

# **GYPSY 2009 DLL Interface**

## **Version 2010.03.29.35**

### **Overview**

The GYPSY DLL provides one or more objects (called 'Stands') that provide methods for performing calculations according to the GYPSY document ("A Growth and Yield Projection System (GYPSY) for Natural and Post-harvest Stands in Alberta" by Shongming Huang, Shawn X. Meng and Yuqing Yang, May 21, 2009).

An individual stand contains 4 'Species' objects and provides methods for setting the forecast characteristics and accessing the Species objects. Each Species object provides methods for setting the observed values as well as obtaining the various index and output values for the given observed values.

All calculated values are based on "lazy evaluation", where calculations are performed only if needed to supply a value requested by the calling application. The DLL also provides some of the low level functions directly, without the need of a Stand object.

## Installation

### ***Registering the GYPSY 2009 DLL***

The GYPSY DLL is a Microsoft COM type DLL which means that it needs to be registered on your system. After registration, the DLL components can be freely accessed by any program without having to explicitly reference it again. The DLL can be placed in any folder of your choice on your system. Registering it differs slightly on Windows 95/2000/XP and Vista operating systems.

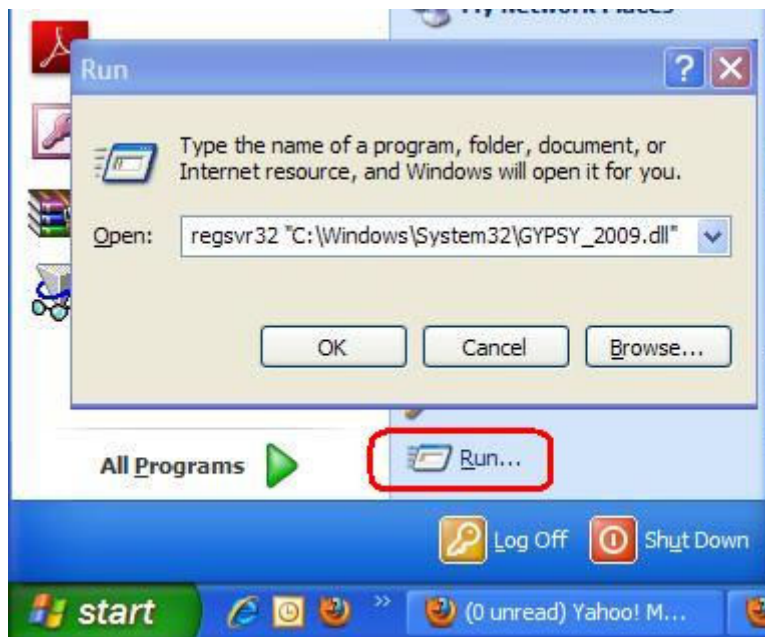


Figure 1. Registering the DLL on Windows XP

### ***Windows 95/2000/XP***

1. Click the Start button located in the lower left corner of your screen.
2. Select Run as shown in Figure 1.
3. Enter `regsvr32 "path\to\your\dll\folder\GYPSY_2009.dll"`

for example if you placed the DLL into `C:\Windows\System32\` then

regsvr32 "C:\Windows\System32\GYPSY\_2009.dll" will need to be entered. Make sure that you also use the double quotation marks as given.

