Adacalc aeronautical math dynamic-link library (Adacalc.dll)
Version 3.21

Introduction

This library (Adacalc.dll) provides accurate aeronautical mathematical functions for developers. Both compile-time static linking (C,C++ via the enclosed .lib file and Delphi) and run-time linking (C, C++, Delphi, VB) are achievable.
Note that DLL is linked statically to CRT (multithreaded /MT option) so as to avoid dependencies.
Considering there could be some problems for linking the DLL in C++ projects build with other versions of the compiler, you may need to tell the linker to ignore the dynamically linked CRT explicitly or make use of a dynamically linked version I will provide you on request.

[_stdcall] C++ calling convention has been used for compilation of all functions;

A prototype for static linking (C and C++) and run-time linking (VB) is provided for each function (see reference).

In Delphi, DLL import can be achieved as:

```
function Standard_Temperature (z: Double): Double ; external ‘Adacalc.dll’;
```

Be aware that passing a wrong variable type to the DLL function will usually result in a crash of the calling program

This document provides the reference list of exported functions including:
- Function name
- Description
- Declaration prototypes in C and VB
- Input parameters (with units and limits as appropriate)
- Output results.

Note that some functions provide results that are passed “by reference”.

Calculations are based on accurate algorithms. However, in some “limit” cases, inaccuracies may occur (although most of them will be detected by the code and managed appropriately).

All exported functions will return the special value _NADBL (equal to -1E+32) in case of input error, mathematical error or indeterminate result. It is of the responsibility of the calling program to check for this before using the calculated data.

Microsoft Flight Simulator and P3D use

If you intend to use DLL functions in a FS/P3D oriented application, I advise you make use of the AdacalcFS.dll. The reason for that is FS and P3D wrongly perform some speed conversions from calibrated airspeed (CAS) and some supersonic speed calculations are unreliable in FS/P3D.
For a technical discussion see this link.

Note: In FS and P3D, CAS if needed can be approximated from indicated airspeed (IAS) using the following equation:
CAS = [(IAS/scalar)-offset]/Cos(AoA)
Scalar and offset being the values defined in the [airspeed_indicators] section of the aircraft.cfg file and AoA being the angle of attack.

Input value validity for some function calls may also differ (indicated in red)

Real aviation applications should make use of the regular Adacalc.dll.

Usage and Distribution

This library is distributed under a PUBLIC license. This means that the user of this product shall comply in all respects to the following terms and conditions:

1. Usage and Distribution
This software is released as freeware. As so, permission is granted to distribute it unchanged on any free media or network that does not have a per-file download charge. You are not allowed to distribute, rent, lease or sublicense this program to other users. You can include it to your own package, software or web site, as far as you indicate its origin and Copyright mention.

2. Limitation of Warranty
This software is provided "as-is" without any express or implied warranty. In no event shall the author be held liable for any direct, indirect, incidental, special, exemplary or consequential damages (including but not limited to loss of data or profits) arising from the use of this software, even if advised of the possibility of such damage.

3. Copyright
This software is copyrighted by Hervé Sors, 2019-2020

Contact and updates
http://www.aero.sors.fr/adacalc.html

History of changes

Version 1.00
- First release

Version 1.10
- Corrects Wind_Prediction calculation that was buggy
- Adds Pressure_Altitude function

Version 1.20
- Corrects EAS calculations that were wrong (EAS_From_TAS and EAS_From_Mach functions) [also check passed parameters that have changed]

Version 1.30
- Implements IGRF-11 and WMM2010 magnetic models for magnetic variation calculations up to 2015

Version 1.31
- Corrects a bug in magnetic variation calculation for dates after 2010

Version 2.00
- Adds magnetic variation calculation for 2015-2020 (WMM and IGRF models); note that calculations for years <2010 are not anymore supported
- Adds "standard" magnetic variation calculation on a given year (2010 to 2020) (see RefYearMagVar function)
- Adds Changeover altitude calculation
- Adds Descent calculation
- Provides a modified DLL (AdacalcFS.dll) for use in MS Flight Simulator and P3D (see above)
- DLL recompiled with Visual C++ 2010
- Few other minor changes
- Documentation revised

Version 3.00
- Changes in Earth and navigation calculations with addition of several models and functions and correction of a few bugs (see function descriptions)
- Removes magnetic variation calculation for 2010-2014
- Small correction to magnetic variation calculation for years >2015
- Changes DISA max for functions requiring this parameter (now +70°C)
- Adds some new functions: Mach_From_Q, QNH_Altitude, Temp_Corrected_Altitude, Radial_Intersection (2 models), Signed_Dif_Heading, Cross_Along_Track, Turn_Anticipation, Unit_Conversion
- Adds a version information resource
- Few other minor changes
- Documentation revised

Version 3.10

- Refines atmosphere model constants and associated calculations according to ICAO/ISA (Doc 7488/3 Manual of the ICAO Standard Atmosphere, 1993)
  [Note that this will only marginally change function results as compared to versions 2.00 & 3.00 that used US 1976 model]
- Changes lower/upper limits of geopotential altitudes for most calculations to -5000/80000 m respectively (ICAO standard atmosphere bounds)
  [Note: corresponding values for geometrical altitudes are -4996/81019 m
- Corrects a bug that may occur in some cases when normalizing EW tracks and/or longitude differences in some functions and procedures
- Corrects a bug in rhumb line (loxodromic) calculations (wrong results for some pairs of latitude/longitude)
- Improves ellipsoid radial intersection calculation
- Documentation revised

Version 3.20

- Provides magnetic variation calculation from 2020 to 2025 based on IGRF-13 and WMM2020 models
- Simplify spherical radial intersection calculation that is now more stable
- Corrects ellipsoid radial intersection calculation that failed in many cases and was buggy, using a more stable algorithm
- Minor changes in unit conversion function
- Documentation revised

