NAL-NL2.dll

Scott Brewer
Version 1.0.0.0 (updated documentation)
Monday, September 15th, 2020.

Background

NAL-NL2 is the second generation of prescription procedures from The National Acoustic Laboratories (NAL) for fitting wide dynamic range compression (WDRC) instruments. Like its predecessor (NAL-NL1), NAL-NL2 is a threshold-based procedure that prescribes gain-frequency responses for different input levels, or the compression ratios at different frequencies, in wide dynamic range compression hearing aids. This allows speech intelligibility to be maximised for any input level of speech above the compression threshold, while keeping the overall loudness of speech at or below normal overall loudness.

NOTES

Where a parameter is defined as an array of 9 elements, e.g. AC[9] then, unless otherwise stated it will be an array of the standard audiometric frequencies used to enter thresholds in NAL NL2. 0=250Hz, 1=500Hz, 2=1kHz, 3=1.5kHz, 4=2kHz, 5=3kHz, 6=4kHz, 7=6kHz, and 8=8kHz

A parameter defined as an array of 19 elements e.g. REIG[19], contains data at third-octave frequencies. 0=125Hz, 1=160Hz, 2=200Hz, 3=250Hz, 4=315Hz, 5=400Hz, 6=500Hz, 7=630Hz, 8=800Hz, 9=1000Hz, 10=1250Hz, 11=1600Hz, 12=2000Hz, 13=2500Hz, 14=3150Hz, 15=4000Hz, 16=5000Hz, 17=6300Hz 18=8000Hz.

The functions in the NAL-NL2.cpp section have a “Requires” section. This lists the number of the functions that need to be called before the function itself can be used, and the order in which to call them. This is to ensure internal variables are initialised prior to use or retrieve variables to be used in the function call.

File Documentation

NL2.h File Reference

Functions

void ExtractBinResource (int nResourceId, double DataPts[20], double* rows);
void ExtractBinResource2 (int nResourceId, double DataPts[400], double* rows, double* columns);
void reig (double outputHat[20], double HTL[20], double dbSPL, double AC[20], double ACother[], double AB[], int noOfAids);
void reig_calc (double outputHat[20], double HTL[20], double dbSPL, double ACother[], double AB[], int noOfAids, int compSpeed);
void RealEarInsertionGain_linear (double REIGlin[19], double H[9], double AC[9], double ACother[9], double AB[9], int noOfAids);
int RealEarAidedGain_NL2_Internal (double REAG[19], double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, double ACother[9], int noOfAids, bool useLevel, double level);
int round (double d);
int ageIndex (int d1, int d2);
int ageIndex2 (int d1, int d2);
int ageMonths (int d1, int d2);
int getDirMicOffset (int direction, int mic);
int meanMaxMin (double *mean, double *max, double *min, double AC[19]);
void calcUCT (bool calc[19], double CT[19], int WBCT, int bandWidth, int selection, int aidType, int mic, int direction);
void convertToNL2Freqs (double in[9], double out[9]);
void convertToTOF (double in[9], double out[19]);
void resr (double RESR[19], double H[19], double AB[19], int channels, int limiting);
void setBWC (int channels, double FC[]);
void sort (double list[], int size);
void sspl (double SSPL[19], double H[19], double AB[19], int channels, int limiting);
void getCurrentDate (int date[]);
void GetAB (double AB[9], double AC[9], double BC[9]);
double GetRECDaa (int m);
double GetRECDbaby (int m);
int GetRECDpred (double RECD[19], int dateOfBirth, int fittingDepth, bool foamTip);
double tansigLocal (double input);
double T (int noOfAids, double AC, double ACother, double L);
int daysInMonth (int month, int year);
double GetRECD_MoldFoam(int m);
double GetRECD_HA1_HA2(int m);

Function Documentation

NL2.cpp

This file contains the formula calculations.

void ExtractBinResource(int nResourceId, double DataPts[20], double* rows);

Extract single column binary data from DLL

Parameters:

nResourceId	Input	ID of the resource to extract
DataPts[20]	**Output**	Format [0] to [rows-1] :– data points
rows	**Output**	Number of rows in output array

void ExtractBinResource2(int nResourceId, double DataPts[400], double* rows, double* columns);

Extract multi-column binary data from DLL

Parameters:

nResourceId	Input	ID of the resource to extract
DataPts[20]	**Output**	Format [0] to [(rows*columns)-1] :– data points
rows	**Output**	Number of rows in output array
columns	**Output**	Number of columns in output array

void reig(double outputHat[20], double HTL[20], double dbSPL, double AC[20], double ACother[], double AB[], int noOfAids);

Calculate the real ear insertion gain, based on set internal options, for internal function use

Parameters:

outputHat[]	**Output**	Real Ear Insertion Gain – third octaves
HTL[]	Input	Hearing thresholds – third octaves
dbSPL	Input	Required input Level in dB
AC[]	Input	Hearing thresholds - third octaves
ACother[]	Input	Hearing thresholds for the other ear – third octaves
AB[]	Input	Air-Bone gap – third octaves
noOfAids	Input	Number of hearing aids – 0 monoaural, 1 binaural

void reig_calc(double outputHat[20], double HTL[20], double dbSPL, double ACother[], double AB[], int noOfAids, int compSpeed);

