Guide to Migration from Astro32

This document describes how to use supported Platform components to migrate away from Astrol32.dll. A key reason for migration is that Astro32 is a 32bit only DLL and so fails to work in 64bit applications. The alternate approaches described here are both 32 and 64bit compatible and can also be called from languages that support COM late binding.

Appendix 1 contains a list of Astro32 functions and their recommended Platform replacements or suggested computational approaches, while Appendix 2 lists the Astro32.Bas file that accompanied Astro32, which documented it's features.

Modified Julian Dates

Astro32 functions frequently work with "modified Julian dates" rather than "full Julian dates". Since the conversion is straightforward, no conversion function is provided within the Platform. Modified Julian dates are related to Julian dates by the formula:

JD = MJD + 2400000.5

The Platform does provide replacements for converting modified Julian dates to date values and to OLE automation dates.

Appendix 1 - Astro32 Astrometry Functions and Platform Equivalents

Function	Astro32.Bas Description	Recommended Platform 6 alternative
aa_hadec	' given latitude (n+, radians), lat, altitude (up+, radians), alt, and ' azimuth (angle around to the east from north+, radians), ' return hour angle (radians), ha, and declination (radians), dec.	**Use Astrometry.Transform** This accepts Alt/AZ co-ordinates and returns RA/Dec coordinates. To derive the hour angle, subtract the current sidereal time from the returned RA.
hadec_aa	' given latitude (n+, radians), lat, hour angle (radians), ha, and declination ' (radians), dec, return altitude (up+, radians), alt, and azimuth (angle ' round to the east from north+, radians),	**Use Astrometry.Transform** Convert the ha into a right ascension by subtracting it from the current sidereal time and then supply the RA/Dec to the Transform component.
nut_eq	' given the modified JD, mjd, correct, IN PLACE, the right ascension *ra ' and declination *dec (both in radians) for nutation.	**Use Astrometry.NOVAS.NOVAS31** Convert the RA/Dec to a vector using NOVAS31.RADec2Vector Calculate the nutated vector with NOVAS31.Nutate Convert the nutated vector back to RA/Dec with NOVAS31.Vector2RADec
nutation	' given the modified JD, mjd, find the nutation in obliquity, *deps, and ' the nutation in longitude, *dpsi, each in radians.	**Use Astrometry.NOVAS.NOVAS31.ETilt** Required values are returned directly but you must supply Julian dates rather than modified Julian dates.
obliquity	' given the modified Julian date, mjd, find the mean obliquity of the ' ecliptic, *eps, in radians.	**Use Astrometry.NOVAS.NOVAS31.ETilt** Required value is returned directly.
refract	' correct the true altitude, ta, for refraction to the apparent altitude, aa, ' each in radians, given the local atmospheric pressure, pr, in mbars, and ' the temperature, tr, in degrees C.	**Use Astrometry.NOVAS.NOVAS31.Refract** Create an ASCOM.Astrometry.OnSurface structure holding the local conditions such as temperature, height, pressure and provide this as a parameter to NOVAS31.Refract. Note that Refract works in zenith distance (90.0 - altitude) not altitude itself!
unrefract	' correct the apparent altitude, aa, for refraction to the true altitude, ta, ' each in radians, given the local atmospheric pressure, pr, in mbars, and ' the temperature, tr, in degrees C.	**Use Astrometry.AstroUtils.AstroUtils.UnRefract** This iteratively calls NOVAS3.Refract and adjusts the input unrefracted zenith distance until the refracted result matches the supplied observed zenith distance. Note that like NOVAS31.Refract, this routine works in zenith distance (90.0 - altitude) not altitude itself!

Astro32 Utility Functions

Function	Astro32.Bas Description	Recommended Platform 6 alternative
delra	' given the difference in two RA's, in rads, return their difference, ' accounting for wrap at 2*PI. caller need *not* first force it into the	**Use Astrometry.AstroUtils.Range** This flexible function provides a wide variety of ranging operations. To

Function	Astro32.Bas Description	Recommended Platform 6 alternative
	' range 0..2*PI.	recreate the delra function use a call of the form: RetVal = AstroUtils.Range(v, 0.0, True, 2 * PI, False)
range	' insure 0 <= *v < r. Used to range angles and times	**Use Astrometry.AstroUtils.Range** This flexible function provides a wide variety of ranging operations. To recreate the range function use a call of the form: RetVal = AstroUtils.Range(v, 0.0, True, r, False)

