NWTC Library – short overview of subroutines and functions

This documentation was developed for version 1.05.00 of the NWTC Library. Some changes may take place in later versions of the library. Documentation was updated as follows:

26-Jul-2012: A. Platt
24-Oct-2012: B. Jonkman
4-Dec-2012: A. Platt (v 1.05.02a)
7-Dec-2012: B. Jonkman (v1.06.00b)
12-Dec-2012: M. Buhl (v1.06.00c)
3-Feb 2013: B. Jonkman (v2.00.00c) [incomplete]

Each file is listed separately with its MODULE and contained subroutines and functions. Unless noted otherwise, the listed routines are subroutines.

SingPrec.f90 (DoublePrec.f90)

Declares kind for single- or double-precision floating-point variables.

MODULE Precision: Stores constants to specify the KIND of variables. This module only contains constants.

NWTC_Library.f90

Requires:

ModMesh.f90, ModMesh_Types.f90, NWTC_IO.f90, NWTC_Library.f90, and NWTC_Num.f90.

Requires one, but not both, of the following files:

DoubPrec.f90 or SingPrec.f90.

Your project must include one, and only one, of the following files:

SysIVF.f90, SysGnuLinux.f90, SysGunWin.f90, SysIFL.f90, SysMatlab.f90, or SysIVF_Labview.f90.

Compilation order for command-line compilation:

SingPrec.f90 (DoubPrec.f90)
SysIVF.f90 (or other Sys*.f90 file)
NWTC_IO.f90
NWTC_Num.f90
ModMesh_Types.f90
ModMesh.f90
NWTC_Library.f90

Invoking programs should call NWTC_Init() to initialize data important to the use of the library.

MODULE NWTC_Library

Name	Arguments	Description
NWTC_Init	ProgNameIn, ProgVerIn	Initialize *ProgName* and *ProgVer* if parameters have been passed. This routine then calls all required initialization routines. Write the version of the NWTC subroutine library that we are running

SysIVF.f90 (SysGnuLinux.f90, SysGnuWin.f90, SysIFL.f90, SysMatlab.f90, SysIVF_Labview.f90)
Contains routines with system-specific logic and references. It also contains standard (but not system-specific) routines that it uses.

Sys File	Intended Compiler/System
SysIVF.f90	Intel Visual Fortran for Windows compiler
SysIFL.f90	Intel Fortran for Linux compiler
SysGnuLinux.f90	GNU Fortran for Linux compiler
SysGnuWin.f90	GNU Fortran for Linux compiler
SysMatlab.f90	Intel Visual Fortran for Windows compiler with Matlab's mex functions
SysIVF_Labview.f90	Intel Visual Fortran for Windows compiler with references to IFPORT removed and no writing to the screen (output to a file named "Console.txt" instead)

MODULE SysSubs:

Name	Arguments	Description
FileSize	FileName, Size	Calls the routine FSTAT to obtain the size of the specify file or returns -1 on error.
FlushOut	Unit	Flushes the buffer on the specified *Unit*. It is especially useful when printing "running..." type messages.
Get_CWD	DirName, Status	Retrieves the path of the current working directory.
Is_NaN *(function)*	DblNum	Determines if a REAL(DbKi) variable holds a proper number.
OpenCon		Opens the console for standard output.
OpenUnfInpBEFile	Un, InFile, RecLen, Error	Opens a binary input file with data stored in Big Endian format (created on a UNIX machine). Data are stored in *RecLen*-byte records.
ProgExit	StatCode	Stops the program. If the compiler supports the EXIT routine, pass the program status to it. Otherwise, do a STOP
UsrAlarm		Generates an alarm to warn the user that something went wrong.
WrNR	Str	Writes out a string to the screen without following it with a new line.
WrOver	Str	Writes out a string that overwrites the previous line.
WriteScr	Str, Frm	Writes out a string to the screen. Break long messages into multiple lines.

NWTC_Num.f90

Contains numeric-type routines with non-system-specific logic and references.

It also contains global numeric-related variables.

MODULE NWTC_Num:

Name	Arguments	Description
AddOrSub2Pi	OldAngle, NewAngle	This routine is used to convert *NewAngle* to an angle within 2**Pi* of *OldAngle* by adding or subtracting 2**Pi* accordingly; it then sets *OldAngle* equal to *NewAngle*. This routine is useful for converting angles returned from a call to the ATAN2() FUNCTION into angles that may exceed the -Pi to Pi limit of ATAN2(). This routine assumes that the angle change between calls is not more than 2*Pi in absolute value. *OldAngle* should be SAVEd in the calling routine.
BSortReal	RealAry, NumPts	This routine sorts a list of real numbers. It uses the bubble sort algorithm, which is only suitable for short lists.
Cross_Product *(function)*	Vector1, Vector2	This function computes the cross product of two 3-element arrays: *Cross_Product = Vector1* X *Vector2* (resulting in a vector).
EqualRealNos *(function)*	ReNum1, ReNum2	This function compares 2 real numbers and determines if they are "almost" equal, *i.e.* within some relative tolerance.
GaussElim	AugMat, NumEq, x, ErrStat ErrMsg	This routine uses the Gauss-Jordan elimination method to solve a system of linear equations Ax=b for x; AugMat = [A b]. *The routine works if the pivots are nonzero and you don't want the reduced/eschelon form of the augmented matrix.*
GetSmllRotAngs *(function)*	DCMat, ErrStat	This subroutine computes the angles that make up the input direction cosine matrix, *DCMat.*
GL_Pts	IPt, NPts, Loc, Wt, ErrStat	Returns the non-dimensional (-1:+1) location of the given Gauss-Legendre Quadrature point and its weight. The values came from Carnahan, Brice; Luther, H.A.; Wilkes, James O. (1969) "Applied Numerical Methods."
IndexCharAry *(function)*	CVal, CAry	Returns an integer index such that *CAry*(*IndexCharAry*) = *CVal*. If no element in the array matches *CVal*, the value -1 is returned. The routine performs a binary search on the input array to determine if *CVal* is an element of the array; thus, *CAry* must be sorted and stored in increasing alphebetical (ASCII) order. The routine does not check that the array is sorted. The routine assumes that *CVal* is type CHARACTER and *CAry* is an array of CHARACTERS.
InterpBin *(function interface)*	XVal, XAry, YAry, ILo, AryLen	Returns a y-value that corresponds to an input x-value by interpolating into the arrays. It returns the first or last *YAry()* value if *XVal* is outside the limits of *XAry()*. **Note:** This is an interface for InterpBinComp and InterpBinReal and will call the appropriate one (depending if *YAry* is complex or real).

Name	Arguments	Description
InterpStp *(function interface)*	XVal, XAry, YAry, Ind, AryLen	Returns a y-value that corresponds to an input x-value by interpolating into the arrays. It uses the passed index as the starting point and does a stepwise interpolation from there. This is especially useful when the calling routines save the value from the last time this routine was called for a given case where *XVal* does not change much from call to call. When there is no correlation from one interpolation to another, InterpBin() may be a better choice. It returns the first or last *YAry()* value if *XVal* is outside the limits of *XAry()*. **Note:** This is an interface for InterpStpComp and InterpStpReal and will call the appropriate one (depending if *YAry* is complex or real).
IsSymmetric *(function)*	A	Returns a logical TRUE/FALSE value that indicates if the given (2-dimensional) matrix, A, is symmetric. If A is not square it returns FALSE.
LocateBin	XVal, XAry, Ind, AryLen	Finds the lower-bound index of an input x-value located in an array. On return, Ind has a value such that *XAry*(*Ind*) <= *XVal* < *XAry*(*Ind*+1), with the exceptions that *Ind* = 0 when *XVal* < *XAry*(1), and *Ind* = *AryLen* when *XAry*(*AryLen*) <= *XVal*. **Note:** If the index doesn't change much between calls, *LocateStp*() may be a better option.
LocateStp	XVal, XAry, Ind, AryLen	Finds the lower-bound index of an input x-value located in an array. On return, *Ind* has a value such that *XAry*(*Ind*) <= *XVal* < *XAry*(*Ind*+1), with the exceptions that *Ind* = 0 when *XVal* < *XAry*(1), and *Ind* = *AryLen* when *XAry*(*AryLen*) <= *XVal*. It uses the passed index as the starting point and does a stepwise search from there. This is especially useful when the calling routines save the value from the last time this routine was called for a given case where *XVal* does not change much from call to call. When there is no correlation from one interpolation to another, a binary search may be a better choice.
Mean *(function)*	Ary, AryLen	Function to calculate the mean value of a vector array.
MPi2Pi	Angle	Ensures that *Angle* lies between -*Pi* and *Pi*.
SetConstants		Computes some useful constants based upon *Pi* and IEEE arithmetic.
RombergInt	f, a, b, R, err, eps, ErrStat	Used to integrate a function *f* over the interval [*a*, *b*] (*f* is an external function). This routine is useful for sufficiently smooth (*e.g.*, analytic) integrands, integrated over intervals which contain no singularities, and where the endpoints are also nonsingular.
SmllRotTrans	RotationType, Theta1, Theta2, Theta3, TransMat, ErrTxt	This routine computes the 3x3 transformation matrix, *TransMat*, to a coordinate system *x* (with orthogonal axes x_1, x_2, x_3) resulting from three rotations (*Theta1*, *Theta2*, *Theta3*) about the orthogonal axes (X_1, X_2, X_3) of coordinate system *X*. All angles are assumed to be small, as such, the order of rotations does not matter and Euler angles do not need to be used. This routine is used to compute the transformation matrix (*TransMat*) between undeflected (*X*) and deflected (*x*) coordinate systems. *See the subroutine in the file NWTC_Num.f90 for more details.*

Name	Arguments	Description
SortUnion	Ary1, N1, Ary2, N2, Ary, N	Takes two sorted arrays and finds the sorted union of the two. **Note:** If the same value is found in both arrays, only one is kept. However, if either array as multiple occurrences of the same value, the largest multiple will be kept. Duplicates should be eliminated externally if this is not desirable
StdDevFn *(function)*	Ary, AryLen, Mean	Calculates the standard deviation of a population contained in *Ary*.