4. Click the OK button.

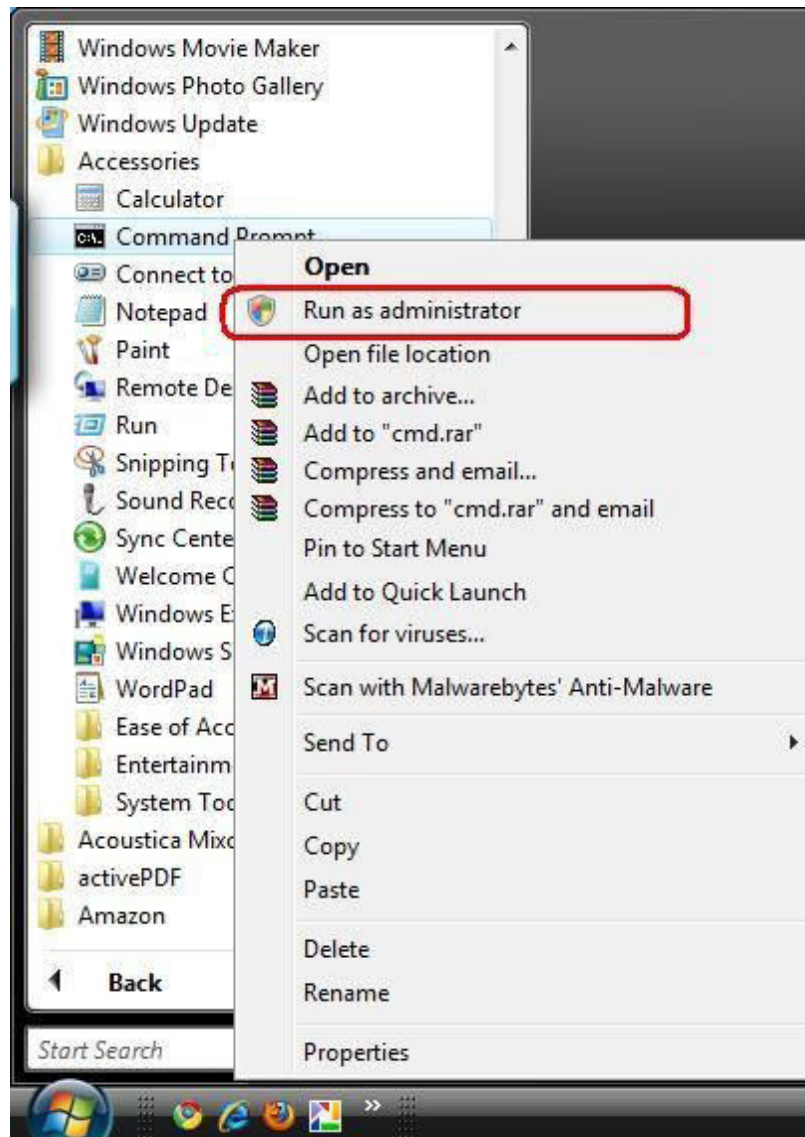


Figure 2. Registering the DLL on Windows Vista

## **Windows Vista**

1. Click the Start button located in the lower left corner of your screen.
2. Select the Accessories folder.

3. Right click on the Command Prompt icon.
4. Select Run as administrator from the menu as shown in Figure 2.
5. Enter `regsvr32 "path\to\your\dll\folder\GYPSY_2009.dll"` at the prompt.

for example if you placed the DLL into `C:\Windows\System32\` then `regsvr32 "C:\Windows\System32\GYPSY_2009.dll"` will need to be entered. Make sure that you also use the double quotation marks as given.

6. Close the Command window.

**Note:** You must follow these instructions on a Vista machine, even if you are logged in as Administrator!

If your registration was successful, you will see a small window pop up with a message similar to the one shown in Figure 3.

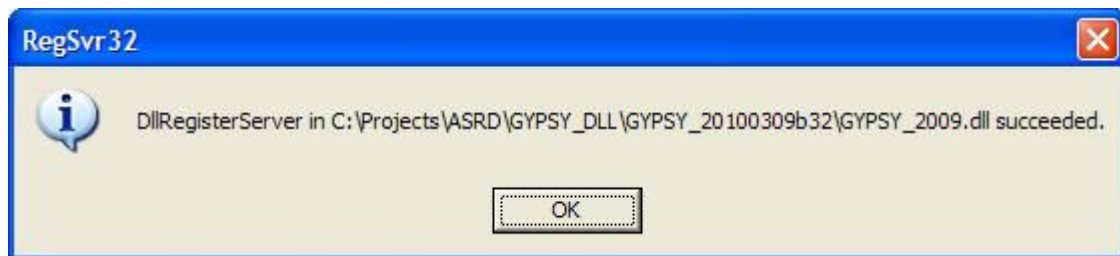


Figure 3. Successful registration of the DLL

## Species Used

The GYPSY DLL uses an integer to indicate species, and provides a set of functions for converting species codes as strings to the appropriate integer. A value of -1 is used to indicate an invalid species code.

```
Integer num_species;  
Integer species_index(String species_code);  
    // where species_code is "AW", "SB", "SW", or "PL"  
Integer invalid_species;
```

