**Using the National Volume Estimator Library DLL**

The National Volume Estimator Library DLL is a FORTRAN DLL compiled using Intel Visual Fortran version 10. The following files are the volume library files:

vollib.lib

vollib.dll

**C++ access to the volume library**

Additionally, the following files are provided to aid the developer in linking the National Volume Estimator Library DLL into a C++ application:

DllVolume.cpp C++ class used to call the National Volume Estimator Library DLL

DllVolume.h Header file for DllVolume.cpp file.

f77matrx.h Class used to convert FORTRAN arrays into C++ arrays

f77char.h Class used to convert FORTRAN strings into C++ strings

The following instructions describe the procedure to link in and access the National Volume Estimator Library DLL using the provided DllVolume class. The example was run with Visual C++ version 6.

Run Visual C++ and start a new project. Once the C++ project is set up, add the files VolLib.lib, DllVolume.cpp, DllVolume.h, f77matrx.h, and f77char.h to your project. The file VolLib.dll will need to reside in the same directory as your project’s executable.

The values for the parameter list will need to be set using the ‘Set’ member function for each variable. Once the variables have been set, a call into the DllVolume routine will reconfigure the C++ variables into FORTRAN readable variables, call the National Volume Estimator Library DLL and return the volumes. The volume information can be retrieved using the ‘Get’ member functions.

**C# .net access to the volume library**

Also included in this download is a C# application that can be used for testing the volume library on individual trees. This application also includes and an updated version of the Profile Model Tutorial originally developed by Ken Cormier. The new version allows users to display and compare up to 2 trees, change merchandizing rules and change models.

The PMT only works, however for a subset of profile models. These include Clark, Flewelling, Wenzel/Olsen, Demars and a few others. In addition to the PMT, for certain equations, you can also get the diameter inside bark (DIB) at a user defined height. Exaples of how to access the volume library via C# .net are including in the files Form1.cs, MRules.cs and PMTForm.cs

**Python 2.7 access to the volume library**

Included in this download is also an example (vollibTest.py) of how to access the volume library via python version 2.7. As of 3/7/2011 this application has not undergone extensive testing but has worked in all tests runs conducting here at the Forest Management Service Center.

**Additional information about the volume library**

Most projects described in this document are currently developed in and compatible with Visual Studio 2008. If you wish to see the implementation of these applications or would like the entire source code and project files or have any other questions regarding the National Volume Estimator Library then please email [yingfangwang@fs.fed.us](mailto:yingfangwang@fs.fed.us) .

// **DllVolume.cpp**: implementation of the DllVolume class.

//

//////////////////////////////////////////////////////////////////////

#include "stdafx.h"

#include <stdio.h>

#include "DllVolume.h"

#ifdef \_DEBUG

#undef THIS\_FILE

static char THIS\_FILE[]=\_\_FILE\_\_;

#define new DEBUG\_NEW

#endif

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

CDllVolume::CDllVolume()

{

m\_iErrflag = 0;

}

CDllVolume::~CDllVolume()

{

}

void CDllVolume::Get\_DllVolume()

{

// standard variables

int REGN,HTLOG,HTREF,FCLASS,HTTFLL,ERRFLAG,TLOGS,BA,SI;

int CUTFLG,BFPFLG,CUPFLG,CDPFLG,CUSFLG,CDSFLG,SPFLG,VERSION;

float DBHOB,DRCOB,HTTOT,HT1PRD,HT2PRD,STUMP;

float UPSHT1,UPSHT2,UPSD1,UPSD2,AVGZ1,AVGZ2;

float DBTBH,BTR,MTOPP,MTOPS,NOLOGP,NOLOGS;

// array defintions

int I3 = 3;

int I7 = 7;

int I15 = 15;

int I20 = 20;

int I21 = 21;

const int i3 = 3;

const int i7 = 7;

const int i15 = 15;

const int i20 = 20;

const int i21 = 21;

float VOL[i15];

float LOGLEN[i20];

float BOLHT[i21];