Version 3.21

- Restores magnetic variation calculation from 2015 to 2020 in addition to more recent IGRF-13 and WMM2020 models
- Corrects versioning information
- Documentation revised

Atmosphere and altitude calculations

Geopotential_Altitude

Description: calculates geopotential altitude from true geometric altitude

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall Geopotential_Altitude(double z)
```
VB
```
Declare Function Geopotential_Altitude Lib "Adacalc.dll" (ByVal z as Double) as Double
```

Input
[z]: geometric altitude (m) [Valid range: ≈ -4996 to 81019]

Output
Geopotential altitude (m)
Error return (out of range): _NADBL (-1E+32)

Pressure_Altitude

Description: calculates pressure altitude from true geometric altitude and current sea-level pressure (altimeter setting)

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall Pressure_Altitude(double z, double qnh)
```
VB
```
Declare Function Pressure_Altitude Lib "Adacalc.dll" (ByVal z as Double, ByVal qnh as Double) as Double
```

Input
[z]: geometric altitude (m) [Valid range: ≈ -4996 to 81019]
[qnh]: current sea-level pressure (inHg or hPa) [Valid range: 25.69 to 32.01 inHg or 870 to 1084 hPa]

Output
Pressure altitude (m)
Error return (out of range): _NADBL (-1E+32)

Notes
Pressure altitude is a geopotential altitude (geometric altitude is first converted to a geopotential altitude). Results “out of range” will return _NADBL
Sea-level pressure range 25.69-32.01 will be interpreted as inHg, and 870-1084 as hPa ; other values will raise an “out of range” result

Standard_Temperature

Description: calculates standard temperature of ICAO standard atmosphere at a given geopotential altitude

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Standard_Temperature(double z)
```

VB

```
Declare Function Standard_Temperature Lib "Adacalc.dll" (ByVal z as Double) as Double
```

Input

[z]: geopotential altitude (m) [Valid range: -5000 to 80000]

Output

Standard temperature (°C)

Error return (out of range): _NADBL (-1E+32)

Notes

Standard temperature ratio (θ) is [(Standard_Temerature(z)+273.15)/288.15]

Standard_Pressure_Ratio

Description: calculates standard pressure ratio ($\delta = P/P_0$) of ICAO standard atmosphere at a given geopotential altitude

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Standard_Pressure_Ratio (double z)
```

VB

```
Declare Function Standard_Pressure_Ratio Lib "Adacalc.dll" (ByVal z as Double) as Double
```

Input

[z]: geopotential altitude (m) [Valid range: -5000 to 80000]

Output

Standard pressure ratio (no units)

Error return (out of range): _NADBL (-1E+32)

Notes

Standard ambient air pressure (P, hPa) is $P = \delta \times P_0$ [P_0=1013.25 hPa]

Density_Ratio

Description: calculates true density ratio ($\sigma_t = \rho_t/\rho_0$) from geopotential altitude and temperature offset from standard

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Density_Ratio (double z, double disa)
```

VB

```
Declare Function Density_Ratio Lib "Adacalc.dll" (ByVal z as Double, ByVal disa as Double) as Double
```

Input

[z]: geopotential altitude (meters) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output

Density ratio (no units)
Error return (out of range): _NADBL (-1E+32)

Notes

Ambient air density (kg/m^3) is = σ_t x ρ_0 [ρ_0=1.225 kg/m^3]

Standard_Density_Ratio

Description: calculates standard density ratio ($\sigma = \rho/\rho_0$) of ICAO standard atmosphere at a given geopotential altitude

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Standard_Density_Ratio (double z)
```

VB

```
Declare Function Standard_Density_Ratio Lib "Adacalc.dll" (ByVal z as Double) as Double
```

Input

[z]: geopotential altitude (m) [Valid range: -5000 to 80000]

Output

Standard density ratio (no units)
Error return (out of range): _NADBL (-1E+32)

Notes

Standard ambient air density (kg/m^3) is = σ x ρ_0 [ρ_0=1.225 kg/m^3]

Alt_From_Pressure_Ratio

Description: calculates geopotential altitude of the ICAO standard atmosphere corresponding to a given pressure ratio (δ)

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Alt_From_Pressure_Ratio (double delta)
```

VB

```
Declare Function Alt_From_Pressure_Ratio Lib "Adacalc.dll" (ByVal delta as Double) as
Double
```

Input

[delta]: pressure ratio [Valid range: $8.747\ 10^{-6} \leq$ delta ≤ 1.754]

Output

Geopotential altitude (m)

Error return (out of range): _NADBL (-1E+32)

Notes

Delta values >1 will return a negative altitude

Alt_From_Density_Ratio

Description: calculates geopotential altitude of the international standard atmosphere corresponding to a given density ratio (σ)

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Alt_From_Density_Ratio (double sigma)
```

VB

```
Declare Function Alt_From_Density_Ratio Lib "Adacalc.dll" (ByVal sigma as Double) as
Double
```

Input

[sigma]: density ratio [Valid range: $1.282\ 10^{-5} \leq$ sigma ≤ 1.576]

Output

Geopotential altitude (m)

Error return (out of range): _NADBL (-1E+32)

Notes

Sigma values >1 will return a negative altitude

Sound_Speed

Description: calculates sound speed (m/s) at a given geopotential altitude and temperature offset from standard

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Sound_Speed (double z, double disa)
```

VB

```
Declare Function Sound_Speed Lib "Adacalc.dll" (ByVal z as Double, ByVal disa as Double) as Double
```

Input

[z]: geopotential altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output

Sound speed (m/s)
Error return (out of range): _NADBL (-1E+32)

Relative_Humidity

Description: calculates relative humidity (%) for a given air temperature and dew point

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Relative_Humidity (double t, double dp)
```

VB

```
Declare Function Relative_Humidity Lib "Adacalc.dll" (ByVal t as Double, ByVal dp as Double) as Double
```