## Direct Functions

Some of the lower level functions and attributes are available directly (without use of a Stand object).

```
Integer DLL_version;    // (((year * 100) + month) * 100)  
                        + day  
Integer DLL_build;     // exact build number of DLL  
Integer GYPSY_version; // year month day of reference  
                        GYPSY document  
Integer SAS_version;   // year month day of reference SAS  
                        code  
Double top_height(Integer species_index, Double SI_t,  
Double tage);
```

## Managing Stand objects

The GYPSY DLL provides the following functions for creating, enumerating and disposing of stand objects:

```
Integer Add(String description);
```

```

    // returns the stand_id of a newly created stand
Stand *stand(int stand_id);
    // returns the Stand with stand_id
void Delete(Integer stand_id);
    // releases all storage for stand_id

```

## The Stand object

The stand object provides functions and attributes for identifying the Stand itself and accessing stand characteristics, managing the Species objects, setting forecast and utilization characteristics and for detecting and reporting errors and warnings.

### ***Stand characteristics***

Each Stand object has the following attributes:

```

Integer stand_id; /* read only */
String description; /* Read/write */
Double stand_age; /* Read/write */
Double projection_offset;
    /* Read/write, default (stand_age-floor(stand_age)) */
Integer projection_origin; /* Read only */
    // The earliest acceptable projected_year, usually 0,
    // but may be < 0 when stand age < max(species ages).
Double p_stand_age(Integer projected_year);
    /* Read only */
    /* defined to be projection_offset + projected_year */
Integer species_observed(Integer species_index);
    /* Read only, 1 if den > 0 or N0 > 0 */
Integer species_present(Integer sndx, Double
forecast_stnd_age);

```

```

    /* Read only, 1 if species_observed(sndx) <> 0 and
       the species relative total age corresponding to the
       forecast_stnd_age is >= 0.0 */
Integer species_present_bh(Integer sndx, Double
    forecast_stnd_age);
    /* Read only, 1 if species_observed(sndx) <> 0 and the
       species relative breast height age corresponding to
       the forecast_stnd_age is >= 0.0 */

```

The Stand object also provides the following methods:

```

void clear_calculations();
    /* clear away any calculated values */
void clear_all_inputs();
    /* clear away all inputs and calcs. */

```

## ***Accessing Species***

Each Stand object also contains 4 Species objects corresponding to the 4 GYPSY species and provides the following method for accessing the Species objects:

```

Species *species(Integer species_index);

```

## ***Forecast and Utilization Characteristics***

The stand object provides the following attributes for setting forecast characteristics:

```

Integer spatial;
    /* Read/write */
    // set spatial to 0 to disable the spatial model
    // if set to 0, all observed PS values are ignored.

```

```

// if set to 1, observed PS values are used for the
// spatial model when available, falling back to non-
// spatial, if not set, defaults to 0 if there are ANY
// observed species missing PS or with PS <= 0,
// otherwise 1 (i.e. 1 if PS set and PS > 0 for all
// observed species)
Integer species_spatial(Integer species_index);
/* Read only */
// 0 if spatial set to 0, or PS not set, or PS <= 0,
// else 1.
Integer BA_known;
/* Read/write */
// 0 to use BA, 1 to use locBA
// if set to 0, all observed BA values are ignored.
// if set to 1, observed BA values are used to localize
// p_locBA when available, falling back to p_BA when
// not. if not set, defaults to 1 if there are ANY
// observed species with observed BA
Double sdob_con; /* Read/write, default 15.0 */
// stump diameter outside bark (cm) for conifer
Double stht_con; /* Read/write, default 0.3 */
// stump height (m) for conifer
Double tdib_con; /* Read/write, default 10.0 */
// top diameter inside bark (cm) for conifer
Double sdob_dec; /* Read/write, default 15.0 */
// stump diameter outside bark (cm) for deciduous
Double stht_dec; /* Read/write, default 0.3 */
// stump height (m) for deciduous
Double tdib_dec; /* Read/write, default 10.0 */
// top diameter inside bark (cm) for deciduous