Calculate the base NAL-NL2 formula results for internal function use

Parameters:

outputHat[]	**Output**	Output: Real Ear Insertion Gain – third octaves
HTL[]	Input	Hearing thresholds – third octaves
dbSPL	Input	Required input Level in dB
ACother[]	Input	Hearing thresholds for the other ear – third octaves
AB[]	Input	Air-Bone gap – third octaves
noOfAids	Input	Number of hearing aids – 0 monoaural, 1 binaural
compSpeed	Input	Compression speed - 0 slow, 1, fast

void RealEarInsertionGain_linear(double REIGlin[19], double H[9], double AC[9], double ACother[9], double AB[9], int noOfAids);

Linear (no compression) part of the NAL-NL2 formula

Parameters:

REIGlin[]	**Output**	linear output of the NL2 formula (no compression) – third octaves
H[]	Input	Hearing thresholds – third octaves
AC[]	Input	Hearing thresholds– third octaves
ACother[]	Input	Hearing thresholds for the other ear – third octaves
AB[]	Input	Air-Bone gap – third octaves
noOfAids	Input	Number of hearing aids – 0 monoaural, 1 binaural

int RealEarAidedGain_NL2_Internal(double REAG[19], double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, double ACother[9], int noOfAids, bool useLevel, double level);

Real Ear Aided Gain targets for internal function use

Parameters:

REAG[]	**Output**	
AC[]	Input	Air Conduction Loss– third octaves
BC[]	Input	Bone Conduction thresholds – third octaves
L	Input	Input level in dB
Limiting	Input	Type of limiting
channels	Input	Number of channels
direction	Input	Direction of microphone
mic	Input	Location of microphone
ACother	Input	AC Loss for the other ear – third octaves

noOfAids	Input	Number of hearing aids – 0 monoaural, 1 binaural
useLevel		NOT USED
level		NOT USED

Returns:

0

int round(double d);

Round input to up to next integer

Parameters:

d	Input	Input value

Returns:

Rounded value

int ageIndex(int d1, int d2);

Calculate an age index based on the age in days

Parameters:

d1	Input	Start date (e.g. date of birth) - yyyymmdd
d2	Input	End date (e.g. current date) - yyyymmdd

Returns:

0 – 0 to 3 months
1 – 3 to 6 months
2 – 6 to 12 months
3 – 12 to 24 months
4 – 24 to 36 months
5 – greater than 36 months

int ageIndex2(int d1, int d2);

Calculate an age index based on the age in days

Parameters:

d1	Input	Start date (e.g. date of birth) - yyyymmdd
d2	Input	End date (e.g. current date) - yyyymmdd

Returns:

0 – 0 to 15 years
1 – greater than 15 years

int ageMonths(int d1, int d2);

Calculate age in months

Parameters:

d1	Input	Current date (yyyymmdd)
d2	Input	Date of birth (yyyymmdd)

Returns:

Age in months

int getDirMicOffset(int direction, int mic);

Get index value based on microphone location and direction

Parameters:

direction	Input	Direction of microphone – 0 = 0 degrees, 1 = 45 degrees
mic	Input	Location of microphone – 0 = undisturbed field, 1 = head surface

Returns:

Offset index

int meanMaxMin(double *mean, double *max, double *min, double AC[19]);

Calculate the average, maximum and minimum thresholds for the given hearing loss

Parameters:

mean	**Output**	Output: Average threshold
Max	**Output**	Output: Maximum threshold
min	**Output**	Output: Minimum threshold
AC []	Input	Hearing loss at third octaves

Returns:

0 (success).

void calcUCT (bool calc[19], double CT[19], int WBCT, int bandWidth, int selection, int aidType, int mic, int direction);

Calculate underlying compression thresholds

Parameters:

Calc[]	**Output**	Output: underlying compression thresholds at third octaves
CT[]	Input	Compression thresholds at third octaves
WBCT	Input	Wideband Compression Threshold
bandwidth	Input	0 = Broad band, 1 = Narrow band
selection	Input	Target type (0 = REIG, 1 = REAG, 2 = 2cc Coupler, 3 = Ear Simulator)
aidType	Input	Type of hearing aid
mic	Input	Location of microphone – 0 = undisturbed field, 1 = head surface
direction	Input	Direction of microphone – 0 = 0 degrees, 1 = 45 degrees

void convertToNL2Freqs(double in[9], double out[9]);

Convert from NAL-NL1 frequencies to NAL-NL2 frequencies (125Hz, 250Hz, 500Hz, 1000Hz, 2000Hz, 3150Hz, 4000Hz, 6000Hz, 8000Hz)

Parameters:

in[]	Input	(250Hz, 500Hz, 1000Hz, 1500Hz, 2000Hz, 3000Hz, 4000Hz, 6000Hz,

		8000Hz)
out[]	**Output**	(125Hz, 250Hz, 500Hz, 1000Hz, 2000Hz, 3150Hz, 4000Hz, 6000Hz, 8000Hz)

void convertToTOF(double in[9], double out[19]);

