++ Builder) use ILINK32 rather than TLINK32. Be careful with linker response files (*.RSP) -- they must NOT end with a carriage return / line feed!

Borland programs always link with MGC32BCB.LIB.

Also refer to BORLAND.TXT.

4.4.1 Borland Command Line Makefiles

Programs can be compiled using command line makefiles. All Borland (Win32) command line makefiles end with "`._b_`" (or "`._i_`" for Borland 5.5). To compile using a makefile, use the Borland MAKE utility. For example,

```
MAKE -f LATLON._B_
MAKE -f LATLON._I_
```

4.4.2 Borland IDE

Win32 Borland users re-create create MGCBCB.LIB from MGC32.DLL by running the "IMPLIB" program in the Borland compiler \BIN directory.

```
implib mgcbcb.lib mgc32.dll
```

To create a new project, first turn off LINKER case sensitivities: Choose "Options", "Projects", "Linker", "General". Turn off the "case sensitive link" and "case sensitive exports and imports" boxes.

Next, choose "Project", then "New Project". Use the INS (Insert) key to pop up a dialog box into which the project file names are entered.

Select "GUI" or "Console" for the "Target Model:" Only "Runtime" and "Dynamic" should be checked for "Standard Libraries:"

NOTE1: If, after linking in the IDE, you get unresolved external references to the library functions in which each function name is all upper case, then you have NOT turned off case sensitivity as described above.

NOTE2: If you get errors compiling the windows header file "WINDOWS.H", turn on "Borland Extensions" in "Options", "Project", "Compiler", "Source".

4.5 Borland Turbo C/C++ for Windows

Borland Turbo C/C++ for Windows does not have command line tools, so all programs must be compiled from the Turbo C/C++ integrated environment.

Follow the same directions as above (Borland IDE), except that the "Target Model:" can be any listed.

4.6 Borland C++ Builder

Borland C++ Builder does not have command line tools, so all programs must be compiled from the Borland C++ Builder integrated environment. Compile the BCB example program BCB_PRJ with BCB_PRJ.MAK if running BCB version 1 through 3, and compile with BCB_PRJ.BPR if running BCB version 4 or above.

To load the BCB_PRJ example project, Choose "File" / "Open Project" on the menu bar. Load BCB_PRJ.MAK (or BCB_PRJ.BPR). Then, choose "Build All" from "Project" to create the executable.

Note that MGCBCB.LIB is the LIB file used with Borland C++ Builder. MGCBCB .LIB can be created from MGC.DLL by using the Borland IMPLIB program:

```
IMPLIB MGCBCB.LIB MGC.DLL
```

4.7 Watcom C/C++

Watcom C/C++ programs can be compiled from either the command line or from within the Watcom development environment.

Watcom v10.5and v10.6 do not recognize the "declspec" keyword, and there require "legacy DLL's", which can be downloaded from www.marshallsoft.com/mgc4c.htm

4.7.1 Watcom Command Line Makefiles

Programs can be compiled using command line makefiles. All Watcom command line makefiles end with "._w_" for (Win32) makefiles. Use WCC386.EXE to compile C programs and WPP386.EXE to compile C++ programs.

To compile using a makefile, use the Watcom WMAKE utility. For example,

```
WMAKE -f LATLON._W_
```

4.7.2 Watcom IDE

To create a new project, choose "File", then "New Project". Enter the project name and then choose Win32 as the target. Use the INS (Insert) key to pop up a dialog box into which the project file names are entered.

Select "Options" from the main window, then "C Compiler Switches", then "10". Memory Models and Processor Switches". Check "80386 Stack based calling [-3s]", then check "32-bit Flat model [-mf]".

4.8 LCC-Win32 C/C++

Lcc-Win32 C/C++ programs can be compiled from either the command line or from within the development environment.

Lcc-Win32 is a freeware C compiler developed and distributed by Jacob Navia at

```
http://www.cs.virginia.edu/~lcc-win32/
```

To use our DLL's with Lcc-Win32, you must link with MGCLCC.LIB. This file can also be re-created using the Lcc-Win32 utility BUILDLIB.

```
buildlib mgc.lcc mgclcc.lib
```

Then, compile and link as normal. For example, to compile the MGCVER example program,

```
lcc -DWIN32 mgcver.c
lcclnk mgcver.obj mgc.lib -subsystem:console
```

To compile the GUI mode example LATLON,

```
lcc -DWIN32 latlon.c
lrc latlon.rc
lcclnk latlon.obj mgclcc.lib latlon.res -subsystem:windows
```

See MGCVER$.BAT for an example of a console mode command line batch file and LATLON$.BAT for an example of a GUI mode command line batch file.