```



## ***Errors and Warnings***

Each Stand object provides independent errors and warnings, which are cleared whenever the calculated values are cleared, through the following attributes:

```
Integer error_count;  
String error_message(Integer error_index);  
Integer warning_count;  
String warning_message(Integer warning_index);
```

The following checks are made for each species on any given input attributes:

```
tage < 7.5 for AW and PL (error)  
tage < 9.5 for SB and SW (error)  
PS < 0.0 or PS > 100.0 (error)  
missing both den and N0 if PS given (error)  
missing PS if den > 0 and Stand.spatial set to 1 (error)  
den < 50.0 (warning)  
N0 < 50.0 (warning)  
topht < 1.3 or topht > 35.0 for AW (warning)  
topht < 0.3 or topht > 35.0 for SB, SW, and PL (warning)  
SI_bh < 5.0 or SI_bh > 30.0 (warning)  
SI_t < 5.0 or SI_t > 30.0 (warning)  
PS <= 60.0 and den or N0 >= 1500 (warning)  
PS > 50.0 and den or N0 < 500 (warning)  
BA > 60.0 (warning)
```

In addition, if the stand level attribute stand\_age is set,

```
stand_age < 10.0 or stand_age > 250.0 (warning).
```

Input checks are performed (if required) when the error\_count or warning\_count attributes are first read. Reading a calculated or forecast value may also

generate errors if there are missing input values. Applications should make use of the `species_observed` attribute to avoid requesting calculated or forecast values of absent species.

## The Species object

The Species object provides attribute for input values, index values (which may also be input), species specific utilization levels, and forecast values. The Species object also provides attributes for “projected” values, where a projected value is just a forecast value for a particular (integer) projected year. In other words, a projected value for projected year PY is simply the forecast value at the `forecast_stand_age = (projection_offset + PY)`. Some of the Species attributes accept an arbitrary Double `forecast_stand_age`, or `forecast_species_age` parameter, and others (whose names start with “p\_”) accept an Integer `projected_year`.

### ***Input Attributes***

The Species object provides the following attributes:

```
Integer species_id;
    /* Read only */
Double tage;
    /* Read/write, observed total age */
Integer given_tage;
    /* Read only, 1 if tage has been set, else 0 */
Double breast_height_age;
    /* rarely used, read/write */
Integer given_breast_height_age;
    /* Read only, 1 if breast_height_age has been set */
Double stump_age;
    /* rarely used, read/write */
```

```

Integer given_stump_age;
    /* Read only, 1 if stump_age is set */
Double bage;
    /* Read only, calculated from tage and y2bh */
Double sage;
    /* Read only, calculated from tage and y2sh */
Double topht;
    /* Read/write, observed top height */
Integer given_topht;
    /* Read only, 1 if topht has been set */
Double ps;
    /* Read/write, observed percent stocking */
Integer given_ps;
    /* Read only, 1 if ps has been set, else 0 */
Double den;
    /* Read/write, observed density. */
Integer given_den;
    /* Read only, 1 if den has been set, else 0 */
Double BA;
    /* Read/write, observed BA. */
Integer given_BA;
    /* Read only, 1 if BA has been set, else 0 */

```

## ***Index Attributes***

Index attributers are usually calculated from the input attributes, but some (marked as read/write) may be used as input values.

```

Double SI_t;
    /* Read/write, site index at total age 50 */
Integer given_SI_t;
    /* Read only, 1 if SI_t has been set, else 0 */