Astro32 Date and Time Methods

Function	Astro32.Bas Description	Recommended Platform 6 alternative
cal_mjd	' given a date in months, mn, days, dy, years, yr, ' return the modified Julian date (number of days elapsed since 1900 jan 0.5), ' *mjd.	**Use Astrometry.AstroUtils.CalendarToMJD** MJD = AstroUtils.CalendarToMJD(Day, Month, Year)
deltat	given the modified Julian date, mjd, find delta-T (TT-UTC)	**Use Astrometry.AstroUtils.DeltaT** DeltaT = AstroUtils.DeltaT
fmt_mjd	' Format a date string into buf, given a modified julian date and the ' selected format (m/d/y, etc.). Typically mm/dd.ddd/yyyy (note the ' fractional days).	**Use Astrometry.AstroUtils.FormatMJD** This function takes the modified Julian day and a formatting string as parameters and returns the formated date-time string. e.g. String = AstroUtils.FormatMJD(MJD, "HH:mm:ss.fff")

fmt_sexa	' format the Double (e.g., mjd, lst) in sexagesimal format into buf[].
	' w is the number of spaces for the whole part.
	' fracbase is the number of pieces a whole is to broken into; valid options:
	' 360000: <w>:mm:ss.ss
	' 36000: <w>:mm:ss.s
	' 3600: <w>:mm:ss
	' 600: <w>:mm.m
	' 60: <w>:mm
		Use ASCOM.Utilities.Util.
		There are many functions in this component for converting doubles to various sexagesimal forms.E.g.
		String = Util.DegreesToDMS(Double)
		String = Util.HoursToHM(Double)

gst_utc	' given a modified julian date, mjd, and a greenwich mean siderial time, gst, ' return universally coordinated time, *utc.	?? TBC - Bob to advise
mdy_vb	' Convert "MM/DD/YY" to VB Date	?? TBC - Bob to advise
mjd_2000	' return the Modified Julian Date of the epoch 2000	**Use 51544.5** This is a constant - the modified Julian day of the J2000 astronomical epoch that occurred at approximately 12:00 UTC on 1 January 2000.

Function	Astro32.Bas Description	Recommended Platform 6 alternative
mjd_cal	' given the modified Julian date, mjd, return the calendar date in months, *mn, ' days, *dy, and years, *yr.	**Use ASCOM.Utilities.Util.DateJulianToUTC or Util.DateJulianToLocal** Convert the modified Julian date to a Julian date Get a date back from Util.Date JulianToUTC or Util.DateJulianToLocal Parse the date to extract the information you need
mjd_day	' given an mjd, truncate it to the beginning of the whole day	
mjd_dow	' given an mjd, set *dow to 0..6 according to which day of the week it falls ' on (0=sunday). return 0 if ok else -1 if can't figure it out.	**Use ASCOM.Utilities.Util.DateJulianToUTC or Util.DateJulianToLocal** Convert the modified Julian date to a Julian date Get a date back from Util.Date JulianToUTC or Util.DateJulianToLocal Parse the date to extract the information you need
mjd_dpm	' given a mjd, return the the number of days in the month.	**Use ASCOM.Utilities.Util.DateJulianToUTC or Util.DateJulianToLocal** Convert the modified Julian date to a Julian date Get a date back from Util.Date JulianToUTC or Util.DateJulianToLocal Parse the date to extract the information you need
mjd_hr	' given an mjd, return the number of hours past midnight of the ' whole day	**Use ASCOM.Utilities.Util.DateJulianToUTC or Util.DateJulianToLocal** Convert the modified Julian date to a Julian date Get a date back from Util.Date JulianToUTC or Util.DateJulianToLocal Parse the date to extract the information you need
mjd_vb	' Return the Visual Basic Date given a Modified Julian Date	**Use Astrometry.AstroUtils.MJDToOADate** VbDate = AstroUtils.MJDToOADate(MJD)
mjd_year	' given a mjd, return the year as a double.	**Use ASCOM.Utilities.Util.DateJulianToUTC or Util.DateJulianToLocal** Convert the modified Julian date to a Julian date Get a date back from Util.Date JulianToUTC or Util.DateJulianToLocal Parse the date to extract the information you need
now_lst	' Return the current Local Apparent Sidereal Time (LST) from the clock and longitude (rad, - west) (+east)	**Use ASCOM.Utilities.Util.JulianDate and Astrometry.NOVAS.NOVAS31.SiderealTime** Get the Julian date from Util.JulianDate Get the Greenwich apparent sidereal time from NOVAS31.SiderealTime and the Julian date. Convert the longitude from radians to degrees Convert the longitude in degrees to hours by dividing by 15.0 Add the longitude to the Greenwich apparent Sidereal time to give local apparent sidereal time.
now_mjd	' Return the current Modified Julian Date derived from the system clock	**Use ASCOM.Utilities.Util.JulianDate** Double = Util.JulianDate