Convert from required NAL-NL2 frequencies (125Hz, 250Hz, 500Hz, 1000Hz, 2000Hz, 3150Hz, 4000Hz, 6000Hz, 8000Hz) to Third-Octave Frequencies (125Hz – 8kHz)

Parameters:

in[]	Input	(125Hz, 250Hz, 500Hz, 1000Hz, 2000Hz, 3150Hz, 4000Hz, 6000Hz, 8000Hz)
DataPts[20]	**Output**	Output: (125Hz, 160Hz, 200Hz, 250Hz, 315Hz, 400Hz, 500Hz, 630Hz, 800Hz, 1000Hz, 1250Hz, 1600Hz, 2000Hz, 2500Hz, 3150Hz, 4000Hz, 5000Hz, 6300Hz, 8000Hz)

void resr (double RESR[19], double H[19], double AB[19], int channels, int limiting);

Calculate the Real Ear Saturation Response

Parameters:

RESR[]	**Output**	Output: Real Ear Saturation Response at third octaves
H[]	Input	Hearing loss thresholds at third-octave frequencies
AB[]	Input	Air-Bone gap at third-octave frequencies
channels	Input	Number of channels
limiting	Input	Type of limiting used

void setBWC (int channels, double FC[]);

Calculate bandwidth correction factor for each third octave for the given number of channels

Parameters:

channels	Input	Number of channels
FC[]	Input	Crossover frequency of each channel

void sort (double list[], int size);

Sort array of doubles

Parameters:

list[]	**Input \ Output**	Input\Output: Array to be sorted
size	Input	Size of array

void sspl (double SSPL[19], double H[19], double AB[19], int channels, int limiting);

Calculate RESR and convert to SSPL (Saturated Sound Pressure Level) at third octaves

Parameters:

SSPL[]	**Output**	SSPL at third octaves

H[]	Input	Hearing Thresholds – third octaves
AB[]	Input	Air-bone gap – third octaves
channels	Input	Number of channels
limiting	Input	Type of limiting to be used

void getCurrentDate(int date[]);

Get the current system date

Parameters:

date[]	**Output**	Output for current date

Returns:

date[0] current day (0-31)
date[1] current month (0-12)
date[2] current year

void GetAB(double AB[9], double AC[9], double BC[9]);

Calculate the air-bone gap for the given AC and BC thresholds

Parameters:

AB[]	**Output**	Air-Bone gap – third octaves
AC[]	Input	Air Conduction thresholds – third octaves
BC[]	Input	Bone Conduction thresholds – third octaves

double GetRECDaa(int m);

Calculate Adult Average RECD at selected frequency

Parameters:

m	Input	Selected third octave frequency index (0 = 125Hz – 18 = 8000Hz)

Returns:

Adult average RECD at m

double GetRECDbaby(int m);

Calculate infant RECD at selected frequency

Parameters:

m	Input	Selected third octave frequency index (0 = 125Hz – 18 = 8000Hz)

Returns:

Infant RECD at m

int GetRECDpred(double RECD[19], int dateOfBirth, int fittingDepth, bool foamTip);

Get predicted RECD values for given age

Parameters:

RECD[]	**Output**	Output: RECD values
dateOfBirth	Input	Date Of Birth (yyyymmdd)

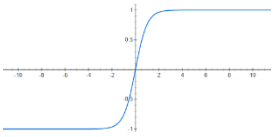
fittingDepth	Input	Depth of Fitting: 1 = deep, 2 = shallow, 0 - normal
foamTip	Input	Type of Tip: 1 = Foam, 0 = Mold

Returns:

0

double tansigLocal(double input);

Limit input value to be between –2 and 2.



Parameters:

input	Input	**double**

Returns:

double

double T(int noOfAids, double AC, double ACother, double L);

Adjustment for bilateral or unilateral fitting

Parameters:

noOfAids	Input	0 = Unilateral, 1 = Bilateral
AC	Input	Air Conduction threshold
Acother	Input	Air Conduction threshold of the other ear
L	Input	Input level in dB

Returns:

Correction factor

int daysInMonth (int month, int year);

Calculate the number of days in the specified month and year

Parameters:

month	Input	Month required (01 to 12)
year	Input	Year (yyyy)

Returns:

Number of days in specified month and year.

double GetRECD_MoldFoam(int m);

Get the RECD correction for selected tip (mold\foam) at specified frequency

Parameters:

m	Input	Selected frequency

Returns:

Correction factor

double GetRECD_HA1_HA2(int m);

Get the RECD correction for selected 2-cc coupler style (HA1\HA2) at specified frequency

Parameters:

m	Input	Selected frequency

Returns:

Correction factor

Function Documentation

Macro Definition Documentation

#define DllExport extern "C" __declspec(dllexport)

Export the function for external use

Function Documentation

NAL-NL2.cpp

This file contains API functions.

1 DllExport int WINAPI dllVersion (int *major, int *minor)

Returns the Major and minor parts of the dll Version number.