```

```

Double SI_bh;
    /* Read/write, site index at breast height age 50 */
Integer given_SI_bh;
    /* Read only, 1 if SI_bh has been set */
Double y2bh;
    /* Read only, years to breast height */
Double y2sh;
    /* Read only, years to stump height */
Double psi;
    /* Read only, percent stocking index (if ps given) */
Double N0;
    /* Read/write, density at age 0 (Aspen=bha, rest=ta) */
Integer given_N0;
    /* Read only, 1 if N0 has been set, else 0 */
Double SDF;
    /* Read only, stand density factor */
Double germination_time;
    /* Read only, stand_age - species_tage */
    // set to a large number (2000+) if species not observed

```

## ***Forecast Attributes***

Most of the forecast attributes accept a forecast\_species\_age parameter, but some (e.g. SC) accept a forecast stand age because the underlying equation uses values from other species in the stand. Note that the density forecast attribute is base on a forecast\_species\_breast\_height\_age for Aspen, not a total age.

```

Double density(Double forecast_species_age);
    // again, Aspen=BH age
Double SC(Double forecast_stand_age);
    // Read only, STAND total age

```

```

Double SC_bh(Double forecast_stand_age); /* Read only */
    // STAND total age, but densities = 0
    // for species < breast ht.
Double top_height(Double forecast_species_total_age);
    // Uses SI_t and total_age to return the top height
    // at total_age
Double percent_stocking(Double forecast_species_total_age)

```

### ***Utilization Attributes***

The Species object provides three optional utilization attributes for projecting merchantable volume. If these attributes are set, they are used in preference to the stand utilization attributes.

```

Double sdob;
    /* Read/write, stump diameter outside bark (cm) */
Double stht;
    /* Read/write, stump height (m) */
Double tdib;
    /* Read/write, top diameter inside bark (cm) */

```

### ***Projection Attributes***

The Species object provides a number of projection related attributes, all parameterized by an Integer projection age, which is converted to a projected stand age by adding the Stand.projection\_offset. The projected stand age is then converted to a species total or breast height age as needed. All projected attributes are read only.

```

Double p_tage(Integer projected_year);
    // projected total age
    // = Stand.p_stand_age(projected_year) -

```

```

Species.germination_time
Double p_bage(Integer projected year);
    // projected breast height age
    // = p_tage(projected_year)-y2bh
Double p_topht(Integer projected year);
    // projected top height
Double p_ps(Integer projected year);
    // projected percent stocking
Double p_den(Integer projected year);
    // projected density
Double p_SC(Integer projected year);
    // projected species composition
    // at p_stand_age(projected_year)
Double p_SC_bh(Integer projected year);
    // projected species composition
    // at p_stand_age(projected_year)
    // where densities of species < breast height
    // are set to 0.
Double p_BA(Integer projected year);
    // projected basal area at an integral projected bh_age
    // (this projected bh_age is not the same
    // as p_bage above)
Double p_BAinc(Integer projected year);
    // projected basal area increment at the
    // same age as p_BA
Double p_locBA(Integer projected year);
    // localized p_BA if BA observed and
    // Stand.BA_known <> 0,
    // otherwise the same as p_BA
Double p_locBAinc(Integer projected year);
    // localized p_BAinc if BA observed
    // and Stand.BA_known <> 0,

```

```

        // otherwise the same as p_BAinc
Double p_tv(Integer projected year);
        // projected gross total volume based on p_locBA
Double p_mv(Integer projected year);
        // projected merchantable volume according
        // to the species utilization
        // attributes if given, otherwise according
        // to the stand utilization attributes of the
        // appropriate species type
Double p_tmai(Integer projected year);
        // projected mai from total volume and
        // species total age
Double p_mmai(Integer projected year);
        // projected mai from merch volume and
        // species total age
Double p_tmai2(Integer projected year);
        // projected mai from total volume and stand total age
Double p_mmai2(Integer projected year);
        // projected mai from merch volume and stand total age
Double p_mdbh(Integer projected year);
        // projected DBH from utilization level
Double p_cum(Integer projected year);
        // projected CUM from utilization level
Double p_ms(Integer projected year);
        // projected merch. stems per ha from utilization level