Function	Astro32.Bas Description	Recommended Platform 6 alternative
scn_date	' crack a floating date string, bp, of the form X/Y/Z determined by the ' DATE_DATE_FORMAT preference into its components. allow the day to be a ' floating point number. A lone component is always a year if it contains ' a decimal point or pref is MDY or DMY and it is not a reasonable month ' or day value, respectively. Leave any unspecified component unchanged. ' (actually, the slashes may be anything but digits or a decimal point) ' 'pref' indicates the format of the date (DATE_xxx).	**Create your own solution** This function is complex with many possible uses, we recommend that you implement a solution to match your specific needs.

scn_sexa	' scan a sexagesimal string and update a double. the string, bp, is of the form
	' H:M:S. a negative value may be indicated by a '-' char before any
	' component. All components may be integral or real. In addition to ':' the
	' separator may also be '/' or ';' or ',' or '-'.
	' any components not specified in bp[] are copied from old's in 'o'.
	' eg: ::10 only changes S
	' 10 only changes H
	' 10:0 changes H and M
		Use ASCOM.Utilities.Util.HMSToHours or Util.DMSToDegrees
		Double = ASCOM.Utilities.Util.HMSToHours and
		Double = ASCOM.Utilities.Util.DMSToDegrees
		There are other similar methods in the Util component that may also be helpful.

tz_name	' Fill buffer with the name of the current timezone, given a preference ' flag, pref, (DATE_UTCTZ = always "UTC", DATE_LOCALTZ = e.g., "PDT") ' Returns 0/1 indicating whether DST is currently in effect.	**Use ASCOM.Utilities.Util.TimeZoneName** String = Util.TimeZoneName
utc_gst	' given a modified julian DATE, mjd, and a universally coordinated time, utc, ' return greenwich mean siderial time, *gst. ' NOTE: mjd must be at the beginning of the day!	**Use Astrometry.NOVAS.NOVAS31.SiderealTime** Convert the modified Julian date to a Julian date Convert the time to a day fraction and add the Julian date Get the sidereal time from NOVAS31.SiderealTime
utc_offs	' Return the current UTC offset (+ = West) in seconds	**Use ASCOM.Utilities. Util.TimeZoneOffset** Get the time zone offset in hours from Util.TimeZoneOffset Multiply by 3600.0 to get seconds
vb_mjd	' Return a Modified Julian Date given a Visual Basic Date	**Use ASCOM.Utilities.Util.DateJulianToUTC** Convert the modified Julian date to a Julian date Get the date from Util.DateJulianToUTC
year_mjd	' given a decimal year, return mjd	**Use ASCOM.Utilities.Util.DateUTCToJulian** Express the year as a UTC date Get the Julian date from Util.DateUTCToJulian Convert the Julian date to a modified Julian date

Functions Provided by Astro32.Bas

Function	Implementation	Recommended Platform 6 alternative

degrad	`'   Degrees to Radians`
	`Public Function degrad(d As Double) As Double`
	`   degrad = (d * PI) / 180.0`
	`End Function`
		These are simple functions and can be implemented straightforwardly in your own program.