// 2-dimentional array definitions

FMATRIX<float> LOGVOL(i7,i20);

FMATRIX<float> LOGDIA(i21,i3);

// character defintions

const size\_t len2=2;

const size\_t len3=3;

const size\_t len5=5;

const size\_t len11=11;

// char buff[10];

char cVoleq[11],cHttype[2],cLive[2],cForst[3],cProd[3],cConspec[5],cCtype[2];

// assign the Fortran variables

sprintf(cVoleq,"%s",m\_sVoleq);

sprintf(cHttype,"%s",m\_sHttype);

sprintf(cLive,"%s",m\_sLive);

if(m\_sForst == "") m\_sForst = "10"; // check for null forest number

sprintf(cForst,"%s",m\_sForst);

sprintf(cProd,"%s",m\_sProd);

sprintf(cConspec,"%s",m\_sConspec);

CString sCtype = "F";

sprintf(cCtype,"%s",sCtype);

CHARACTER HTTYPE(cHttype,len2);

CHARACTER LIVE(cLive,len2);

CHARACTER FORST(cForst,len3);

CHARACTER PROD(cProd,len3);

CHARACTER CONSPEC(cConspec,len5);

CHARACTER VOLEQ(cVoleq,len11);

CHARACTER CTYPE(cCtype,len2);

REGN = m\_iRegn;

MTOPP = m\_fMtopp;

MTOPS = m\_fMtops;

STUMP = m\_fStump;

DBHOB = m\_fDbhob;

DRCOB = m\_fDrcob;

HTTOT = m\_fHttot;

HTLOG = m\_iHtlog;

HT1PRD = m\_fHt1prd;

HT2PRD = m\_fHt2prd;

UPSHT1 = m\_fUpsht1;

UPSHT2 = m\_fUpsht2;

UPSD1 = m\_fUpsd1;

UPSD2 = m\_fUpsd2;

HTREF = m\_iHtref;

AVGZ1 = m\_fAvgz1;

AVGZ2 = m\_fAvgz2;

FCLASS = m\_iFclass;

DBTBH = m\_fDbtbh;

BTR = m\_fBtr;

CUTFLG = m\_iCutflg;

BFPFLG = m\_iBfpflg;

CUPFLG = m\_iCupflg;

CDPFLG = m\_iCdpflg;

// CUSFLG = m\_iCusflg;

// CDSFLG = m\_iCdsflg;

SPFLG = m\_iSpflg;

HTTFLL = m\_iHttfll;

TLOGS = m\_iTlogs;

NOLOGP = 0.0;

NOLOGS = 0.0;

BA = 0;

SI = 0;

for(int nn=0;nn<20;nn++)

LOGLEN[nn] = m\_fLoglen[nn];

ERRFLAG = 0;

// call to version number

VERNUM(&VERSION);

m\_nVersionNum = VERSION;

// call to FORTRAN VolLib.DLL. All non-character variables are passed as references.

VOLUMELIBRARY(&REGN,FORST,VOLEQ,&MTOPP,&MTOPS,&STUMP,&DBHOB,&DRCOB,HTTYPE,&HTTOT,

&HTLOG,&HT1PRD,&HT2PRD,&UPSHT1,&UPSHT2,&UPSD1,&UPSD2,&HTREF,&AVGZ1,

&AVGZ2,&FCLASS,&DBTBH,&BTR,I3,I7,I15,I20,I21,VOL,LOGVOL,LOGDIA,LOGLEN,

BOLHT,&TLOGS,&NOLOGP,&NOLOGS,&CUTFLG,&BFPFLG,&CUPFLG,&CDPFLG,&SPFLG,

CONSPEC,PROD,&HTTFLL,LIVE,&BA,&SI,CTYPE,&ERRFLAG);