NWTC_IO.f90

Contains I/O-related variables and routines with non-system-specific logic.

MODULE NWTC_IO:

Name	Arguments	Description
AdjRealStr *(interface)*	NumStr	Removes leading spaces and trailing zeros from strings created by real numbers.
AllocAry *(interface)*	Ary, AryDim1, [AryDim2], [AryDim3], Descr, [ErrStat], [ErrMsg]	Allocates logical, character, integer, and real arrays. Values are passed for *AryDim2*, and *AryDim3* when 2 or 3 dimensional arrays are requested. **Note:** This interface will call the appropriate allocation subroutine depending on the type and dimensionality of the array requested. This interfaces to: – character array allocation subroutines (AllCAry1, AllCAry2, AllCAry3) – logical array creation subroutines (AllLAry1, AllLAry2, AllLAry3) – integer array allocation subroutines (AllIAry1, AllIAry2, AllIAry3) – real array allocation subroutines (AllRAry1, AllRAry2, AllRAry3)
AllocPAry *(interface)*	Ary, AryDim1, [AryDim2], [AryDim3], Descr, [ErrStat], [ErrMsg]	Allocates integer and real pointer arrays.
CheckArgs	InputFile, ErrStat	Checks for command-line arguments.
CheckIOS	IOS, Fil, Variable, VarType, [TrapErrors, ErrMsg]	Checks the I/O status and prints either an end-of-file or an invalid-input message, and then aborts the program.
Conv2UC	Str	Converts all the text in *Str* to upper case.
CountWords *(function)*	Line	Function that counts the number of "words" in a line of text. It uses spaces, tabs, commas, semicolons, single quotes, and double quotes ("whitespace") as word separators.
CurDate *(function)*		Function that a character string encoded with the date in the form dd-mmm-ccyy.
CurTime *(function)*		Function that returns a character string encoded with the time in the form "hh:mm:ss".
DispNVD *(interface)*	-- ProgDesc, Name/Ver	Displays the name of the program, its version, and its release date. **Note:** This interface will call the appropriate allocation subroutine depending on the type and number of arguments passed. This interfaces to: – DispNVD0 – no inputs. The global variables ProgName and ProgVer are used – DispNVD1 – Single input of type ProgDesc. – DispNVD2 – Two arguments of character type containing the name and version info
FindLine	Str, MaxLen, StrEnd	Finds one line of text with a maximum length of *MaxLen* from the *Str*. It tries to break the line at a blank.
GetNewUnit	UnIn [ErrStat], [ErrMsg]	Returns a unit number in the range [10, 99] that is not currently in use.
GetNVD *(function)*	ProgDesc	Returns a string with the program name, version, and date (converts data structure to single string)

Name	Arguments	Description
GetPath	GivenFil, PathName	Parses the path name from the name of the given file. It counts everything before (and including) the last "\" or "/".
GetRoot	GivenFil, RootName	Parses the root file name from the name of the given file. It counts everything after the last period as the extension.
GetTokens	Line, NumTok, Tokens, Error	Parses *Line* for *NumTok* "tokens" and return them in the *Tokens* array. This routine differs from GetWords() in that it uses only spaces as token separators.
GetWords	Line, Words, NumWords	Retrieves *NumWords* "words" from a *Line* of text.
NameOFile	InArg, OutExten, OutFile, ErrStat	Get the name of the input file from the *InArg*th command-line argument. Remove the extension if there is one, and append *OutExten* to the end.
NormStop		Performs a normal termination of the program.
Num2LStr *(function interface)*	Num	Converts a floating point number to a left-aligned string. It eliminates trailing zeroes and the decimal point on floating point numbers. Note: This is an interface to several the functions Int2LStr, R2LStr4, R2LStr8, and R2LStr16. It will call the appropriate one depending on the type of *Num*.
OpenBInpFile	Un, InFile, ErrStat, ErrMsg	Opens a binary input file.
OpenBOutFile	Un, OutFile, ErrStat, ErrMsg	Opens a binary output file.
OpenEcho	Un, OutFile, [ErrStat], [ErrMsg], [ProgVer]	Opens a formatted output file for the echo file, and writes a header line if the Program Description is an input to the routine..
OpenFInpFile	Un, InFile, [ErrStat], [ErrMsg]	Opens a formatted input file.
OpenFOutFile	Un, OutFile, [ErrStat], [ErrMsg]	Opens a formatted output file.
OpenFUnkFile	Un, OutFile, FailAbt, Failed, Exists, ErrStat	Opens a formatted output file and returns a flag (*Exists*)telling if it already existed.