```

## Example Visual Basic for Applications Code

The following example shows some aspects of using the GYPSY DLL from VBA in Excel.

```
Sub GYPSY_Tester()  
' GYPSY 2009 COM DLL test harness  
' Date: March 11, 2010  
' Copyright: Timberline Natural Resource Group Ltd. 2010  
  
Dim cell As Range  
Dim myAw As species  
Dim mySb As species  
Dim mySw As species  
Dim myPl As species  
  
Application.ScreenUpdating = False  
  
' create a new GYPSY object  
Set objGYPSY = New GYPSY2009  
  
' setup the proper species index numbers  
' AW=0 SB=1 SW=2 PL=3 Invalid species = -1  
AW = objGYPSY.species_index("AW")  
SB = objGYPSY.species_index("SB")  
SW = objGYPSY.species_index("SW")  
PL = objGYPSY.species_index("PL")  
  
' add a stand to the GYPSY object  
x = objGYPSY.Add("Stand 1")  
Set myStand = objGYPSY.Stands(CInt(x))
```



' setup the 4 species objects within the stand

Set myAw = myStand.species(AW)

Set mySb = myStand.species(SB)

Set mySw = myStand.species(SW)

Set myPl = myStand.species(PL)

' set the spatial flag if given

If Range("is\_spatial").Value <> "" Then

    myStand.spatial = Range("is\_spatial").Value

End If

' set the stand age if given (otherwise the max. species total age is used

If Range("stand\_age").Value <> "" Then

    myStand.stand\_age = Range("stand\_age").Value

End If

' set ba\_known if given (for basal area adjustment)

If Range("ba\_adjust").Value <> "" Then

    myStand.BA\_known = Range("ba\_adjust").Value

End If

' set utilization...all species the same for this test

' utilization for conifer and deciduous can be set at the stand level

' but utilization can also be set at the individual species level

' for example myAw.sdob = 13 will set the aspen min. stump diameter to 13 cm

' 15 cm stump dob is the default if not set

If Range("stumpdob").Value <> "" Then

    myStand.sdob\_con = Range("stumpdob").Value

    myStand.sdob\_dec = Range("stumpdob").Value

End If

' 0.3 m stump height is the default if not set

If Range("stumph").Value <> "" Then

    myStand.stht\_con = Range("stumph").Value

    myStand.stht\_dec = Range("stumph").Value

End If

' 10 cm top dib is the default if not set

If Range("topdib").Value <> "" Then

    myStand.tdib\_con = Range("topdib").Value

    myStand.tdib\_dec = Range("topdib").Value

End If

' clear output ranges before repopulating with data

Range("out\_index").Cells.ClearContents

Range("out\_proj").Cells.ClearContents

' load Aw input

If Range("B4").Value <> "" Then

    myAw.tage = Range("B4").Value

End If

If Range("C4").Value <> "" Then

    myAw.topht = Range("C4").Value

End If

If Range("D4").Value <> "" Then

    myAw.den = Range("D4").Value

End If

If Range("E4").Value <> "" Then

    myAw.ps = Range("E4").Value

End If

If Range("F4").Value <> "" Then

```

    myAw.BA = Range("F4").Value
End If

' load Sb input
If Range("B5").Value <> "" Then
    mySb.tage = Range("B5").Value
End If
If Range("C5").Value <> "" Then
    mySb.topht = Range("C5").Value
End If
If Range("D5").Value <> "" Then
    mySb.den = Range("D5").Value
End If
If Range("E5").Value <> "" Then
    mySb.ps = Range("E5").Value
End If
If Range("F5").Value <> "" Then
    mySb.BA = Range("F5").Value
End If

```

```

' load Sw input
If Range("B6").Value <> "" Then
    mySw.tage = Range("B6").Value
End If
If Range("C6").Value <> "" Then
    mySw.topht = Range("C6").Value
End If
If Range("D6").Value <> "" Then
    mySw.den = Range("D6").Value
End If
If Range("E6").Value <> "" Then
    mySw.ps = Range("E6").Value