// save FORTRAN VolLib.DLL variables to DllVolume member variables

m\_iErrflag = ERRFLAG;

m\_iTlogs = TLOGS;

m\_fNologp = NOLOGP;

m\_fNologs = NOLOGS;

for(int i=0;i<15;i++)

m\_fVol[i] = VOL[i];

for(int j=0;j<20;j++)

{

m\_fLoglen[j] = LOGLEN[j];

m\_fLogvol\_Bdft[j] = LOGVOL(0,j); // scribner board foot volumes for each log

m\_fLogvol\_Cuft[j] = LOGVOL(3,j); // merchantable cubic foot volumes for each log

}

for(int k=0;k<20;k++)

{

m\_fLogsc[k] = LOGDIA(k,0);

m\_fLogdib[k] = LOGDIA(k,1);

m\_fLogdob[k] = LOGDIA(k,2);

m\_fBolht[k] = BOLHT[k];

}

}

**///////////////////////////////////////////////////////////////////////////**

**// the following are all the ‘Set’ and ‘Get’ routines**

**///////////////////////////////////////////////////////////////////////////**

// Voleq - Volume Equation Number

CString CDllVolume::Get\_Voleq()

{

return m\_sVoleq;

}

void CDllVolume::Set\_Voleq(CString sVoleq)

{

m\_sVoleq = sVoleq;

}

// sForst - Forest Number

CString CDllVolume::Get\_Forst()

{

return m\_sForst;

}

void CDllVolume::Set\_Forst(CString sForst)

{

m\_sForst = sForst;

}

// sProd - Product

CString CDllVolume::Get\_Prod()

{

return m\_sProd;

}

void CDllVolume::Set\_Prod(CString sProd)

{

m\_sProd = sProd;

}

// VosHttyp - Height type (L = logs, F = feet)

CString CDllVolume::Get\_Httype()

{

return m\_sHttype;

}

void CDllVolume::Set\_Httype(CString sHttype)

{

m\_sHttype = sHttype;

}

// sLive - (L = Live, D = Dead)

CString CDllVolume::Get\_Live()

{

return m\_sLive;

}

void CDllVolume::Set\_Live(CString sLive)

{

m\_sLive = sLive;

}

// sConspec - Contract species

CString CDllVolume::Get\_Conspec()

{

return m\_sConspec;

}

void CDllVolume::Set\_Conspec(CString sConspec)

{

m\_sConspec = sConspec;

}

// iHtref - Reference height

int CDllVolume::Get\_Htref()

{

return m\_iHtref;

}

void CDllVolume::Set\_Htref(int iHtref)

{

m\_iHtref = iHtref;

}

// iFclass - Form Class

int CDllVolume::Get\_Fclass()

{

return m\_iFclass;

}

void CDllVolume::Set\_Fclass(int iFclass)

{

m\_iFclass = iFclass;

}

// iHttfll - Height to first live limb

int CDllVolume::Get\_Httfll()

{

return m\_iHttfll;

}

void CDllVolume::Set\_Httfll(int iHttfll)

{

m\_iHttfll = iHttfll;

}

// iRegn - Region number

int CDllVolume::Get\_Regn()

{

return m\_iRegn;

}

void CDllVolume::Set\_Regn(int iRegn)

{

m\_iRegn = iRegn;

}

// iHtlog - Height in logs

int CDllVolume::Get\_Htlog()

{

return m\_iHtlog;

}

void CDllVolume::Set\_Htlog(int iHtlog)

{

m\_iHtlog = iHtlog;

}

// iCutflg - Total cubic volume flag (1 = calculate this volume)

int CDllVolume::Get\_Cutflg()

{

return m\_iCutflg;

}

void CDllVolume::Set\_Cutflg(int iCutflg)

{

m\_iCutflg = iCutflg;

}

// iBfpflg - Board foot volume flag (1 = calculate this volume)

int CDllVolume::Get\_Bfpflg()

{

return m\_iBfpflg;

}

void CDllVolume::Set\_Bfpflg(int iBfpflg)

{

m\_iBfpflg = iBfpflg;