Input

[t]: air temperature (°C) [Valid range: -75 to +85]
[dp]: dew point (°C) [Valid range: -75 to +85]

Output

Relative humidity (%)
Error return (out of range): _NADBL (-1E+32)

Notes

Dew point temperature must be ≤ air temperature

Density_Altitude

Description: calculates density altitude (true altitude) from geometric altitude, altimeter setting (local QNH), ambient temperature and dew point

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall Density_Altitude (double z, double qnh, double t, double dp)
```
VB
```
Declare Function Density_Altitude Lib "Adacalc.dll" (ByVal z as Double, ByVal qnh as double, ByVal t as double, ByVal dp as Double) as Double
```

Input
[z]: geometric altitude (m) [Valid range: -500 to 10000]
[qnh]: altimeter setting (local QNH, inHg or hPa) [Valid range: 25.69 to 32.01 inHg or 870 to 1084 hPa]
[t]: ambient air temperature (°C) [Valid range: -75 to +85]
[dp]: dew point (°C) [Valid range: -75 to +85]

Output
Density altitude (true geometric altitude, m)
Error return (out of range): _NADBL (-1E+32)

Notes
Altimeter setting [qnh] is the value of Kollsman window when altimeter is adjusted to read the correct altitude (local QNH). Calculation takes into account relative humidity. Altitude range will cover all airport altitudes

QNH_Altitude

Description: calculates true geometric altitude from pressure altitude and altimeter setting (local QNH)

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall QNH_Altitude (double pa, double qnh)
```
VB
```
Declare Function QNH_Altitude Lib "Adacalc.dll" (ByVal pa as Double, ByVal qnh as double) as Double
```

Input
[pa]: pressure altitude (m) [Valid range: -5000 to 80000]
[qnh]: altimeter setting (local QNH, inHg or hPa) [Valid range: 25.69 to 32.01 inHg or 870 to 1084 hPa]

Output
Geometric altitude (m)
Error return (out of range): _NADBL (-1E+32)

Notes
Altimeter setting [qnh] is the value of Kollsman window when altimeter is adjusted to current local QNH
Note pressure altitude is a geopotential altitude while result is a geometric altitude

---

Temp_Corrected_Altitude

Description: calculates true geometric altitude from calibrated (indicated) altitude (altimeter reading) at a given temperature offset from standard

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall Temp_Corrected_Altitude (double ia, double disa, double fe=0)
```
VB
```
Declare Function Temp_Corrected_Altitude Lib "Adacalc.dll" (ByVal ia as Double, ByVal qnh as double, Optional ByVal fe as Double) as Double
```

Input
[ia]: calibrated altitude (altitude indicated by altimeter when set to the altimeter setting (m) [Valid range: ≈ -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]
[fe]: (optional) field elevation of station providing the altimeter setting (m) [Valid range: -500 to 10000]-Set to 0 if missing – Must be <=[ia]

Output
True geometric altitude (m)
Error return (out of range): _NADBL (-1E+32)

Notes
Calibrated altitude (as indicated by altimeter) is a geopotential altitude
Temperature decrease from station to aircraft altitude is assumed to follow standard atmosphere equation
[disa] is the average deviation from standard temperature in the air column between the station and the aircraft

Speed calculations

Mach_From_TAS

Description: calculates mach number from true airspeed (TAS) at a given pressure (geopotential) altitude and temperature offset from standard

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall Mach_From_TAS (double tas, double z, double disa)
```
VB
```
Declare Function Mach_From_TAS Lib "Adacalc.dll" (ByVal tas as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input
[tas]: true airspeed (kts) [Valid range: 0 to 1500]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output
Mach number
Error return (out of range): _NADBL (-1E+32)

TAS_From_Mach

Description: calculates true airspeed (TAS, kts) from mach number at a given pressure altitude and temperature offset from standard

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall TAS_From_Mach (double mach, double z, double disa)
```
VB
```
Declare Function TAS_From_Mach Lib "Adacalc.dll" (ByVal mach as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input
[mach]: mach number [Valid range: 0 to 3.0]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output
True airspeed (kts)
Error return (out of range): _NADBL (-1E+32)

Mach_From_CAS

Description: calculates mach number from calibrated airspeed (CAS) at a given pressure altitude

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Mach_From_CAS (double cas, double z)
```

VB

```
Declare Function Mach_From_CAS Lib "Adacalc.dll" (ByVal tas as Double, ByVal z as Double) as Double
```

Input

[cas]: calibrated airspeed (kts) [Valid range: 0 to 1500]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]

Output

Mach number
Error return (out of range or supersonic result in FS version): _NADBL (-1E+32)

Notes

Valid for both subsonic (m <1) and supersonic speeds (m ≥ 1) in regular version, only for subsonic speeds in AdacalcFS.dll (will return NADBL if Mach>=1).

CAS_From_Mach

Description: calculates calibrated airspeed (CAS, kts) from mach number at a given pressure altitude

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall CAS_From_Mach (double mach, double z)
```

VB

```
Declare Function CAS_From_Mach Lib "Adacalc.dll" (ByVal mach as Double, ByVal z as Double) as Double
```

Input

[mach]: mach number [Valid range: 0 to 3.0, <1 for FS version]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]

Output

Calibrated airspeed (CAS, kts)
Error return (out of range): _NADBL (-1E+32)

Notes

Valid for both CAS < aSL and CAS ≥ aSL (aSL: speed of sound, sea level, standard day). For CAS ≥ aSL, precision is within 0.001 kt

TAS_From_CAS

Description: calculates true airspeed (TAS, kts) from calibrated airspeed (CAS) at a given pressure altitude and temperature offset from standard

Declaration
C/C++

```
extern "C" _declspec(dllimport) double _stdcall TAS_From_CAS (double cas, double z, double disa)
```

VB