Name	Arguments	Description
OpenUInBEFile	Un, InFile, RecLen, ErrStat	Opens an unformatted input file of *RecLen*-byte data records stored in Big Endian format.
OpenUInfile	Un, InFile, ErrStat	Opens an unformatted input file.
OpenUOutfile	Un, OutFile, ErrStat	Opens an unformatted output file.
PathIsRelative *(function)*	GivenFil	Determine if the given file name is absolute or relative. A path is considered an absolute path if it satisfies one of the following criteria: 1) It contains ":/" or ":\" 2) It starts with "/" or "\" All others are considered relative.
PremEOF	Fil, Variable, [TrapErrors, ErrMsg]	Write out an EOF message and aborts the program. *(because this same functionality is done in CheckIOS(), the PremEOF() routine shouldn't be necessary)*
ProgAbort	Message, TrapErrors	Outputs fatal error messages and stops the program.
ProgPause		Pauses the program and requires the user enter an <Enter> to resume execution.
ProgWarn	Message	Outputs non-fatal warning *Message* and returns to the calling routine.
ReadAry *(interface)*	UnIn, Fil, Ary, AryLen, AryName, AryDescr, ErrStat, [UnEc]	Reads in *AryLen* values into the array *Ary* from the next *AryLen* lines of the input file. Note: This is an interface to the subroutines ReadCAry, ReadIAry, ReadLAry, and ReadRAry. It will call the appropriate one depending on the type of *Ary*. ReadRAry can read values separated by white space from the same line of the input file as well.
ReadAryLines *(interface)*	UnIn, Fil, Ary, AryLen, AryName, AryDescr, ErrStat, [UnEc]	Reads in *AryLen* values into the array *Ary* from the next *AryLen* lines of the input file. Note: This is an interface to the subroutines ReadCAryLines, ReadDAryLines, and ReadRAry. It will call the appropriate one depending on the type of *Ary*.
ReadCom	UnIn, Fil, ComName, [ErrStat], [ErrMsg], [UnEc]	Reads a comment from the next line of the input file.
ReadFASTBin	UnIn, FASTdata, ErrLev, ErrMsg	Reads the contents of a FAST binary output file and stores it in FASTdata. The name of the data file is input through the FASTdata structure by the calling procedure.

Name	Arguments	Description
ReadNum	UnIn, Fil, Word, VarName, [ErrStat], [ErrMsg]	Reads a single word from a file and tests to see if it's a pure number (no true or false).
ReadOutputList	UnIn, Fil, CharAry, AryLenRead, AryName, AryDescr, ErrStat	Reads a *AryLen* values into a real array from the next *AryLen* lines of the input file.
ReadStr	UnIn, Fil, CharVar, VarName, VarDescr, [ErrStat], [ErrMsg], [UnEc]	Reads a string from the next line of the input file.
ReadVar *(interface)*	UnIn, Fil, Var, VarName, VarDescr, [ErrStat], [ErrMsg], [UnEc]	Reads in variable *Var* from the next line of the input file. *Var* can be of type CHARACTER, DOUBLE, INTEGER, LOGICAL, or REAL. Note: This is an interface to the subroutines *ReadCVar*, *ReadIVar*, *ReadLVar*, and *ReadR*Var*. It will call the appropriate one depending on the type of *Var*.
WaitTime	WaitSecs	Waits for *WaitSecs* before proceeding.
WrBinFAST	FileName, FileID, DescStr, ChanName, ChanUnit, TimeData, AllOutData, ErrStat, ErrMsg	This subroutine opens a binary file named *FileName*, and writes a the *AllOutData* matrix to a 16-bit packed binary file. A text *DescStr* is written to the file as well as the text in the *ChanName* and *ChanUnit* arrays. The file is closed at the end of this subroutine call (and on error). NOTE: Developers may wish to inquire if the file can be opened at the start of a simulation to ensure that it's available before running the simulation (i.e., don't run for a long time only to find out that the file cannot be opened for writing).
WrFileNR	Unit, Str	Writes out the string, *Str*, to the file connected to *Unit* without following it with a new line.
WrML	Str	Writes out the string, *Str*, in the middle of a line.
WrPr	Str	Writes out a prompt with text *Str* to the screen without following it with a new line, though a new line precedes it.
WrScr	Str	Writes out a string to the screen. Breaks long messages into multiple lines. Writes strings on new lines when it finds the NewLine substring