}

// iCupflg – Merchantable cubic volume flag (1 = calculate this volume)

int CDllVolume::Get\_Cupflg()

{

return m\_iCupflg;

}

void CDllVolume::Set\_Cupflg(int iCupflg)

{

m\_iCupflg = iCupflg;

}

// iCdpflg - Mainstem cord wood volume flag (1 = calculate this volume)

int CDllVolume::Get\_Cdpflg()

{

return m\_iCdpflg;

}

void CDllVolume::Set\_Cdpflg(int iCdpflg)

{

m\_iCdpflg = iCdpflg;

}

// iCusflg - Cubic foot volume for topwood (1 = calculate this volume)

int CDllVolume::Get\_Cusflg()

{

return m\_iCusflg;

}

void CDllVolume::Set\_Cusflg(int iCusflg)

{

m\_iCusflg = iCusflg;

}

// iCdsflg - Cordwood volume for topwood (1 = calculate this volume)

int CDllVolume::Get\_Cdsflg()

{

return m\_iCdsflg;

}

void CDllVolume::Set\_Cdsflg(int iCdsflg)

{

m\_iCdsflg = iCdsflg;

}

// iSpflg - Topwood (1 = calculate this volume)

int CDllVolume::Get\_Spflg()

{

return m\_iSpflg;

}

void CDllVolume::Set\_Spflg(int iSpflg)

{

m\_iSpflg = iSpflg;

}

// iTlogs - Total number of logs in the tree (mainstem + topwood)

int CDllVolume::Get\_Tlogs()

{

return m\_iTlogs;

}

void CDllVolume::Set\_Tlogs(int iTlogs)

{

m\_iTlogs = iTlogs;

}

// iErrflag - Volume error flag

int CDllVolume::Get\_Errflag()

{

// 1 = No volume equation match

// 2 = No form class

// 3 = DBH less than one

// 4 = Tree height less than 4.5

// 5 = D2H is out of bounds

// 6 = No species match

// 7 = Illegal primary product log height (Ht1prd)

// 8 = Illegal secondary product log height (Ht2prd)

// 9 = Upper stem measurements required

return m\_iErrflag;

}

void CDllVolume::Set\_Errflag(int iErrflag)

{

m\_iErrflag = iErrflag;

}

// fDbhob - DBH outside bark

float CDllVolume::Get\_Dbhob()

{

return m\_fDbhob;

}

void CDllVolume::Set\_Dbhob(float fDbhob)

{

m\_fDbhob = fDbhob;

}

// fDrcob - Diameter at root collar

float CDllVolume::Get\_Drcob()

{

return m\_fDrcob;

}

void CDllVolume::Set\_Drcob(float fDrcob)

{

m\_fDrcob = fDrcob;

}

// fHttot - Total tree height

float CDllVolume::Get\_Httot()

{

return m\_fHttot;

}

void CDllVolume::Set\_Httot(float fHttot)

{

m\_fHttot = fHttot;

}

// fHt1prd - Height to a merchantable top diameter

float CDllVolume::Get\_Ht1prd()

{

return m\_fHt1prd;

}

void CDllVolume::Set\_Ht1prd(float fHt1prd)

{

m\_fHt1prd = fHt1prd;

}

// fHt2prd - Height to a secondary product top diameter

float CDllVolume::Get\_Ht2prd()

{

return m\_fHt2prd;

}

void CDllVolume::Set\_Ht2prd(float fHt2prd)

{

m\_fHt2prd = fHt2prd;

}

// fDbtbh - Double bark thickness at breast height

float CDllVolume::Get\_Dbtbh()

{

return m\_fDbtbh;

}

void CDllVolume::Set\_Dbtbh(float fDbtbh)

{

m\_fDbtbh = fDbtbh;

}

// fBtr - Bark thickness ratio

float CDllVolume::Get\_Btr()

{

return m\_fBtr;

}

void CDllVolume::Set\_Btr(float fBtr)

{

m\_fBtr = fBtr;