```
Declare Function TAS_From_CAS Lib "Adacalc.dll" (ByVal cas as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input
[CAS]: calibrated airspeed (kts) [Valid range: 0 to 1500]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output
True airspeed (kts)
Error return (out of range or supersonic speed for FS version): _NADBL (-1E+32)

Notes: see Mach_From_CAS

CAS_From_TAS

Description: calculates calibrated airspeed (CAS, kts) from true airspeed (TAS) at a given pressure altitude and temperature offset from standard

Declaration
C/C++

```
extern "C" _declspec(dllimport) double _stdcall CAS_From_TAS (double tas, double z, double disa)
```

VB

```
Declare Function CAS_From_TAS Lib "Adacalc.dll" (ByVal tas as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input
[tas]: true airspeed (kts) [Valid range: 0 to 1500]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output
Calibrated airspeed (kts)
Error return (out of range or supersonic speed for FS version): _NADBL (-1E+32)

Notes: see CAS_From_Mach function

EAS_From_TAS

Description: calculates equivalent airspeed (EAS, kts) from true airspeed (TAS) at a given pressure altitude and temperature offset from standard

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall EAS_From_TAS (double tas, double z, double disa)
```
VB
```
Declare Function EAS_From_TAS Lib "Adacalc.dll" (ByVal tas as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input
[tas]: true airspeed (kts) [Valid range: 0 to 1500]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output
Equivalent air speed (kts)
Error return: _NADBL (-1E+32) [out of range input or resulting Mach ≥ 1]

Notes: EAS is only defined at subsonic speeds

EAS_From_Mach

Description: calculates equivalent airspeed (EAS, kts) from mach number at a given pressure altitude

Declaration
C/C++
```
extern "C" _declspec(dllimport) double _stdcall EAS_From_Mach (double mach, double z)
```
VB
```
Declare Function EAS_From_Mach Lib "Adacalc.dll" (ByVal mach as Double, ByVal z as Double) as Double
```

Input
[mach]: mach number [Valid range: 0 to <1.0]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]

Output
Equivalent air speed (kts)
Error return (out of range): _NADBL (-1E+32)

Notes: EAS is only defined at subsonic speeds

TAT_From_Mach

Description: calculates total air temperature (TAT, °C) from mach number at a given pressure altitude and temperature offset from standard

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall TAT_From_Mach (double mach, double z, double disa)
```

VB

```
Declare Function TAT_From_Mach Lib "Adacalc.dll" (ByVal mach as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input

[mach]: mach number [Valid range: 0 to 3.0]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output

Total air temperature (°C)
Error return (out of range): _NADBL (-1E+32)

Notes: Valid for subsonic & supersonic speeds in regular and FS versions

Q_From_Mach

Description: calculates dynamic pressure (q, Pa) from mach number at a given pressure altitude and temperature offset from standard

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Q_From_Mach (double mach, double z, double disa)
```

VB

```
Declare Function Q_From_Mach Lib "Adacalc.dll" (ByVal mach as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input

[mach]: mach number [Valid range: 0 to 3.0]
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +70]

Output

Dynamic pressure (Pa)
Error return (out of range): _NADBL (-1E+32)

Notes: Valid for subsonic & supersonic speeds in regular and FS versions

Mach_From_Q

Description: calculates mach number from dynamic pressure (q, Pa) at a given pressure altitude and temperature offset from standard

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Mach_From_Q (double q, double z, double disa)
```

VB

```
Declare Function Mach_From_Q Lib "Adacalc.dll" (ByVal q as Double, ByVal z as Double, ByVal disa as Double) as Double
```

Input

[q]: Dynamic pressure (Pa)
[z]: pressure altitude (m) [Valid range: -5000 to 80000]
[disa]: temperature offset from standard (°C) [Valid range: -90 to +40]

Output

Mach number

Error return (out of range): _NADBL (-1E+32)

Notes: Valid for subsonic & supersonic speeds in regular and FS versions

Geo_Distance_Track

Description: calculates distance and true tracks (initial and final) between 2 points on different earth representation models

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Geo_Distance_Track (double lat1, double lon1, double lat2, double lon2, int model, double &d, double &rvi, double &rvf)
```

VB

```
Declare Sub Geo_Distance_Track Lib "Adacalc.dll" (ByVal lat1 as Double, ByVal lon1 as Double, ByVal lat2 as Double, ByVal lon2 as double, ByVal model as long, ByRef d as double, ByRef rvi as double, ByRef rvf as Double)
```

Input

[lat1]: initial latitude (degrees) [Valid range: -90 to +90]
[lon1]: initial longitude (degrees) [Valid range: -180 to +180]
[lat2]: final latitude (degrees) [Valid range: -90 to +90]
[lon2]: final longitude (degrees) [Valid range: -180 to +180]
[model]: as below [valid range: 1 to 7]

1= Spherical great circle (1°=60 NM)
2= Spherical GRS 1980
3= Ellipsoid WGS84
4= Loxodromic (rhumb line, 1°=60 NM)
5= Loxodromic GRS 1980
6= Ellipsoid loxodrome
7= Equirectangular projection

Output [passed by reference in variables d, Rvi, Rvf]
d: calculated distance (NM)
Rvi: initial true track (°)
Rvf: final true track (°)
Error return (out of range or indeterminate value): _NADBL (-1E+32)

Notes

North latitudes are positive, South latitudes are negative
East longitudes are positive, West longitudes are negative
Models 1 and 4 are based on the classical spherical earth sphere representation (1'= 1 NM ; earth radius = 6366.707 km) while *models 2 and 5* (GRS1980) use an earth radius of 6371.009 km)
Model 3 is based on the WGS84 earth elliptical representation (major radius = 6378.1370 km, minor radius = 6356.7523 km, flattening = 1/298.257223563
Models 4 to 7 will return Rvf=Rvi
Model 7 should only be used for obtaining approximations on small distances (flat earth model)

Geo_Distance_Track_Inverse

Description: calculates [latitude, longitude] of a destination point from distance (NM) and true track (°) from origin on different earth representation models

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Geo_Distance_Track_Inverse (double lat1,
double lon1, double rvi, double d, int model, double &latd, double &lond)
```