raddeg	`'   Radians to Degrees`
	`Public Function raddeg(r As Double) As Double`
	`   raddeg = (r * 180.0) / PI`
	`End Function`	

hrdeg	`'   Hours to Degrees`
	`Public Function hrdeg(h As Double)`
	`   hrdeg = h * 15.0`
	`End Function`	

deghr	`'   Degrees to Hours`
	`Public Function deghr(d As Double) As Double`
	`   deghr = d / 15.0`
	`End Function`	

hrrad	`'   Hours to Radians`
	`Public Function hrrad(h As Double) As Double`
	`   hrrad = degrad(hrdeg(h))`
	`End Function`	

radhr	`'   Radians to Hours`
	`Public Function radhr(r As Double) As Double`
	`   radhr = deghr(raddeg(r))`
	`End Function`	

Appendix 2 - Astro32.Bas

```
Attribute VB_Name = "AstronomyFuncs"
'-----------------------------------------------------------------------
'
'   ===========
'   ASTRO32.BAS
'   ===========
'
' Interface declarations for the Astronomy Library. Drop this into
' any VB project to get access to the astronomical support functions
' in astro32.dll. For the latest copy of astro32.dll, contact the
' author at the address below.
'
' Routines in astronomy DLL have been taken from various open source
' and freeware applications as well as original code by the author.
' Astro32.dll and this VB module are freely usable in any software
' project. The author assumes no responsibilities for bugs, etc.
'
' Written:  18-Jul-96   Robert B. Denny <rdenny@dc3.com>
'
' Edits:
'
' When      Who     What
' --------- ---     --------------------------------------------------
' 18-Jul-96 rbd     Initial edit (yes, 1996!)
' 19-Jul-98 rbd     1.2 of astro32.dll, add deltat()
' 20-Jul-98 rbd     Change comments on now_lst, change interface to take
'                   longitude - West
' 10-Aug-98 rbd     tz_name now returns 0/1 indicaing whether DST is in effect
' ---------------------------------------------------------------------------
Option Explicit

' You know what this is!
Public Const PI = 3.14159265358979
' Ratio of from synodic (solar) to sidereal (stellar) rate
Public Const SIDRATE = 0.9972695677
' Seconds per day
Public Const SPD = 86400#
'
' Modified Julian Date (MJD) calculations. The epoch for MJD is
'
Public Const MJD0 = 2415020#    ' MJD Julian epoch (JD = MJD + MJD0)
Public Const J2000 = 36525#     ' MJD for 2000 (2451545.0 - MJD0)
'
' Date formatting preferences for fmt_mjd() and scn_date()
'
Public Const DATE_YMD = 0
Public Const DATE_MDY = 1
Public Const DATE_DMY = 2

'
' Timezone name preferences for tz_name()
'
Public Const DATE_UTCTZ = 3
Public Const DATE_LOCALTZ = 4

'
' =================
' LIBRARY FUNCTIONS
' =================
'
' NOTES:
'
' (1) For whatever reason, the authors of the original C functions chose
'     to pass back and forth via parameters only for most functions.
'
' (2) The descriptive comments below were lifted straight out of the C
'     functions. Some variables are listed with the C dereferening '*'.
'     Note that these are passed ByRef in the declarations, then forget
'     about the '*'.
'
' (3) Modified Julian Dates (number of days elapsed since 1900 jan 0.5,)
'     are used for most times. Several functions are provided for converting
'     between mjd and other time systems (C runtime, VB, Win32).
'
'
' given latitude (n+, radians), lat, altitude (up+, radians), alt, and
' azimuth (angle around to the east from north+, radians),
' return hour angle (radians), ha, and declination (radians), dec.
'
Declare Sub aa_hadec Lib "astro32" (ByVal lat As Double, ByVal Alt As Double, ByVal Az As Double, ByRef ha As Double, ByRef dec As Double)
```

```
'
' given a date in months, mn, days, dy, years, yr,
' return the modified Julian date (number of days elapsed since 1900 jan 0.5),
' *mjd.
'
Declare Sub cal_mjd Lib "astro32" (ByVal mn As Long, ByVal dy As Double, ByVal yr As Long, ByRef mjd As Double)
'
' given the difference in two RA's, in rads, return their difference,
'   accounting for wrap at 2*PI. caller need *not* first force it into the
'   range 0..2*PI.
'
Declare Function delra Lib "astro32" (ByVal dRA As Double) As Double
'
' given the modified Julian date, mjd, find delta-T (TT-UTC)