```

```

End If
If Range("F6").Value <> "" Then
    mySw.BA = Range("F6").Value
End If

' load PI input
If Range("B7").Value <> "" Then
    myPl.tage = Range("B7").Value
End If
If Range("C7").Value <> "" Then
    myPl.topht = Range("C7").Value
End If
If Range("D7").Value <> "" Then
    myPl.den = Range("D7").Value
End If
If Range("E7").Value <> "" Then
    myPl.ps = Range("E7").Value
End If
If Range("F7").Value <> "" Then
    myPl.BA = Range("F7").Value
End If

```

'INDEX values are output

```

' output Aw calculated values
If myStand.species_observed(AW) <> 0 Then
    Range("G4").Value = myAw.SI_bh
    Range("H4").Value = myAw.SI_t
    Range("I4").Value = myAw.y2bh
    Range("J4").Value = myAw.y2sh
    Range("K4").Value = myAw.bage
    Range("L4").Value = myAw.sage

```

```

Range("M4").Value = myAw.psi
Range("N4").Value = myAw.SDF
Range("O4").Value = myAw.N0
End If

```

' output Sb calculated values

```

If myStand.species_observed(SB) <> 0 Then
    Range("G5").Value = mySb.SI_bh
    Range("H5").Value = mySb.SI_t
    Range("I5").Value = mySb.y2bh
    Range("J5").Value = mySb.y2sh
    Range("K5").Value = mySb.bage
    Range("L5").Value = mySb.sage
    Range("M5").Value = mySb.psi
    Range("N5").Value = mySb.SDF
    Range("O5").Value = mySb.N0
End If

```

' output Sw calculated values

```

If myStand.species_observed(SW) <> 0 Then
    Range("G6").Value = mySw.SI_bh
    Range("H6").Value = mySw.SI_t
    Range("I6").Value = mySw.y2bh
    Range("J6").Value = mySw.y2sh
    Range("K6").Value = mySw.bage
    Range("L6").Value = mySw.sage
    Range("M6").Value = mySw.psi
    Range("N6").Value = mySw.SDF
    Range("O6").Value = mySw.N0
End If

```

' output PI calculated values

```
If myStand.species_observed(PL) <> 0 Then
```

```
    Range("G7").Value = myPl.SI_bh
```

```
    Range("H7").Value = myPl.SI_t
```

```
    Range("I7").Value = myPl.y2bh
```

```
    Range("J7").Value = myPl.y2sh
```

```
    Range("K7").Value = myPl.bage
```

```
    Range("L7").Value = myPl.sage
```

```
    Range("M7").Value = myPl.psi
```

```
    Range("N7").Value = myPl.SDF
```

```
    Range("O7").Value = myPl.NO
```

```
End If
```

### ' PROJECTIONS

```
For Each cell In Range("proj_ages").Cells ' stand age
```

```
py = cell.Value
```

### ' AW

```
If myStand.species_observed(AW) <> 0 Then
```

```
    cell.Offset(0, 1).Value = myAw.p_tage(py)
```

```
    cell.Offset(0, 2).Value = myAw.p_bage(py)
```

```
    cell.Offset(0, 3).Value = myAw.p_topht(py)
```

```
    cell.Offset(0, 4).Value = myAw.p_den(py)
```

```
    cell.Offset(0, 5).Value = myAw.p_ps(py)
```

```
    cell.Offset(0, 6).Value = myAw.p_SC(py)
```

```
    cell.Offset(0, 7).Value = myAw.p_locBA(py)
```

```
    cell.Offset(0, 8).Value = myAw.p_ms(py)
```

```
    cell.Offset(0, 9).Value = myAw.p_mv(py)
```

```
    cell.Offset(0, 10).Value = myAw.p_mmai2(py)
```

```
End If
```

' SB

```
If myStand.species_observed(SB) <> 0 Then
    cell.Offset(0, 11).Value = mySb.p_tage(py)
    cell.Offset(0, 12).Value = mySb.p_bage(py)
    cell.Offset(0, 13).Value = mySb.p_topht(py)
    cell.Offset(0, 14).Value = mySb.p_den(py)
    cell.Offset(0, 15).Value = mySb.p_ps(py)
    cell.Offset(0, 16).Value = mySb.p_SC(py)
    cell.Offset(0, 17).Value = mySb.p_locBA(py)
    cell.Offset(0, 18).Value = mySb.p_ms(py)
    cell.Offset(0, 19).Value = mySb.p_mv(py)
    cell.Offset(0, 20).Value = mySb.p_mmai2(py)
End If
```