VB

```
Declare Sub Geo_Distance_Track_Inverse Lib "Adacalc.dll" (ByVal lat1 as Double, ByVal lon1
as Double, ByVal rvi as Double, ByVal d as double, ByVal model as long, ByRef latd as
double, ByRef lond as double)
```

Input

[lat1]: initial latitude (degrees) [Valid range: -90 to +90]
[lon1]: initial longitude (degrees) [Valid range: -180 to +180]
[rvi]: true track from origin (degrees) [Valid range: 0 to 360]
[d]: distance from origin (NM) [Valid range: 0 to 5400]
[model]: as below [valid range: 1 to 7]

1= Spherical great circle (1°=60 NM)
2= Spherical GRS 1980
3= Ellipsoid WGS84
4= Loxodromic (rhumb line, 1°=60 NM)
5= Loxodromic GRS 1980
6= Ellipsoid loxodrome
7= Equirectangular projection

Output [passed by reference in variables latd, lond]

latd: destination latitude (degrees)
lond: destination longitude (degrees)
Error return (out of range or indeterminate value): _NADBL (-1E+32)

Notes

North latitudes are positive, South latitudes are negative
East longitudes are positive, West longitudes are negative
See Geo_Distance_Track function for model details

Radial_Intersection

Description: calculates [latitude, longitude] of intersection of 2 radials on a classical earth sphere representation or WGS84 ellipsoid

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Radial_Intersection (double lat1, double lon1,
double trk1, double lat2, double lon2, double trk2, int model, double &latd, double &lond)
```

VB

```
Declare Sub Radial_Intersection Lib "Adacalc.dll" (ByVal lat1 as Double, ByVal lon1 as Double, ByVal trk1 as Double, ByVal lat2 as Double, ByVal lon2 as Double, ByVal trk2 as Double, ByVal model as long, ByRef latd as Double, ByRef lond as Double)
```

Input

[lat1]: latitude of 1st point (degrees) [Valid range: -90 to +90]
[lon1]: longitude of 1st point (degrees) [Valid range: -180 to +180]
[trk1]: true radial from 1st point (degrees) [Valid range: 0 to 360]
[lat2]: latitude of 2nd point (degrees) [Valid range: -90 to +90]
[lon2]: longitude of 2nd point (degrees) [Valid range: -180 to +180]
[trk2]: true radial from 2nd point (degrees) [Valid range: 0 to 360]
[model]: as below [valid: 1 or 2]
    1=Spherical earth model (1°=60NM)
    2= Ellipsoid WGS84

Output [passed by reference in variables latd, lond]
latd: intersection latitude (degrees)
lond: intersection longitude (degrees)
Error return (out of range or indeterminate value): _NADBL (-1E+32)

Notes
North latitudes are positive, South latitudes are negative
East longitudes are positive, West longitudes are negative
Result may be indeterminate in some cases

Signed_Dif_Heading

Description: calculates signed heading difference

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Signed_Dif_Heading (double initialhdg, double finalhdg)
```

VB

```
Declare Function Signed_Dif_Heading Lib "Adacalc.dll" (ByVal initialhdg as Double, ByVal finalhdg as Double) as double
```

[initialhdg]: Initial heading [Valid range: 0 to 360°]
[finalhdg]: Final heading [Valid range: 0 to 360°]

Output
Normalized signed value of (finalhdg-initialhdg)
Error return (initialhdg or finalhdg out of range): _NADBL (-1E+32)

Notes
Positives values indicate a clockwise difference while negative values indicate a counter clockwise difference

Cross_Along_Track

Description: calculates cross/along distances at a given position on a defined spherical great circle (GC) path

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Cross_Along_Track (double lat1, double lon1,
double lat2, double lon2, double curlat, double curlonl, double &dorigin, double &dend,
double &dx)
```

VB

```
Declare Sub Cross_Along_Track Lib "Adacalc.dll" (ByVal lat1 as Double, ByVal lon1 as
Double, ByVal lat2 as Double, ByVal lon2 as double, ByVal curlat as double, ByVal curlon as
double, ByRef dorigin as double, ByRef dend as double, ByRef dx as double)
```

Input

[lat1]: latitude of GC start point (degrees) [Valid range: -90 to +90]
[lon1]: longitude of GC start point (degrees) [Valid range: -180 to +180]
[lat2]: latitude of GC end point (degrees) [Valid range: -90 to +90]
[lon2]: longitude of GC end point (degrees) [Valid range: -180 to +180]
[curlat]: current latitude (degrees) [Valid range: -90 to +90]
[curlon]: current longitude (degrees) [Valid range: -180 to +180]

Output [passed by reference in variables dorigin, dend and dx]
dorigin: distance from GC leg origin on projected position along GC (NM)
dend: distance from GC leg end on projected position along GC (NM)
dx: cross track distance (NM, + if right from GC, - if left)
Error return (indeterminate solution or “off course”): _NADBL (-1E+32)

Notes
North latitudes are positive, South latitudes are negative
East longitudes are positive, West longitudes are negative
Result may be indeterminate in some cases
“Off course” position will return _NADBL for all returned variables

Wind calculations

Wind_Effect

Description: calculates heading and ground speed from true airspeed, route track, wind speed and direction

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Wind_Effect (double tas, double trk, double ws,
double wd, double &hdg, double &gs)
```

VB

```
Declare Sub Wind_Effect Lib "Adacalc.dll" (ByVal tas as Double, ByVal trk as Double, ByVal
ws as Double, ByVal wd as Double, ByRef hdg as Double, ByRef gs as Double)
```