'
Declare Function delta_t Lib "astro32" Alias "deltat" (ByVal mjd As Double) As Double
'
' Format a date string into buf, given a modified julian date and the
' selected format (m/d/y, etc.). Typically mm/dd.ddd/yyyy (note the
' fractional days).
'
Declare Sub fmt_mjd Lib "astro32" (ByVal buf As String, ByVal mjd As Double, ByVal pref As Long)
'
' format the Double (e.g., mjd, lst) in sexagesimal format into buf[].
' w is the number of spaces for the whole part.
' fracbase is the number of pieces a whole is to broken into; valid options:
'  360000: <w>:mm:ss.ss
'  36000:  <w>:mm:ss.s
'  3600:   <w>:mm:ss
'  600:    <w>:mm.m
'  60:     <w>:mm
'
Declare Sub fmt_sexa Lib "astro32" (ByVal buf As String, ByVal val As Double, ByVal w As Long, ByVal fracbase As
Long)
'
' given a modified julian date, mjd, and a greenwich mean siderial time, gst,
' return universally coordinated time, *utc.
'
Declare Sub gst_utc Lib "astro32" (ByVal mjd As Double, ByVal gst As Double, ByRef utc As Double)
'
' given latitude (n+, radians), lat, hour angle (radians), ha, and declination
'   (radians), dec, return altitude (up+, radians), alt, and azimuth (angle
'   round to the east from north+, radians),
'
Declare Sub hadec_aa Lib "astro32" (ByVal lat As Double, ByVal ha As Double, ByVal dec As Double, ByRef Alt As
Double, ByRef Az As Double)
'
' Convert "MM/DD/YY" to VB Date
'
Declare Function mdy_vb Lib "astro32" (ByVal mdy As String) As Date
'
' return the Modified Julian Date of the epoch 2000
'
Declare Function mjd_2000 Lib "astro32" () As Double
'
' given the modified Julian date, mjd, return the calendar date in months, *mn,
' days, *dy, and years, *yr.
'
Declare Sub mjd_cal Lib "astro32" (ByVal mjd As Double, ByRef mn As Long, ByRef dy As Double, ByRef yr As Long)
'
' given an mjd, truncate it to the beginning of the whole day
'
Declare Function mjd_day Lib "astro32" (ByVal jd As Double) As Double
'
' given an mjd, set *dow to 0..6 according to which day of the week it falls
' on (0=sunday). return 0 if ok else -1 if can't figure it out.
'
Declare Function mjd_dow Lib "astro32" (ByVal mjd As Double, ByRef dow As Long) As Long
'
' given a mjd, return the the number of days in the month.
'
Declare Sub mjd_dpm Lib "astro32" (ByVal mjd As Double, ByRef ndays As Long)
'
' given an mjd, return the number of hours past midnight of the
' whole day
'
Declare Function mjd_hr Lib "astro32" (ByVal jd As Double) As Double
'
' Return the Visual Basic Date given a Modified Julian Date
'
Declare Function mjd_vb Lib "astro32" (ByVal mjd As Double) As Date
'
' given a mjd, return the year as a double.
'
```

```
Declare Sub mjd_year Lib "astro32" (ByVal mjd As Double, ByRef yr As Double)
'
' Return the current Local Apparent Sidereal Time (LST) from the clock and longitude (rad, - west)
'
Declare Function now_lst Lib "astro32" (ByVal lng As Double) As Double
'
' Return the current Modified Julian Date derived from the system clock
'
Declare Function now_mjd Lib "astro32" () As Double
'
' given the modified JD, mjd, correct, IN PLACE, the right ascension *ra
' and declination *dec (both in radians) for nutation.
'
Declare Sub nut_eq Lib "astro32" (ByVal mjd As Double, ByRef RA As Double, ByRef dec As Double)
'
' given the modified JD, mjd, find the nutation in obliquity, *deps, and
' the nutation in longitude, *dpsi, each in radians.
'
Declare Sub nut Lib "astro32" Alias "nutation" (ByVal mjd As Double, ByRef deps As Double, ByRef dpsi As Double)
'
' given the modified Julian date, mjd, find the mean obliquity of the
' ecliptic, *eps, in radians.
'
Declare Sub obliq Lib "astro32" Alias "obliquity" (ByVal mjd As Double, ByRef eps As Double)
'
' insure 0 <= *v < r. Used to range angles and times
'
Declare Sub range Lib "astro32" (ByRef v As Double, ByVal r As Double)
'
' correct the true altitude, ta, for refraction to the apparent altitude, aa,
' each in radians, given the local atmospheric pressure, pr, in mbars, and
' the temperature, tr, in degrees C.
'
Declare Sub refract Lib "astro32" (ByVal pr As Double, ByVal tr As Double, ByVal ta As Double, ByRef aa As