Command files are used for Lcc-Win32 rather than makefiles since the makefile that comes with LCC-Win32 does not work well (unlike the actual compiler).

4.9 MinGW C/C++

MinGW (Minimalist GNU for Windows) is part of the GNU Compiler Collection (GCC), and GNU Binutils, for use in the development of native Microsoft Windows applications. See http://www.mingw.org

Console mode programs are compiled from the command line; for example

```
gcc -Wall Example.c csc32.lib -o Exampler.exe
```

The current version of MinGW does not come with a resource compiler (for RC files), so GUI Window applications that use resource files cannot be compiled with MinGW.

The file, gcc.zip, contains the MinGW (gcc) project command files.

5 Compiling Programs

MarshallSoft GPS Component Library (MGC) applications can be compiled using an IDE or command line compiler tools. The following sections provide general compiler information.

5.1 Compiling MGC

The MGC32.DLL has been compiled using Microsoft Visual C/C++ and is callable from applications written using Microsoft, Borland, or Watcom compilers. If you recompile MGC.C using Borland or Watcom compilers, then the resulting MGC32.DLL can only be used by applications compiled with the same compiler, unless the "_stdcall" and "_declspec" keywords are specified.

For Microsoft C, type:

```
NMAKE -f MGC._M_
```

For Borland C, type:

```
MAKE -f MGC._B_
```

For Watcom C, type:

```
WMAKE -f MGC._W_
```

5.2 Compiling Example Programs

There are makefiles provided for each of the example programs. For example, to compile LATLON:

For Microsoft C, type:

```
NMAKE -f LATLON._M_
```

For Borland C, type:

```
MAKE -f LATLON._B_
```

For Watcom C, type:

```
WMAKE -f LATLON._W_
```

There is also a Microsoft Developer Studio generated makefile for LATLON. See LATLON.MAK.

5.3 Static Linking

The registered user can also statically link with **mgc.obj** and **wsc32mgc.obj**, rather than making calls to the DLL's. To create an application that links MGC.OBJ statically:

(1) All application code that includes MGC.H must define STATIC_LIBRARY before including MGC.H
(2) Your application must **not** link with MGC.LIB.

For an example of linking with **mgc.obj** and **wsc32mgc.obj**, see LATLONS._M_ .

If using Microsoft Developer Studio, make these changes:

(1) To your project file: Do NOT add MGC.LIB to your project file.
(2) To the settings: (See "Build Settings" or "Project/Settings")
(2a) C/C++ Tab: Add STATIC_LIBRARY to "preprocessor definitions:".
(2b) Link Tab: Add **mgc.obj** and **wsc32mgc.obj** to "object/library modules:"

Alternatively, if source code has been purchased, MGC.C and WSC32GPS.C can be edited so that they can be compiled and linked in like any other program file. In order to do this, remove all code (from MGC.C) from

```
#ifndef STATIC_LIBRARY
```

to the matching

```
#endif
```

6 Example Programs

Many of the example programs are written in Win32 console mode. This was done in order to provide the clearest possible code, without the complication and complexity of GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code. Example programs are located in the \APPS sub-directory created by SETUP.

Project files are classified as follows:

```
//$  *.vcproj  Visual Studio makefile.
//$  *.dsp     VC_6.0 / Visual Studio makefile.
//$  *.mak     Microsoft C/C++ (or Borland) makefile.
//$  *._m_     Microsoft C/C++ command line makefile.
//$  *._b_     Borland C/C++ 5 command line makefile.
//$  *._i_     Borland C/C++ 5.5 command line makefile.
//$  *._w_     Watcom C/C++ 11 command line makefile.
```

6.1 BCB_PGM

The BCB_PGM.CPP program is a C++ Builder program that continuously displays latitude and longitude.

Load project file BCB_PRJ.MAK for BCB version_1.0 and BCB_PRJ.BPR for later versions of BCB.

6.2 CALLBACK

The CALLBACK example console mode program demonstrates the use of the **mgcCallback** function.

6.3 COMPUTE

COMPUTE is a console mode program that computes great circle distances and bearings from a pair of latitude/longitude values.

6.4 CONVERT

CONVERT is a console mode program that coverts latitude (or longitude) between (deg, min, min/1000), (deg, min, sec), and single integer encoding.

6.5 CS_VERS

The CS_VERS example program is the Visual C# .NET equivalent of the MGCVER program.

6.6 CS_LATLON

The CS_LATLON example program is the Visual C# .NET equivalent of the LATLON program.