Parameters:

major	**Output**	major version number
minor	**Output**	minor version number

Returns:

0

Requires:

None

2 DllExport int WINAPI RealEarInsertionGain_NL2(double REIG[19], double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, double ACother[9], int noOfAids)

Calculate Real Ear Insertion Gain

Parameters:

REIG[]	**Output**	Real Ear Insertion Gain data at third-octaves
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
L	Input	level of broadband signal in dB
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral

Returns:

0

Requires:

23, 21, 34, 35, 36, 37, 38

3 DllExport int WINAPI RealEarAidedGain_NL2(double REAG[19], double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, double ACother[9], int noOfAids)

Calculate Real Ear Aided Gain

Parameters:

REAG	**Output**	Real Ear Aided Gain data at third-octaves
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
L	Input	level of broadband signal in dB
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral

Returns:

0

Requires:

23, 21, 34, 35, 36, 37, 38

double GainAt_NL2_Internal(int freqRequired, int targetType, double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, double ACother[9], int noOfAids)

Calculate {targetType} gain at one frequency for internal dll use only

Parameters:

freqRequired	Input	Index of Third Octave Frequency
targetType	Input	Gain type to use; 0 = REIG, 1 = REAG, 2 = 2cc Coupler, 3 = Ear Simulator
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
L	Input	level of broadband signal in dB
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral

Returns:

Gain at selected frequency for selected target type

Requires:

?????

int TccCouplerGain_NL2_Internal(double TccCG[19], double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, int target, int aidType, double ACother[9], int noOfAids, bool useLevel, double level, int tubing, int vent, int /*RECDmeasType*/, int lineType[19])

Calculate 2cc Coupler Gain for internal dll use only

Parameters:

TccCG[]	**Output**	2cc Coupler Gain data at third-octaves
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds

L	Input	level of broadband signal in dB
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
target	Input	Gain Targets to be used; 0 = REIG, 1 = REAG
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral
useLevel	Input	True == use braodband level
Level	Input	Broadband level in dB
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
RECDmeasType		NOT USED
lineType	**Output**	type of gain line to be used when drawing the gain curve; 0 = solid line, 1 = dotted line

Returns:

0

Requires:

?????

4 DllExport int WINAPI TccCouplerGain_NL2(double TccCG[19], double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, int target, int aidType, double ACother[9], int noOfAids, int tubing, int vent, int RECDmeasType, int lineType[19])

Calculate 2cc Coupler Gain

Parameters:

TccCG	**Output**	2cc Coupler Gain data at third-octaves
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
L	Input	level of broadband signal in dB
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
target	Input	Gain Targets to be used; 0 = REIG, 1 = REAG
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
RECDmeasType	Input	RECD measurement method; 0 = Predicted, 1 = Measured
lineType[]	**Output**	type of gain line to be used when drawing the gain curve; 0 = solid line, 1 = dotted line

Returns:

0

Requires:

23, 43 or 44, 17 or 18, 21, 34, 35, 36, 37, 38

5 DllExport int WINAPI EarSimulatorGain_NL2(double ESG[19], double AC[9], double BC[9], double L, int direction, int mic, int limiting, int channels, int target, int aidType, double ACother[9], int noOfAids, int /*tubing*/, int vent, int /*RECDmeasType*/, int lineType[19])

Calculate Ear Simulator Gain

Parameters:

ESG[]	**Output**	Ear Simulator Gain data at third octaves
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
L	Input	level of broadband signal in dB
Direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
target	Input	Gain Targets to be used; 0 = REIG, 1 = REAG
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral
Tubing	Input	NOT USED
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
RECDmeasType	Input	NOT USED
lineType	**Output**	type of gain line to be used when drawing the gain curve; 0 = solid line, 1 = dotted line

Returns:

0

Requires:

23, 43 or 44, 17 or 18, 21, 34, 35, 36, 37, 38

6 DllExport int WINAPI RealEarInputOutputCurve_NL2(double REIO[100], double REIOunl[100], double AC[9], double BC[9], int graphFreq, int startLevel, int finishLevel, int limiting, int channels, int direction, int mic, int target, double ACother[9], int noOfAids)

Calculate Real Ear Input\Output curves with and without limiting

Parameters:

REIO[]	**Output**	Real Ear Input\Output curve data for selected range
REIOunl[]	**Output**	Real Ear Input\Output curve data for selected range (no limiting)
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
graphFreq	Input	Index of Third Octave Frequency; 0 = 125Hz, 1 = 160Hz, …., 17 = 6300Hz, 18 = 8000Hz
startLevel	Input	starting input level (dB) for a narrowband test signal
finishLevel	Input	final input level (dB) for a narrowband test signal
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
Mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head

		Surface
target	Input	Gain Targets to be used; 0 = REIG, 1 = REAG
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral

Returns:

0

Requires:

23, 17 or 18, 21, 34, 35, 36, 37, 38

7 DllExport int WINAPI TccInputOutputCurve_NL2(double TccIO[100], double TccIOunl[100], double AC[9], double BC[9], int graphFreq, int startLevel, int finishLevel, int limiting, int channels, int direction, int mic, int target, int aidType, double ACother[9], int noOfAids, int /*tubing*/, int vent, int /*RECDmeasType*/, int lineType[100])