Input

[tas]: true airspeed (kts) [Valid range: 50 to 1500]
[trk]: true or magnetic route track (degrees) [Valid range: 0 to 360]
[ws]: wind speed (kts) [valid range 0 to 150]
[wd]: wind direction (degrees, true or magnetic) [Valid range: 0 to 360]

Output [passed by reference in variables hdg, gs]
hdg: aircraft heading (degrees, true or magnetic)
gs: ground speed (kts)
Error return (out of range or indeterminate value): _NADBL (-1E+32)

Notes

Route track and wind direction can be entered as BOTH true or magnetic values ; aircraft heading will be calculated accordingly
Result may be indeterminate in some cases (too strong wind for course)

Wind_Prediction

Description: calculates wind speed and direction from true airspeed, ground speed, route track and aircraft heading

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Wind_Prediction (double tas, double trk, double
gs, double hdg, double &ws, double &wd)
```

VB

```
Declare Sub Wind_Prediction Lib "Adacalc.dll" (ByVal tas as Double, ByVal trk as Double,
ByVal gs as Double, ByVal hdg as Double, ByRef ws as Double, ByRef wd as Double)
```

Input

[tas]: true airspeed (kts) [Valid range: 50 to 1500]
[trk]: true or magnetic route track (degrees) [Valid range: 0 to 360]
[gs]: ground speed (kts) [valid range 50 to 1500]
[hdg]: true or magnetic aircraft heading (degrees) [Valid range: 0 to 360]

Output [passed by reference in variables ws, wd]
ws: wind speed (kts)
wd: wind direction (degrees, true or magnetic)
Error return (out of range or indeterminate value): _NADBL (-1E+32)

Notes
Route track and aircraft heading can be entered as BOTH true or magnetic values; wind direction will be calculated accordingly
Result may be indeterminate in some cases

XHTWind

Description: calculates cross wind and head/tail wind components from reference direction, wind speed and wind direction

Declaration
C/C++

```
extern "C" _declspec(dllimport) void _stdcall XHTWind (double refdir, double ws, double wd, double &xwind, double &htwind)
```

VB

```
Declare Sub XHTWind Lib "Adacalc.dll" (ByVal refdir as Double, ByVal ws as Double, ByVal wd as Double, ByRef xwind as Double, ByRef htwind as Double)
```

Input
[refdir]: true or magnetic reference direction (degrees) [Valid range: 0 to 360]
[ws]: wind speed (kts) [valid range 0 to 150]
[wd]: true or magnetic wind direction (degrees) [Valid range: 0 to 360]

Output [passed by reference in variables xwind, htwind]
xwind: cross wind speed (kts) ; right is positive, left is negative
htwind: head/tail wind speed (kts) ; head wind is negative, tail wind is positive
Error return (out of range value or indeterminate): _NADBL (-1E+32)

Notes
Reference direction and wind direction can be entered as BOTH true or magnetic values

Turn calculations

Turn_Data

Description: calculates turn parameters from true airspeed (TAS), wanted turn rate and bank limitations

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Turn_Data (double tas, double wtr, double maxbank, double &ebank, double &etr, double &tr)
```

VB

```
Declare Sub Turn_Data Lib "Adacalc.dll" (ByVal tas as Double, ByVal wtr as Double, ByVal maxbank as Double, ByRef ebank as Double, ByRef etr as Double, ByRef tr as Double)
```

Input

[tas]: true airspeed (kts) [Valid range: 50 to 1500]
[wtr]: wanted turn rate (degrees/s) [Valid range: 1-5]
[maxbank]: maximum allowed bank (degrees) [Valid range: 20-40° or -1 for unlimited maximum bank]

Output [passed by reference in variables ebank, etr and tr]
ebank: effective bank (degrees)
etr: effective turn rate (degrees/s)
tr: turn radius (m)
Error return (out of range value): _NADBL (-1E+32)

Notes

IFR standard turn rate is 3°/s – Half standard rate is 1.5°/s
IFR maxbank is usually 25 or 30°
Effective bank (ebank) will be limited by [maxbank] input unless -1 is entered
Effective turn rate [etr] may be less than wanted turn rate [wtr] in some cases

Turn_Anticipation

Description: calculates turn anticipation to join a station from true airspeed (TAS), wanted turn rate, bank limitation, intercept angle and initial distance to station

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Turn_Anticipation (double tas, double wtr, double maxbank, double intercept_angle, double dstation, double &anticipation_angle, double &fdstation)
```

VB

```
Declare Sub Turn_Anticipation Lib "Adacalc.dll" (ByVal tas as Double, ByVal wtr as Double, ByVal maxbank as Double,ByVal intercept_angle as double, ByVal dstation as double, ByRef anticipation_angle as Double, ByRef fdstation as Double)
```

Input
[tas]: true airspeed (kts) [Valid range: 50 to 1500]
[wtr]: wanted turn rate (degrees/s) [Valid range: 1-5]
[maxbank]: maximum allowed bank (degrees) [Valid range: 20-40° or -1 for unlimited maximum bank]
[intercept_angle]: intercept angle (degrees): unsigned angle difference between feeding (current) track and final path to station [Valid range: 10-90°]
[dstation]: distance to station at start of turn (NM) [Valid range: 5-50 NM]

Output [passed by reference in variables anticipation_angle and fdstation]
Anticipation_angle: (degrees): angle difference from final track at which turn should be initiated (unsigned)
fdstation: distance to reference station at end of turn (NM)
Error return (out of range value): _NADBL (-1E+32)

Notes
IFR standard turn rate is 3°/s – Half standard rate is 1.5°/s
IFR maxbank is usually 25 or 30°
Assumes no wind conditions

Magnetic variation

MagVar

Description: calculates magnetic variation at any date from IGRF-12/IGRF-13 or WMM2015/WMM2020 magnetic model for 2015-2025

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall MagVar (double lat, double lon, double alt, int day, int month, int year, int model)
```