}

// fUpsht1 - Height as some upper stem diameter

float CDllVolume::Get\_Upsht1()

{

return m\_fUpsht1;

}

void CDllVolume::Set\_Upsht1(float fUpsht1)

{

m\_fUpsht1 = fUpsht1;

}

// fUpsht2 - Second height to some second upper stem diameter

float CDllVolume::Get\_Upsht2()

{

return m\_fUpsht2;

}

void CDllVolume::Set\_Upsht2(float fUpsht2)

{

m\_fUpsht2 = fUpsht2;

}

// fUpsd1 - Diameter at some upper stem height

float CDllVolume::Get\_Upsd1()

{

return m\_fUpsd1;

}

void CDllVolume::Set\_Upsd1(float fUpsd1)

{

m\_fUpsd1 = fUpsd1;

}

// fUpsd2 - Second diameter at some second upper stem height

float CDllVolume::Get\_Upsd2()

{

return m\_fUpsd2;

}

void CDllVolume::Set\_Upsd2(float fUpsd2)

{

m\_fUpsd2 = fUpsd2;

}

// fAvgz1 - Flewelling’s Average Z-Score

float CDllVolume::Get\_Avgz1()

{

return m\_fAvgz1;

}

void CDllVolume::Set\_Avgz1(float fAvgz1)

{

m\_fAvgz1 = fAvgz1;

}

// fAvgz2 - Second Flewelling’s average Z-score

float CDllVolume::Get\_Avgz2()

{

return m\_fAvgz2;

}

void CDllVolume::Set\_Avgz2(float fAvgz2)

{

m\_fAvgz2 = fAvgz2;

}

// fMtopp - Merchantable top diameter for primary product

float CDllVolume::Get\_Mtopp()

{

return m\_fMtopp;

}

void CDllVolume::Set\_Mtopp(float fMtopp)

{

m\_fMtopp = fMtopp;

}

// fMtops - Merchantable top diameter for secondary prodcut

float CDllVolume::Get\_Mtops()

{

return m\_fMtops;

}

void CDllVolume::Set\_Mtops(float fMtops)

{

m\_fMtops = fMtops;

}

// fStump - Stump height

float CDllVolume::Get\_Stump()

{

return m\_fStump;

}

void CDllVolume::Set\_Stump(float fStump)

{

m\_fStump = fStump;

}

// fNologp - Number of logs in the main stem

float CDllVolume::Get\_Nologp()

{

return m\_fNologp;

}

void CDllVolume::Set\_Nologp(float fNologp)

{

m\_fNologp = fNologp;

}

// fNologs - Number of logs in the topwood

float CDllVolume::Get\_Nologs()

{

return m\_fNologs;

}

void CDllVolume::Set\_Nologs(float fNologs)

{

m\_fNologs = fNologs;

}

// Vol - Tree volumes array

float CDllVolume::Get\_Vol(int i)

{

// 0 = total cubic

// 1 = gross scribner boards

// 2 = net scribner boards

// 3 = gross cubic

// 4 = net cubic

// 5 = gross cordwood

// 6 = Gross secondary product volume in cubic feet

// 7 = Net secondary product volume in cubic feet

// 8 = Secondary product in cordwood

// 9 = Gross International ¼ board foot volume

// 10 = Net International ¼ board foot volume

// 11 = Gross secondary product volume in scribner boards

// 12 = Net secondary product volume in scribner boards

return m\_fVol[i];

}

void CDllVolume::Set\_Vol(float fVol,int i)

{

m\_fVol[i] = fVol;

}

// fLoglen - log lengths array

float CDllVolume::Get\_Loglen(int i)

{

// i = log number

return m\_fLoglen[i];

}

void CDllVolume::Set\_Loglen(float fLoglen,int i)

{

m\_fLoglen[i] = fLoglen;

}

// fLogvol\_Bdft - Log volume in Scribner board feet array

float CDllVolume::Get\_Logvol\_Bdft(int i)