6.7 Example

The "Example" program is a console mode program that displays latitude and longitude in decimal degrees from GGA sentences. Compare to the LatLon.c example program.

6.8 GGA_RMC

GGA_RMC is a console mode program that displays both NMEA GPGGA and GPRMC sentences.

6.9 GPBOD

GPBOD is a console mode program that displays all NMEA GPBOD (bearing, origin to destination) sentences.

6.10 GPGGA

GPGGA is a console mode program that displays all NMEA GPGGA (GPS fix data) sentences.

6.11 GPGLL

GPGLL is a console mode program that displays all NMEA GPGLL (geographic position, latitude / longitude) sentences.

6.12 GPGSA

GPGSA is a console mode program that displays all NMEA GPGSA (GPS DOP and active satellites) sentences.

6.13 GPGSV

GPGSV is a console mode program that displays all NMEA GPGSV (GPS satellites in view) sentences.

6.14 GPRMC

GPRMC is a console mode program that displays all NMEA GPRMC (recommended minimum specific GPS/transit data) sentences.

6.15 GPVTG

GPVTG is a console mode program that displays all NMEA GPVTG (track made good and ground speed) sentences.

6.16 GPWPL

GPWPL is a console mode program that displays all NMEA GPWPL (waypoint location) sentences.

6.17 HELLO

The HELLO example program is similar to MGCVER except that it is written in C++ and uses the MGC class **fMGC**. See fmgc.cpp and fmgc.h.

6.18 LATLON

LATLON is a GUI mode example program that continuously displays latitude and longitude.

6.19 LoadLib

The LoadLib example program demonstrates how to load MGC32.DLL (or MGC64.DLL) from a specified directory.

6.20 MFC_PGM

MFC_PGM is a Microsoft Foundations Class (MFC) version of the LATLON example program. Load MFC_PGM.MAK

6.21 MGCVER

The first example program is the GUI program MGCVER (MGC Version) which displays the MGC library version number and registration string.

There are command line makefiles for Microsoft (MGCVER32._M_), Borland (MGCVER32._B_), and Watcom (MGCVER32._W_), as well as a Microsoft Developer Studio makefile (MGCVER32.MAK), and an Lcc-Win32 command file (LATLON32.BAT).

After compiling, from the command line, type:

```
MGCVER
```

6.22 RAW

RAW is a console mode program that displays all incoming NMEA sentences.

6.23 Trans

The Trans example program translate between the several forms of latitiude and longitude.

6.24 VC_GGA

The VC_GGA example program is the Visual C++ .NET equivalent of the GPGGA program.

6.25 VC_LATLON

The VC_LATLON example program is the Visual C++ .NET equivalent of the LATLON program.

6.26 VC_VERS

The VC_VERS example program is the Visual C++ .NET equivalent of the MGCVER program.

7 Revision History

Version 1.0: January 2, 2002.

- The initial release of MGC4C.

Version 1.1: March 14, 2002.
- Any keycode matches evaluation DLL.
- Can set baud serial port parameters.
- Added mgcRelease().

Version 1.2: February 4, 2003

- Added mgcErrorText().
- Added COMPUTE2 example program.

Version 1.3: August 9, 2004

- Increased maximum number of fields from 19 to 20.
- Receive buffer size increased from 256 to 1024.
- Overflow wraps to empty.
- Added MGC_SET_SLEEP to mgcSetInteger (sets work buffer sleep time).
- Added mgcStatus() function to return GPS data status.

Version 1.4: March 3, 2005

- Added $GPVTG sentence type.
- Interval diagnostics added.
- Added C# support.

Version 2.0: June 22, 2007

The "Coded Integer" angles returned by MGC functions are now returned in units of hundred-thousandths of a degree rather than thousandths of a minute.

- mgcBearing returns the bearing in units of thousandths of a minute (1000 units = 1 min).
- Two sentence types can be specified rather than just one. See GGA_RMC.C
- mgcGetSentenceType added that returns the current sentence type (defined in mgc.h)
- mgcNewLatitude added that returns a new latitude at a specified distance.
- mgcNewLongitude added that returns a new longitude at a specified distance.
- mgcCallback added that implements a callback function. See the CALLBACK.C example program.

Version 2.1: March 2, 2009

- mgcSetInteger (MGC_SET_BAUDRATE, baud-rate) changes baud rate immediately.
- Added GPBOD sentence type.
- Added GPWPL sentence type.
- Number of ports increased to 256 (COM1 through COM256).
- Added documentation in Adobe PDF format.

Version 2.2: June 8, 2011

- Added support for Win-64.
- Added GPGSA sentence type.
- Fixed problem with leaking thread handle.

[END]