Calculate 2cc Coupler Input\Output curves with and without limiting

Parameters:

TccIO[]	**Output**	2cc Coupler Input\Output curve data for selected range
TccIOunl[]	**Output**	2cc Coupler Input\Output curve data for selected range (no limiting)
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
graphFreq	Input	Index of Third Octave Frequency; 0 = 125Hz, 1 = 160Hz, …., 17 = 6300Hz, 18 = 8000Hz
startLevel	Input	starting input level (dB) for a narrowband test signal
finishLevel	Input	final input level (dB) for a narrowband test signal
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
Mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
target	Input	Gain Targets to be used; 0 = REIG, 1 = REAG
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral
Tubing	Input	NOT USED
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
RECDmeasType	Input	NOT USED
lineType	**Output**	type of gain line to be used when drawing the gain curve; 0 = solid line, 1 = dotted line

Returns:

0

Requires:

23, 43 or 44, 17 or 18, 21, 34, 35, 36, 37, 38

8 DllExport int WINAPI EarSimulatorInputOutputCurve_NL2(double ESIO[100], double ESIOunl[100], double AC[9], double BC[9], int graphFreq, int startLevel, int finishLevel, int limiting, int channels, int direction, int mic, int target, int aidType, double ACother[9], int noOfAids, int /*tubing*/, int vent, int /*RECDmeasType*/, int lineType[100])

Calculate Ear Simulator Input\Output curves with and without limiting

Parameters:

ESIO[]	**Output**	Ear Simulator Input\Output curve data for selected range
ESIOunl[]	**Output**	Ear Simulator Input\Output curve data for selected range (no limiting)
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
graphFreq	Input	Index of Third Octave Frequency; 0 = 125Hz, 1 = 160Hz, …., 17 = 6300Hz, 18 = 8000Hz
startLevel	Input	starting input level (dB) for a narrowband test signal
finishLevel	Input	final input level (dB) for a narrowband test signal
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
Mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
target	Input	Gain Targets to be used; 0 = REIG, 1 = REAG
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral
Tubing	Input	NOT USED
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
RECDmeasType	Input	NOT USED
lineType	**Output**	type of gain line to be used when drawing the gain curve; 0 = solid line, 1 = dotted line

Returns:

0

Requires:

23, 43, or 44, 17 or 18, 21, 34, 35, 36, 37, 38

9 DllExport int WINAPI Speech_o_Gram_NL2(double Speech_rms[19], double Speech_max[19], double Speech_min[19], double Speech_thresh[19], double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, double ACother[9], int noOfAids)

Calculate RMS, max, min and threshold curves for Speech-o-gram

Parameters:

Speech_rms[]	**Output**	Speech RMS at third-octaves
Speech_max[]	**Output**	Speech Max at third-octaves
Speech_min[]	**Output**	Speech Min at third-octaves
Speech_thresh[]	**Output**	Speech Thresholds at third-octaves
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
L	Input	level of broadband signal in dB
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral

Returns:

0

Requires:

23, 15 or 16, 21, 34, 35, 36, 37, 38

10 DllExport int WINAPI AidedThreshold_NL2(double AT[19], double AC[9], double BC[9], double CT[19], int dbOption, double ACother[9], int noOfAids, int limiting, int channels, int direction, int mic)

Calculate Aided Thresholds

Parameters:

AT[]	**Output**	Aided Thresholds at third-octaves
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
CT[]	Input	Compression Threshold at third octave frequencies
dbOption	Input	Field Threshold Display Type; 0 – dB HL, 1 = dB SPL
ACother	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface

Returns:

0

Requires:

23, 15 or 16, 17 or 18, 21, 34, 35, 36, 37, 38

11 DllExport int WINAPI GetREDDindiv (double REDD[19], int defValues)

Get predicted or measured REDD data for third octave frequencies.

Parameters:

REDD[]	**Output**	REDD values
defValues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data

Returns:

0

Requires:

15 or 16 (if returning user entered values)

12 DllExport int WINAPI GetREDDindiv9 (double REDD[9], int defValues)

Get predicted or measured REDD data for standard NAL-NL2 frequencies.

Parameters:

REDD[]	**Output**	REDD values
defValues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data

Returns:

0

Requires:

15 or 16 (if returning user entered values)

13 DllExport int WINAPI GetREURindiv(double REUR[19], int defValues, int dateOfBirth, int

direction, int mic)

Get REUR data for third octave frequencies, adjusted for microphone location and direction.

Parameters:

REUR[]	**Output**	REUR values
defValues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface

Returns:

0

Requires:

17 or 18 (if returning user entered values)

14 DllExport int WINAPI GetREURindiv9(double REUR[9], int defValues, int dateOfBirth, int direction, int mic)

Get REUR data for standard NAL-NL2 frequencies, adjusted for microphone location and direction.

Parameters:

REUR[]	**Output**	REUR values
defValues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface

Returns:

0

Requires:

17 or 18 (if returning user entered values)

DllExport int WINAPI SetRECDindiv(double RECD[19])

This function has been deprecated

DllExport int WINAPI SetRECDindiv9(double RECD[9])

This function has been deprecated

15 DllExport int WINAPI SetREDDindiv(double REDD[19], int defValues)

Set the REDD data stored in the DLL to either predicted or client data.