{

// i = log number

return m\_fLogvol\_Bdft[i];

}

void CDllVolume::Set\_Logvol\_Bdft(float fLogvol\_Bdft,int i)

{

m\_fLogvol\_Bdft[i] = fLogvol\_Bdft;

}

// fLogvol\_Cuft - Log volume in Smailian’s cubic foot array

float CDllVolume::Get\_Logvol\_Cuft(int i)

{

// i = log number

return m\_fLogvol\_Cuft[i];

}

void CDllVolume::Set\_Logvol\_Cuft(float fLogvol\_Cuft,int i)

{

m\_fLogvol\_Cuft[i] = fLogvol\_Cuft;

}

// fLogsc - Log end scaling diameters array

float CDllVolume::Get\_Logsc(int i)

{

// i = large end diameter for log number i, small diameter for log number i-1

return m\_fLogsc[i];

}

void CDllVolume::Set\_Logsc(float fLogsc,int i)

{

m\_fLogsc[i] = fLogsc;

}

// fLogdib - Log end diameter inside bark array

float CDllVolume::Get\_Logdib(int i)

{

// i = large end diameter for log number i, small diameter for log number i-1

return m\_fLogdib[i];

}

void CDllVolume::Set\_Logdib(float fLogdib,int i)

{

m\_fLogdib[i] = fLogdib;

}

// fLogdob - Log end diameter outside bark array

float CDllVolume::Get\_Logdob(int i)

{

// i = large end diameter for log number i, small diameter for log number i-1

return m\_fLogdob[i];

}

void CDllVolume::Set\_Logdob(float fLogdob,int i)

{

m\_fLogdob[i] = fLogdob;

}

// fBolht - Log end heights up the bole array

float CDllVolume::GetBolhtl(int i)

{

// i = log number

return m\_fBolht[i];

}

void CDllVolume::Set\_Bolht(float fBolht,int i)

{

m\_fBolht[i] = fBolht;

}

// **DllVolume.h**: interface for the DllVolume class.

//

//////////////////////////////////////////////////////////////////////

#if !defined(AFX\_DLLVOLUME\_H\_\_7B232070\_8E04\_11D4\_B542\_0006290DDAA5\_\_INCLUDED\_)

#define AFX\_DLLVOLUME\_H\_\_7B232070\_8E04\_11D4\_B542\_0006290DDAA5\_\_INCLUDED\_

#if \_MSC\_VER > 1000

#pragma once

#endif // \_MSC\_VER > 1000

#include "f77matrx.h"

#include "f77char.h"

// Define version number

extern "C" \_\_declspec(dllimport) void \_stdcall VERNUM(int \*VERSION);

// Define Fortran DLL calling parameters

extern "C" \_\_declspec(dllimport) void \_stdcall VOLUMELIBRARY(int \*REGN,CHARACTER FORST,

CHARACTER VOLEQ,float \*MTOPP,float \*MTOPS,float \*STUMP,float \*DBHOB,

float \*DRCOB,CHARACTER HTTYPE,float \*HTTOT,int \*HTLOG,float \*HT1PRD,

float \*HT2PRD,float \*UPSHT1,float \*UPSHT2,float \*UPSD1,float \*UPSD2,

int \*HTREF,float \*AVGZ1,float \*AVGZ2,int \*FCLASS,float \*DBTBH,float \*BTR,

int& I3,int& I7,int& I15,int& I20,int& I21,float\* VOL,float\* LOGVOL,

float\* LOGDIA,float\* LOGLEN,float\* BOLHT, int \*TLOGS,float \*NOLOGP,

float \*NOLOGS,int \*CUTFLG,int \*BFPFLG,int \*CUPFLG,int \*CDPFLG,int \*SPFLG,

CHARACTER CONSPEC,CHARACTER PROD,int \*HTTFLL,CHARACTER LIVE,

int \*BA,int \*SI,CHARACTER CTYPE,int \*ERRFLAG);