VB

```
Declare Function MagVar Lib "Adacalc.dll" (ByVal lat as Double, ByVal lon as Double, ByVal alt as Double, ByVal day as long, ByVal month as long, ByVal year as long, ByVal model as long) as Double
```

Input

[lat]: latitude (degrees) [Valid range: -90 to 90]
[lon]: longitude (degrees) [Valid range: -180 to 180]
[alt]: geodetic altitude (km) [valid range: 0 to 20]
[day]: day (should be valid for month and year)
[month]: month (1-12)
[year]: year (e.g. 2020)
[model]: magnetic model (0=IGRF ; 1=WMM)

Output

Magnetic variation (degrees) [see notes for sign convention]
Error return (out of range or indeterminate value): _NADBL (-1E+32)

Notes

Latitude, longitude and altitude are geodetic WGS 84 values
Calculation is limited to dates ranging from 1-Jan-2015 to 1-Jan-2025 (included)
East magnetic variations are positive, west are negative

RefYearMagVar

Description: calculates “reference” (sea level, January 1st) magnetic variation on a given year from IGRF-12/IGRF-13 or WMM2015/WMM2020

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall RefYearMagVar (double lat, double lon, int year, int model)
```

VB

```
Declare Function RefYearMagVar Lib "Adacalc.dll" (ByVal lat as Double, ByVal lon as Double, ByVal year as long, ByVal model as long) as Double
```

Input

[lat]: latitude (degrees) [Valid range: -90 to 90]
[lon]: longitude (degrees) [Valid range: -180 to 180]
[year]: year (e.g. 2020)
[model]: magnetic model (0=IGRF ; 1=WMM)

Output

Magnetic variation (degrees) [see notes for sign convention]
Error return (out of range or indeterminate value): _NADBL (-1E+32)

Notes

Latitude, longitude are geodetic WGS 84 values
Calculation is limited to years 2015 to 2025
East magnetic variations are positive and west are negative

Solar calculations

Sunrise_Sunset

Description: calculates sunrise and sunset times (UTC) (NOAA exact calculation)

Declaration
C/C++

```
extern "C" _declspec(dllimport) void _stdcall Sunrise_Sunset (double lat, double lon, int day, int month, int year, int &sunrise, int &sunset, int &diffday)
```

VB

```
Declare Sub Sunrise_Sunset Lib "Adacalc.dll" (ByVal lat as Double, ByVal lon as Double, ByVal day as long, ByVal month as long, ByVal year as long, ByRef sunrise as long, ByRef sunset as long, ByRef diffday as long)
```

Input
[lat]: latitude (degrees) [Valid range: -90 to 90]
[lon]: longitude (degrees) [Valid range: -180 to 180]
[day]: day of month (should be valid for month and year)
[month]: month of year [valid range: 1 to 12]
[year]: year [valid range: 1950 to 2049]

Output [passed by reference in variables sunrise, sunset, diffday]
sunrise: minutes from 0000Z time (e.g 211 = 02:31Z)
sunset: minutes from 0000Z time (e.g 1252 = 20:52Z)
[Note: in case of permanent day/night both values will return -1]
diffday: 0: sunrise and sunset on same day
-1: sunrise on previous day
+1: sunset on next day
Error return (sunrise and sunset) for out of range input: -2

Notes
Calculation is limited to dates from 1-Jan-1950 to 31-Dec-2049
Always check for sunrise and/or sunset = -2 for input error, then -1 (permanent day/night) before using the provided results

Dawn_Dusk

Description: calculates twilight (dawn and dusk) times (UTC) (NOAA exact calculation)

Declaration
C/C++

```
extern "C" _declspec(dllimport) void _stdcall Dawn_Dusk (double lat, double lon, int day, int month, int year, int ddref, int &dawn, int &dusk, int &diffday)
```

VB

```
Declare Sub Dawn_Dusk Lib "Adacalc.dll" (ByVal lat as Double, ByVal lon as Double, ByVal day as long, ByVal month as long, ByVal year as long, ByVal ddref as long, ByRef dawn as long, ByRef dusk as long, ByRef diffday as long)
```

Input
[lat]: latitude (degrees) [Valid range: -90 to 90]
[lon]: longitude (degrees) [Valid range: -180 to 180]
[day]: day of month (should be valid for month and year)
[month]: month of year [valid range: 1 to 12]
[year]: year [valid range: 1950 to 2049]
[ddref]: 0: civil time (aeronautical time)
1: nautical time
2: astronomical time

Output [passed by reference in variables dawn, dusk, diffday]
dawn: minutes from 0000Z time (e.g. 211 = 02:31Z)
dusk: minutes from 0000Z time (e.g. 1252 = 20:52Z)
[Note: in case of permanent day/night both values will return -1]
diffday: 0: dawn and dusk on same day
-1: dawn on previous day
+1: dusk on next day
Error return (dawn and dusk) for out of range input: -2

Notes
Calculation is limited to dates from 1-Jan-1950 to 31-Dec-2049
Always check for dawn and/or dusk = -2 for input error, then -1 (permanent day/night) before using the provided results

Miscellaneous

Changeover_Altitude

Description: pressure altitude (during climb or descent) at which constant CAS to constant mach number occurs

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Changeover_Altitude (double CAS, double mach, int unitflag)
```

VB

```
Declare Function Changeover_Altitude Lib "Adacalc.dll" (ByVal cas as double, ByVal mach as double, ByVal unitflag as long) as double
```

Input

[CAS]: calibrated airspeed (kts) [Valid range: 0 to 1500]
[mach]: mach number [Valid range: 0.05 to 3.0, <1 for AdacalcFS.dll]
[unitflag]: 0 for result in meters, 1 for result in feet

Output

Changeover altitude (m or ft depending on [unitflag] value)
Error return (out of range) or not calculable from provided CAS and mach number): _NADBL (-1E+32)