Parameters:

REDD[]	Input	REDD values to be stored

defValues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data

Returns:

0

Requires:

45 or 46 (if setting predicted values)

16 DllExport int WINAPI SetREDDindiv9(double REDD[9], int defValues)

Set the REDD data stored in the DLL to either predicted or client data.

Parameters:

REDD[]	Input	REDD values to be stored
defValues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data

Returns:

0

Requires:

45 or 46 (if setting predicted values)

17 DllExport int WINAPI SetREURindiv(double REUR[19], int defValues, int dateOfBirth, int direction, int mic)

Set the REUR data stored in the DLL to either predicted or client data.

Parameters:

REUR[]	Input	REUR values to be stored
defVallues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface

Returns:

0

Requires:

None

18 DllExport int WINAPI SetREURindiv9(double REUR[9], int defValues, int dateOfBirth, int direction, int mic)

Set the REUR data stored in the DLL to either predicted or client data.

Parameters:

REUR[]	Input	REUR values to be stored
defValues	Input	use user supplied or program supplied data; 0 = predicted, 1 = use client data
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface

Returns:

0

Requires:

None

19 DllExport int WINAPI CrossOverFrequencies_NL2(double CFArray[], int channels, double AC[9], double /*BC*/[9], int FreqInCh[19])

Calculate the crossover frequencies (frequency between channels) for each channel

Parameters:

CFArray[]	**Output**	Array of (channels-1) Crossover Frequencies
channels	Input	number of channels (1 to 18)
AC[]	Input	Air Conduction Thresholds
BC[]	Input	NOT USED
FreqInCh[]	**Output**	Output: returns the channel a frequency is contained within

Returns:

0

Requires:

None

20 DllExport int WINAPI CenterFrequencies(int centerF[], double CFArray[], int channels)

Calculate the centre frequency of each channel

Parameters:

CFArray[]	**Output**	Array of (channels-1) Crossover Frequencies
channels	Input	number of channels (1 to 18)

Returns:

0

Requires:

19

21 DllExport int WINAPI CompressionThreshold_NL2(double CT[19], int bandWidth, int selection, int WBCT, int aidType, int direction, int mic, int calcCh[19])

Calculate Compression Thresholds for each third octave frequency

Parameters:

CT[]	**Output**	Compression Thresholds at third octave frequencies
bandWidth	Input	bandWidth of Noise, 0 = broadband, 1 = narrowband
selection	Input	Gain type to use; 0 = REIG, 1 = REAG, 2 = 2cc Coupler, 3 = Ear Simualtor
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
calcCh[]	Input	Recalculate value for frequency; 0 – No, 1 - Yes

Returns:

0

Requires:

23

22 DllExport int WINAPI CompressionRatio_NL2(double CR[], int channels, int centreFreq[], double AC[9], double BC[9], int direction, int mic, int limiting, double ACother[9], int noOfAids)

Calculate Compression ratios for each channel

Parameters:

CR[]	**Output**	Compression Ratios at third octave frequencies
centreFreq[]	Input	Array of Centre Frequencies for each channel
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral

Returns:

0

Requires:

23, 21, 34, 35, 36, 37, 38

23 DllExport int WINAPI setBWC(int channels, double crossOver[])

Set Bandwidth correction data for each third octave frequency

Parameters:

channels	Input	number of channels (1 to 18)
crossover[x]	Input	crossover frequencies, where x = max(1, channels – 1)

Returns:

0

Requires:

19 – unless you have calculated your own crossover frequencies

24 DllExport int WINAPI getMPO_NL2(double MPO[19], int type, double AC[9], double BC[9], int channels, int limiting)

Get MPO (Maximum Power Output) data for selected type and limiting based on Thresholds and channels

Parameters:

MPO[]	**Output**	Maximum Power Output for each third octave
type	Input	type of MPO required; 0 – RESR, 1 = SSPL
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
Channels	Input	number of channels (1 to 18)
Limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel

Returns:

0

Requires:

43 or 44 – For type == SSPL

25 DllExport double WINAPI GainAt_NL2(int freqRequired, int targetType, double AC[9], double BC[9], double L, int limiting, int channels, int direction, int mic, double ACother[9], int noOfAids, int bandWidth, int target, int aidType, int tubing, int vent, int RECDmeasType)

Calculate {targetType} gain at one frequency

Parameters:

freqRequired	Input	Index of Third Octave Frequency
targetType	Input	Gain type to use; 0 = REIG, 1 = REAG, 2 = 2cc Coupler, 3 = Ear Simulator
AC[]	Input	Air Conduction Thresholds
BC[]	Input	Bone Conduction Thresholds
L	Input	level of broadband signal in dB
limiting	Input	Type of limiting to be used; 0 = off, 1 = wideband, 2 = multichannel
channels	Input	number of channels (1 to 18)
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface
ACother[]	Input	AC Thresholds for the other ear
noOfAids	Input	Number of Hearing Aids fitted; 0 = Unilateral, 1 = Bilateral
bandWidth	Input	bandWidth of Noise, 0 = broadband, 1 = narrowband
target	Input	Gain Targets to be used; 0 = REIG, 1 = REAG
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
RECDmeasType	Input	RECD measurement method; 0 = Predicted, 1 = Measured