' SW

```
If myStand.species_observed(SW) <> 0 Then
    cell.Offset(0, 21).Value = mySw.p_tage(py)
    cell.Offset(0, 22).Value = mySw.p_bage(py)
    cell.Offset(0, 23).Value = mySw.p_topht(py)
    cell.Offset(0, 24).Value = mySw.p_den(py)
    cell.Offset(0, 25).Value = mySw.p_ps(py)
    cell.Offset(0, 26).Value = mySw.p_SC(py)
    cell.Offset(0, 27).Value = mySw.p_locBA(py)
    cell.Offset(0, 28).Value = mySw.p_ms(py)
    cell.Offset(0, 29).Value = mySw.p_mv(py)
    cell.Offset(0, 30).Value = mySw.p_mmai2(py)
End If
```

' PL

```
If myStand.species_observed(PL) <> 0 Then
    cell.Offset(0, 31).Value = myPl.p_tage(py)
    cell.Offset(0, 32).Value = myPl.p_bage(py)
```

```

cell.Offset(0, 33).Value = myPl.p_topht(py)
cell.Offset(0, 34).Value = myPl.p_den(py)
cell.Offset(0, 35).Value = myPl.p_ps(py)
cell.Offset(0, 36).Value = myPl.p_SC(py)
cell.Offset(0, 37).Value = myPl.p_locBA(py)
cell.Offset(0, 38).Value = myPl.p_ms(py)
cell.Offset(0, 39).Value = myPl.p_mv(py)
cell.Offset(0, 40).Value = myPl.p_mmai2(py)
End If

```

Next cell

' output errors

```

Range("rng_errors").Cells.ClearContents
ec = 2
For i = 0 To myStand.error_count - 1
    Range("V" & ec).Value = myStand.error_message(i)
    ec = ec + 1
    If ec > 6 Then Exit For ' run out of space
Next i

```

Range("W1").Value = ec - 2 ' actual number of errors

' output warnings

```

Range("rng_warnings").Cells.ClearContents
wc = 2
For i = 0 To myStand.warning_count - 1
    Range("AD" & wc).Value = myStand.warning_message(i)
    wc = wc + 1
    If wc > 6 Then Exit For ' run out of space
Next i

```



```
Range("AE1").Value = wc - 2 ' actual number of warnings
```

```
' output actual stand age used for checking purposes
```

```
Range("calc_sa").Value = myStand.stand_age
```

```
' clears all values of the stand object
```

```
myStand.clear_all_inputs
```

```
' output version numbers for the GYPSY object
```

```
Range("gypsy_version").Value = Format(Format _  
    (objGYPSY.GYPSY_version, "0000-00-00"), "yyyy-mmm-dd")
```

```
Range("sas_version").Value = Format(Format _  
    (objGYPSY.SAS_version, "0000-00-00"), "yyyy-mmm-dd")
```

```
Range("dll_version").Value = Format(Format _  
    (objGYPSY.DLL_version, "0000-00-00"), "yyyy-mmm-dd")
```

```
Range("dll_build").Value = objGYPSY.DLL_build
```

```
' housekeeping to clear all objects
```

```
Set myAw = Nothing
```

```
Set mySb = Nothing
```

```
Set mySw = Nothing
```

```
Set myPl = Nothing
```

```
Set myStand = Nothing
```

```
objGYPSY.Delete (CInt(x))
```

```
Set objGYPSY = Nothing
```

```
' replace -1 values quickly
```

```
' these are empty cells
```

```
For Each c In Range("out_index").Cells
    If c.Value = -1 Then c.Value = ""
Next c

For Each c In Range("out_proj").Cells
    If c.Value = -1 Then c.Value = ""
Next c

Application.ScreenUpdating = True

End Sub
```