Notes

Some unrealistic combinations of CAS/Mach will result in a non-calculable result

---

DescentCalc

Description: accurately calculates time and distance for a fixed ROD or fixed slope descent from an initial to a final pressure altitude at a given Mach/CAS profile (2/3-step numerical integration)

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall DescentCalc (int flini, int flfin, double ias, double mach, int descentmode, double rodslope, double deltaisa, double windspeed, int r250, double &desc_nm, double &desc_min)
```

VB

```
Declare Sub DescentCalc Lib "Adacalc.dll" (ByVal flini as long, ByVal flfin as long, ByVal ias as double, ByVal mach as double, ByVal descentmode as long, ByVal rodslope as double, ByVal deltaisa as double, ByVal windspeed as double, ByVal r250 as long, ByRef desc_nm as double, ByRef desc_min as double)
```

Input

[flini]: initial flight level [Valid range: 30 to 650]
[flfin]: final flight level [Valid range: 0 to 650 and < flini]
[CAS]: descent calibrated air speed (50 to 1500 kts)

[mach]: descent mach number (0.05 to 3.00, <1 for FS version)[optional, enter 0 if CAS descent only]
[descentmode]: 0=fixed rate of descent (ROD); 1=fixed slope
[rodslope]: rate of descent if descentmode=0 (fpm, 300 to 5000), fixed slope if descentmode=1 (degrees, 1.5 to 7.5)
[deltaisa]: temperature offset from standard atmosphere (-90 to +70°C)
[windspeed]: average head/tail wind (-150 to 150 kts; head wind is negative, tail wind is positive)
[r250]: flag indicating CAS will be restricted to 250 KCAS below FL100 (1 or 0) if a greater value is provided

Output [passed by reference in variables desc_nm, desc_min]
desc_nm: descent ground distance (nm)
desc_min: time for descent (minutes, decimal)
Error return (see notes): _NADBL (-1E+32)

Notes
Enter Mach=0 for constant CAS descent; if both valid mach number and CAS are provided, descent calculation will use mach number until CAS is achieved
ROD values should be entered as positive values
In case descent profile is not calculable for whatever reason (inconsistent speeds, too strong wind data, etc) function will return both desc_nm and desc_min as _NADBL (-1E+32)

DegMinSec_To_Decimal

Description: converts degrees/minutes/seconds to a decimal degree value

Declaration
C/C++

```
extern "C" _declspec(dllimport) double _stdcall DegMinSec_To_Decimal (int deg, int min, double sec)
```

VB

```
Declare Function DegMinSec_To_Decimal Lib "Adacalc.dll" (ByVal deg as long, ByVal min as long, ByVal sec as double) as double
```

Input
[deg]: degrees [Valid range: -360 to 360]
[min]: minutes [Valid range: 0 to 59]
[sec]: seconds [Valid range: <60]

Output
Decimal degree value
Error return: _NADBL (-1E+32)

Decimal_To_DegMinSec

Description: converts decimal degrees to degrees/minutes/seconds

Declaration

C/C++

```
extern "C" _declspec(dllimport) void _stdcall Decimal_To_DegMinSec_To_Decimal (double degdeci, int sround, int &deg, int &min, double &sec)
```

VB

```
Declare Sub Decimal_To_DegMinSec Lib "Adacalc.dll" (ByVal degdeci as double, ByVal sround as long, ByRef deg as long, ByRef min as long, ByRef sec as double)
```

Input

[degdeci]: decimal degrees [Valid range: -360 to 360

[sround]: number of significant decimals for the seconds part returned value (1 to 6)

Output

Degrees, minutes and seconds passed by reference in variables deg, min, sec

Error return: 0 for deg and min, _NADBL (-1E+32) for sec

Unit_Conversion

Description: converts values in different units

Declaration

C/C++

```
extern "C" _declspec(dllimport) double _stdcall Unit_Conversion (double ivalue, int unitclass, int i_unitsubclass, int f_unitsubclass)
```

VB

```
Declare Function Unit_Conversion Lib "Adacalc.dll" (ByVal ivalue as double, ByVal unitclass as long, ByVal i_unitsubclass as long, ByVal f_unitsubclass as long) as double
```

Input

[ivalue]: original value

[unitclass]: Unit class (see valid values in table below)

[i_unitsubclass]: Original unit subclass (see valid values in table below)

[f_unitsubclass]: Converted unit subclass (see valid values in table below)

Output

Converted value

Error return: _NADBL (-1E+32)

Unit Class/subclass reference values

- Class=0 ACCELERATION
  - Subclass=0: ft/s2
  - Subclass=1: m/s2
  - Subclass=2: g-unit
- Class=1 ANGLES
  - Subclass=0: degrees
  - Subclass=1: radians
- Class=2 AREA/SURFACE
  - Subclass=0: ft^2

```
        Subclass=1: m^2
        Subclass=2: in^2
Class=3 DENSITY
        Subclass=0: lb/ft3
        Subclass=1: kg/m3
        Subclass=2: sl/ft3
Class=4 LENTH/DISTANCE
        Subclass=0: ft
        Subclass=1: m
        Subclass=2: km
        Subclass=3: nm
        Subclass=4: sm
        Subclass=5: in
Class=5 POWER
        Subclass=0: lb-ft/s
        Subclass=1: kg-m/s
        Subclass=2: watts
        Subclass=3: HP
Class=6 PRESSURE
        Subclass=0: hPa
        Subclass=1: inHg
        Subclass=2: psi
        Subclass=3: mmHg
Class=7 SPEED
        Subclass=0: Kt
        Subclass=1: km/h
        Subclass=2: ft/s (fps)
        Subclass=3: ft/min (fpm)
        Subclass=4: m/s
Class=8 TEMPERATURE
        Subclass=0: °C
        Subclass=1: °F
        Subclass=2: °K
Class=9 WEIGHT
        Subclass=0: lb
        Subclass=1: kg
```