Returns:

Gain at selected frequency for selected target type

Requires:

23, 17 or 18, 21, 34, 35, 36, 37, 38

26 DllExport int WINAPI GetMLE(double MLE[19], int aidType, int direction, int mic)

Get Microphone Location Effects data for the selected microphone location and direction

Parameters:

MLE[]	**Output**	MLE at third octaves
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
direction	Input	direction of sound; 0 = 0 degrees, 1 = 45 degrees
mic	Input	Microphone Reference Position; 0 = undisturbed field, 1 = Head Surface

Returns:

0

Requires:

None

27 DllExport int WINAPI ReturnValues_NL2 (double MAF[19], double BWC[19], double ESCD[19])

Returns the MAF, BWC and ESCD data at third octave frequencies.

Parameters:

MAF[]	**Output**	Minimum Audible Field at third octaves
BWC[]	**Output**	Bandwidth Correction at third octaves
ESCD[]	**Output**	Ear Simulator to Coupler Difference data at third octaves

Returns:

0

Requires:

23

28 DllExport int WINAPI GetTubing_NL2 (double Tubing[19], int tubing)

Return tubing correction data for selected tube type at third octave frequencies

Parameters:

Tubing[]	**Output**	Tubing corrections at third octaves
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None

Returns:

0

Requires:

None

29 DllExport int WINAPI GetTubing9_NL2 (double Tubing[19], int tubing)

Return tubing correction data for selected tube type at standard NAL-NL2 frequencies

Parameters:

Tubing[]	**Output**	Tubing corrections at standard NAL-NL2 frequencies
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None

Returns:

0

Requires:

None

30 DllExport int WINAPI GetVentOut_NL2 (double Ventout[19], int vent)

Return vent correction data for selected vent type at third octave frequencies

Parameters:

Ventout[]	**Output**	Vent corrections at third octaves
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome

Returns:

0

Requires:

None

31 DllExport int WINAPI GetVentOut9_NL2 (double Ventout[9], int vent)

Return Vent correction data for selected vent type at standard NAL-NL2 frequencies

Parameters:

Ventout[]	**Output**	Vent corrections at standard NAL-NL2 frequencies
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome

Returns:

0

Requires:

None

32 DllExport double WINAPI Get_SI_NL2(int s, double REAG[19], double Limit[19])

Calculate SI (Saturation Index) at speech level s for Speech Thresholds

Parameters:

s	Input	Speech level; 0 = 55; 1 = 60; 2 = 65; 3 = 70; 4 = 75; 5 = 72; 6 = 69
REAG[]	Input	measured REAG at third octave frequencies
Limit[]	Input	RESR or SSPL limit at third octave frequencies

Returns:

SI value for selected speech level (s)

Requires:

None

33 DllExport double WINAPI Get_SII(int nCompSpeed, double Speech_thresh[19], int s, double REAG[19], double REAGp[19], double REAGm[19], double REUR[19])

Calculate SII (Speech Intelligibility Index) at speech level s for Speech Thresholds

Parameters:

nCompSpeed	Input	Compression Speed; 0 = Very Slow; 1 = Very Fast; 2 = Dual
Speech_thresh[]	Input	Hearing Threshold(AC) + $REDD_{adult\ avg}$
s	Input	Speech level; 0 = 55; 1 = 60; 2 = 65; 3 = 70; 4 = 75; 5 = 72; 6 = 69
REAG[]	Input	measured REAG
REAGp[]	Input	measured REAG at L + x where x = 15 for fast compression 7.5 for dual compression
REAGm[]	Input	measured REAG at L - x where x = 15 for fast compression 7.5 for dual compression
REUR[]	Input	REURindiv values

Returns:

SII value for selected speech level (s)

Requires:

None

34 DllExport void WINAPI SetAdultChild (int adultChild, int dateOfBirth)

Set client's age

Parameters:

adultChild	Input	0 = adult; 1 = child; 2 = calculate from date of birth

Requires:

None

35 DllExport void WINAPI SetExperience (int experience)

Set the clients hearing aid experience level

Parameters:

experience	Input	0 = experienced; 1 = new user

Requires:

None

36 DllExport void WINAPI SetCompSpeed (int compSpeed)

Set the compression speed

Parameters:

compSpeed	Input	0 = slow; 1 = fast; 2 = dual

Requires:

None

37 DllExport void WINAPI SetTonalLanguage (int tonal)

Set the Language type (Tonal\Non-Tonal)

Parameters:

tonal	Input	0 = non-tonal 1 = tonal

Requires:

None

38 DllExport void WINAPI SetGender (int gender)

Set clients' gender

Parameters:

gender	Input	0 = unknown; 1 = male; 2 = female

Requires:

None

39 DllExport int WINAPI GetRECDh_indiv_NL2 (double RECDh[19], int RECDmeasType, int dateOfBirth, int /*aidType*/, int tubing, int vent, int coupler, int fittingDepth)

Get RECDh data for third octave frequencies, adjusted for HA1 to HA2 conversion and, if predicted values required, tubing, fitting depth, and earpiece type.