// Define Fortran DLL calling parameters

extern "C" \_\_declspec(dllimport) void \_stdcall VOLLIB(int \*REGN,CHARACTER FORST,

CHARACTER VOLEQ,float \*MTOPP,float \*MTOPS,float \*STUMP,float \*DBHOB,

float \*DRCOB,CHARACTER HTTYPE,float \*HTTOT,int \*HTLOG,float \*HT1PRD,

float \*HT2PRD,float \*UPSHT1,float \*UPSHT2,float \*UPSD1,float \*UPSD2,

int \*HTREF,float \*AVGZ1,float \*AVGZ2,int \*FCLASS,float \*DBTBH,float \*BTR,

int& I3,int& I7,int& I11,int& I20,int& I21,float\* VOL,float\* LOGVOL,

float\* LOGDIA,float\* LOGLEN,float\* BOLHT, int \*TLOGS,float \*NOLOGP,

float \*NOLOGS,int \*CUTFLG,int \*BFPFLG,int \*CUPFLG,int \*CDPFLG,int \*CUSFLG,

int \*CDSFLG,CHARACTER CONSPEC,CHARACTER PROD,int \*HTTFLL,CHARACTER live,int \*ERRFLAG);

class CDllVolume

{

private:

CString m\_sVoleq; // Volume equation number

CString m\_sForst; // Forest number

CString m\_sProd; // Product code

CString m\_sHttype; // Height type (F = feet, L = logs)

CString m\_sLive; // Live code (L = live, D = dead)

CString m\_sConspec; // Contract Species

int m\_iHtref; // Reference Height

int m\_iFclass; // Form Class

int m\_iHttfll; // Height to first live limb

int m\_iRegn; // Region number

int m\_iHtlog; // Length of logs used if Httype = L

int m\_iTlogs; // Number of predicted logs in the tree

int m\_iErrflag; // Error flag

int m\_iCutflg; // Total cubic foot volume flag

int m\_iBfpflg; // Board foot volume flag

int m\_iCupflg; // Merch cubic volume flag

int m\_iCdpflg; // Merch cord wood volume flag

int m\_iCusflg; // Cubic secondary product volume flag

int m\_iCdsflg; // Cord wood secondary product volume flag

int m\_iSpflg; // Secondary product volume flag

float m\_fDbhob; // Dbh outside bark

float m\_fDrcob; // Diameter at root collar

float m\_fHttot; // Total tree height in feet

float m\_fHt1prd; // Height to primary product minimum top diameter (feet or logs)

float m\_fHt2prd; // Height to secondary product minimum top diameter (feet or logs)

float m\_fDbtbh; // Double bark thickness at breast height

float m\_fBtr; // Bark thickness ratio

float m\_fUpsht1; // Upper stem height

float m\_fUpsht2; // Second Upper stem height

float m\_fUpsd1; // Upper stem diameter taken at Upper stem height

float m\_fUpsd2; // Second Upper stem diameter taken at Second Upper stem height

float m\_fAvgz1; // Average Z-score: applied at Upsht1 or Refht

float m\_fAvgz2; // Second Average Z-score: applied at Usht2

float m\_fMtopp; // Merchantable top diameter for primary product. inside bark

float m\_fMtops; // Merchantable top diameter for secondary product, inside bark

float m\_fStump; // Stump height

float m\_fNologp; // Number of 16 foot logs for primary product

float m\_fNologs; // Number of 16 foot logs for secondary product

float m\_fVol[15]; // Tree volumes

float m\_fLoglen[20]; // Individual log lengths

float m\_fLogvol\_Bdft[20]; // Individual log board foot volumes

float m\_fLogvol\_Cuft[20]; // Individual log cubic foot volumes

float m\_fLogsc[21]; // Individual log scaling diameters (rounded to nearest inch)

float m\_fLogdib[21]; // Individual log diameters inside bark

float m\_fLogdob[21]; // Individual log diameters outside bark

float m\_fBolht[21]; // Height up stem where each log diameter was predicted

public:

CDllVolume();

virtual ~CDllVolume();

void Get\_DllVolume();

CString Get\_Voleq();

void Set\_Voleq(CString sVoleq);

CString Get\_Forst();

void Set\_Forst(CString sForst);

CString Get\_Prod();

void Set\_Prod(CString sProd);

CString Get\_Httype();

void Set\_Httype(CString sHttype);

CString Get\_Live();

void Set\_Live(CString sLive);

CString Get\_Conspec();

void Set\_Conspec(CString sConspec);

int Get\_Htref();

void Set\_Htref(int iHtref);

int Get\_Fclass();

void Set\_Fclass(int iFclass);

int Get\_Httfll();

void Set\_Httfll(int iHttfll);

int Get\_Regn();

void Set\_Regn(int iRegn);

int Get\_Htlog();

void Set\_Htlog(int iHtlog);

int Get\_Cutflg();

void Set\_Cutflg(int iCutflg);

int Get\_Bfpflg();

void Set\_Bfpflg(int iBfpflg);

int Get\_Cupflg();

void Set\_Cupflg(int iCupflg);

int Get\_Cdpflg();

void Set\_Cdpflg(int iCdpflg);

int Get\_Cusflg();

void Set\_Cusflg(int iCusflg);

int Get\_Cdsflg();

void Set\_Cdsflg(int iCdsflg);

int Get\_Spflg();

void Set\_Spflg(int iSpflg);

int Get\_Tlogs();

void Set\_Tlogs(int iTlogs);

int Get\_Errflag();

void Set\_Errflag(int iErrflag);

float Get\_Dbhob();

void Set\_Dbhob(float fDbhob);

float Get\_Drcob();

void Set\_Drcob(float fDrcob);

float Get\_Httot();

void Set\_Httot(float fHttot);

float Get\_Ht1prd();

void Set\_Ht1prd(float fHt1prd);

float Get\_Ht2prd();

void Set\_Ht2prd(float fHt2prd);

float Get\_Dbtbh();

void Set\_Dbtbh(float fDbtbh);

float Get\_Btr();

void Set\_Btr(float fBtr);

float Get\_Upsht1();

void Set\_Upsht1(float fUpsht1);

float Get\_Upsht2();

void Set\_Upsht2(float fUpsht2);

float Get\_Upsd1();

void Set\_Upsd1(float fUpsd1);

float Get\_Upsd2();

void Set\_Upsd2(float fUpsd2);

float Get\_Avgz1();

void Set\_Avgz1(float fAvgz1);

float Get\_Avgz2();

void Set\_Avgz2(float fAvgz2);

float Get\_Mtopp();

void Set\_Mtopp(float fMtopp);

float Get\_Mtops();

void Set\_Mtops(float fMtops);

float Get\_Stump();

void Set\_Stump(float fStump);

float Get\_Nologp();

void Set\_Nologp(float fNologp);

float Get\_Nologs();

void Set\_Nologs(float fNologs);

float Get\_Vol(int i);

void Set\_Vol(float fVol,int i);

float Get\_Loglen(int i);

void Set\_Loglen(float fLoglen,int i);

float Get\_Logvol\_Bdft(int i);

void Set\_Logvol\_Bdft(float fLogvol\_Bdft,int i);

float Get\_Logvol\_Cuft(int i);

void Set\_Logvol\_Cuft(float fLogvol\_Cuft,int i);

float Get\_Logsc(int i);

void Set\_Logsc(float fLogsc,int i);

float Get\_Logdib(int i);

void Set\_Logdib(float fLogdib,int i);

float Get\_Logdob(int i);

void Set\_Logdob(float fLogdob,int i);

float GetBolhtl(int i);

void Set\_Bolht(float fBolht,int i);

};

#endif // !defined(AFX\_DLLVOLUME\_H\_\_7B232070\_8E04\_11D4\_B542\_0006290DDAA5\_\_INCLUDED\_)