Double)
'
' crack a floating date string, bp, of the form X/Y/Z determined by the
'   DATE_DATE_FORMAT preference into its components. allow the day to be a
'   floating point number. A lone component is always a year if it contains
'   a decimal point or pref is MDY or DMY and it is not a reasonable month
'   or day value, respectively. Leave any unspecified component unchanged.
'   ( actually, the slashes may be anything but digits or a decimal point)
'   'pref' indicates the format of the date (DATE_xxx).
'
Declare Function scn_date Lib "astro32" (ByVal dtstr As String, ByRef m As Long, ByRef d As Double, ByRef y As
Long, ByVal pref As Long)
'
' scan a sexagesimal string and update a double. the string, bp, is of the form
'   H:M:S. a negative value may be indicated by a '-' char before any
'   component. All components may be integral or real. In addition to ':' the
'   separator may also be '/' or ';' or ',' or '-'.
' any components not specified in bp[] are copied from old's in 'o'.
'   eg:  ::10  only changes S
'        10    only changes H
'        10:0  changes H and M
'
Declare Function scn_sexa Lib "astro32" (ByVal o As Double, ByVal sexa As String) As Double
'
' round a time in days, *t, to the nearest second, IN PLACE.
'
Declare Sub rnd_second Lib "astro32" (ByRef t As Double)
'
' Fill buffer with the name of the current timezone, given a preference
' flag, pref, (DATE_UTCTZ = always "UTC", DATE_LOCALTZ = e.g., "PDT")
' Returns 0/1 indicating whether DST is currently in effect.
'
Declare Function tz_name Lib "astro32" (ByVal buf As String, ByVal pref As Long) As Long
'
' correct the apparent altitude, aa, for refraction to the true altitude, ta,
' each in radians, given the local atmospheric pressure, pr, in mbars, and
' the temperature, tr, in degrees C.
'
Declare Sub unrefract Lib "astro32" (ByVal pr As Double, ByVal tr As Double, ByVal aa As Double, ByRef ta As
Double)
'
' given a modified julian DATE, mjd, and a universally coordinated time, utc,
' return greenwich mean siderial time, *gst.
' NOTE: mjd must be at the beginning of the day!
'
Declare Sub utc_gst Lib "astro32" (ByVal mjd As Double, ByVal utc As Double, ByRef gst As Double)
'
' Return the current UTC offset (+ = West) in seconds
'
Declare Function utc_offs Lib "astro32" () As Long
```

```
'
' Return a Modified Julian Date given a Visual Basic Date
'
Declare Function vb_mjd Lib "astro32" (ByVal d As Date) As Double
'
' given a decimal year, return mjd
'
Declare Sub year_mjd Lib "astro32" (ByVal y As Double, ByRef mjd As Double)

'----------------------------------------------------------------------
'   Degrees to Radians
'----------------------------------------------------------------------
Public Function degrad(d As Double) As Double
    degrad = (d * PI) / 180#
End Function

'----------------------------------------------------------------------
'   Radians to Degrees
'----------------------------------------------------------------------
Public Function raddeg(r As Double) As Double
    raddeg = (r * 180#) / PI
End Function

'----------------------------------------------------------------------
'   Hours to Degrees
'----------------------------------------------------------------------
Public Function hrdeg(h As Double)
    hrdeg = h * 15#
End Function

'----------------------------------------------------------------------
'   Degrees to Hours
'----------------------------------------------------------------------
Public Function deghr(d As Double) As Double
    deghr = d / 15#
End Function

'----------------------------------------------------------------------
'   Hours to Radians
'----------------------------------------------------------------------
Public Function hrrad(h As Double) As Double
    hrrad = degrad(hrdeg(h))
End Function

'----------------------------------------------------------------------
'   Radians to Hours
'----------------------------------------------------------------------
Public Function radhr(r As Double) As Double
    radhr = deghr(raddeg(r))
End Function
```