Parameters:

RECDh[]	**Output**	RECD values
RECDmeasType	Input	RECD measurement method; 0 = Predicted, 1 = Measured
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
aidType	Input	NOT USED

tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None
coupler	Input	0 = HA1, 1 = HA2
fittingDepth	Input	0 = standard, 1 = deep, 2 = shallow

Returns:

0

Requires:

43 or 44 (if returning user entered values)

40 DllExport int WINAPI GetRECDh_indiv9_NL2 (double RECDh[9], int RECDmeasType, int dateOfBirth, int aidType, int tubing, int vent, int coupler, int fittingDepth)

Get RECDh data for standard NAL-NL2 frequencies, adjusted for HA1 to HA2 conversion and, if predicted values required, tubing, fitting depth, and earpiece type.

Parameters:

RECDh	**Output**	RECD values
RECDmeasType	Input	RECD measurement method; 0 = Predicted, 1 = Measured
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
coupler	Input	0 = HA1, 1 = HA2
fittingDepth	Input	0 = standard, 1 = deep, 2 = shallow

Returns:

0

Requires:

43 or 44 (if returning user entered values)

41 DllExport int WINAPI GetRECDt_indiv_NL2 (double RECDt[19], int RECDmeasType, int dateOfBirth, int /*aidType*/, int tubing, int vent, int earpiece, int coupler, int fittingDepth)

Get RECDt data for third octave frequencies, adjusted for HA1 to HA2 conversion and, if predicted values required, tubing, vent, fitting depth, and insert type.

Parameters:

RECDt	**Output**	RECD values
RECDmeasType	Input	RECD measurement method; 0 = Predicted, 1 = Measured
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
aidType	Input	NOT USED
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
earpiece	Input	0 = Foam Tip, 1 = Own Mold
coupler	Input	0 = HA1, 1 = HA2
fittingDepth	Input	0 = standard, 1 = deep, 2 = shallow

Returns:

0

Requires:

45 or 46 (if returning user entered values)

42 DllExport int WINAPI GetRECDt_indiv9_NL2 (double RECDt[9], int RECDmeasType, int dateOfBirth, int aidType, int tubing, int vent, int earpiece, int coupler, int fittingDepth)

Get RECDt data for third octave frequencies, adjusted for HA1 to HA2 conversion and, if predicted values required, tubing, vent, fitting depth, and insert type.

Parameters:

RECDt	**Output**	RECD values
RECDmeasType	Input	RECD measurement method; 0 = Predicted, 1 = Measured
dateOfBirth	Input	YYYYMMDD e.g. 20/4/1969 = 19690420
aidType	Input	Type of Hearing Aid fitted; 0 – CIC, 1 – ITC, 2 – ITE, 3 - BTE
tubing	Input	Type of tubing used; 0 – Libby 4, 1 = Libby 3, 2 = #13, 3 = Thin-tube, 4 = RITC, 5 = None
vent	Input	Type of Vent used, 0 = Tight, 1 = Occluded, 2 = Closed Dome, 3 = 1mm, 4 = 2mm, 5 = 3mm, 6 = Open Dome
earpiece	Input	0 = Foam Tip, 1 = Own Mold
coupler	Input	0 = HA1, 1 = HA2
fittingDepth	Input	0 = standard, 1 = deep, 2 = shallow

Returns:

0

Requires:

45 or 46 (if returning user entered values)

43 DllExport int WINAPI SetRECDh_indiv_NL2 (double RECDh[19])

Set the RECDh data stored in the DLL to either predicted or client data. To store predicted data, it must first be retrieved using one of the GetRECDh_indiv*() functions.

Parameters:

RECDh[]	Input	RECD values to be stored

Returns:

0

Requires:

39 or 40 (if setting to predicted values)

44 DllExport int WINAPI SetRECDh_indiv9_NL2 (double RECDh[9])

Set the RECDh data stored in the DLL to either predicted or client data. To store predicted data, it must first be retrieved using one of the GetRECDh_indiv*() functions.

Parameters:

RECDh[]	Input	RECD values to be stored

Returns:

0

Requires:

39 or 40 (if setting to predicted values)

45 DllExport int WINAPI SetRECDt_indiv_NL2 (double RECDt[19])

Set the RECDt data stored in the DLL to either predicted or client data. To store predicted data, it must first be retrieved using one of the GetRECDt_indiv*() functions.

Parameters:

RECDt[]	Input	RECD values to be stored

Returns:

0

Requires:

41 or 42 (if setting to predicted values)

46 DllExport int WINAPI SetRECDt_indiv9_NL2 (double RECDt[9])

Set the RECDt data stored in the DLL to either predicted or client data. To store predicted data, it must first be retrieved using one of the GetRECDt_indiv*() functions.

Parameters:

RECDt[]	Input	RECD values to be stored

Returns:

0

Requires:

41 or 42 (if setting to predicted values)