NOAH

Storing Audiological Measurements

REM/HIT Standard

DataFmtCodeStd 200
Version 1.1

HIMSA II K/S

The information in this document is subject to change according to the review policies established by HIMSA II K/S.

HIMSA II K/S MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OR SUITABILITY FOR A PARTICULAR PURPOSE. HIMSA shall not be liable for errors contained herein or for incidental consequential damages in connection with the supply of, performance of, or use of this material.

Preface

This is the second of a series of documents to be prepared by HIMSA A/S. Its purpose is to present and specify standard data formats for the storage and exchange of audiogram data within the framework of NOAH-compatible measurement and fitting software. The next document in the series will specify standards for Loudness Scaling.

The Hearing Instrument Manufacturers' Software Association A/S (HIMSA A/S) was founded at the beginning of 1993 by a group of hearing instrument manufacturers. It has been HIMSA A/S's mission to develop and market the NOAH software, and to make it a de facto standard for integrated hearing care software within the entire hearing industry.

The NOAH Fitting Framework is a software application that enables fitting and measurement software to share data on a common platform (NOAH). The fitting and measurement applications are provided by manufacturers who have signed a know-how licence agreement with HIMSA and thereby obtained the right to distribute the NOAH software, and to develop NOAH-compatible software applications, also referred to as modules.

Data format standards are a natural prerequisite for the ability to share data. Therefore, in co-operation with its licensees, HIMSA has prepared data format standards for Audiogram, REM/HIT, Loudness Scaling, Impedance, Otoacoustic Emission and Evoked Response Audiometry measurement types.

The documentation for these standards is available in so-called header files. These files are part of the 'software development kit', which HIMSA automatically distributes to its licensees.

Unfortunately, it is our experience that the header files are too easily misinterpreted. It has thus been decided that HIMSA must prepare a comprehensive standard document for each of the aforementioned measurement types. These documents will provide a detailed presentation of the data structure of the measurement formats as well as describe the application of the various types of, e.g. 'specific audiograms'.

The various data standards are subject to revision twice a year by a committee consisting of manufacturers of audiological measurement equipment (AEMs). Based on input prepared by HIMSA, it will be the responsibility of this committee to approve both new standard documents and updates of existing standards. The AEM Committee will meet on the Saturday following the end of the UHA Convention in Germany, i.e. in October, and on the Saturday following the end of the AAA Convention in the US, i.e. in April.

HIMSA also invites non-licensees to take part in the process of preparing and maintaining measurement data standards.

Figure 1 presents the principles by which NOAH administrates the measurement formats. Each block of stored data must be equipped with a header. This header uniquely identifies, e.g. the manufacturer who created the measurement, the type of measurement data contained in the data block and the measurement data format's revision number.

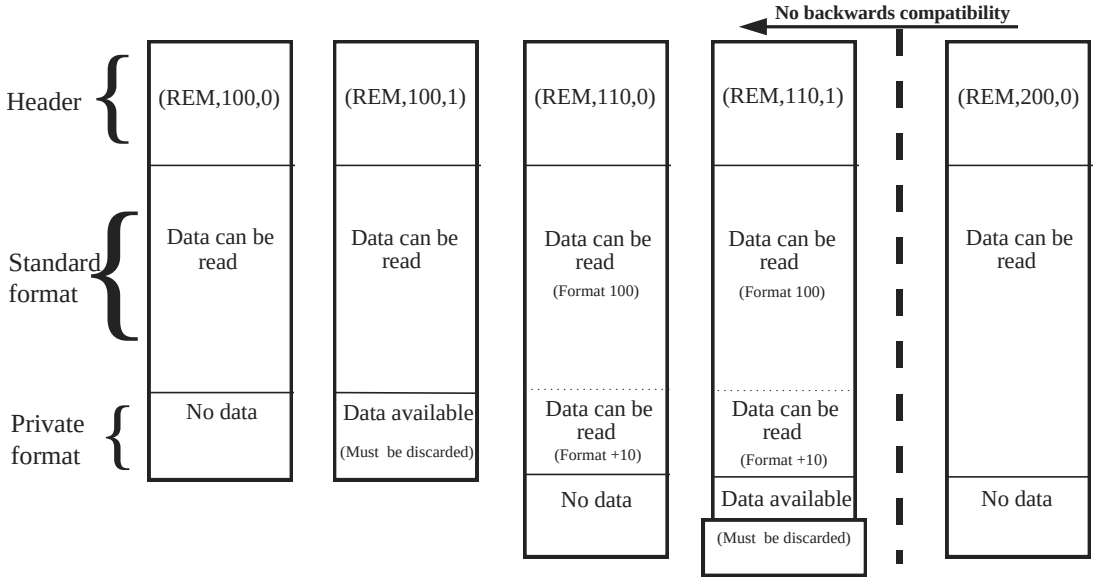

Figure 1: The handling of measurement data by NOAH

The basic revision number for a data format is 100. A data format with the revision number 110 is a direct extension of the basic 100 format. It is therefore possible for a revision 100 module to still read and understand a data block generated by a revision 110 module as it will simply discard the '+10' extension. A data format with the version number 200 would constitute a totally new revision thus making it impossible for revision 1xx modules to read revision 2xx data formats.

It is possible for a manufacturer to add non-standardised measurements to the public data block.

Document History

ver.	0.1	97-01-30	Document Template
ver.	0.6	97-03-07	First draft version
ver.	0.7	97-03-14	Second draft version. “Fact-boxes” added, dictionary completed.
ver.	0.8	97-03-17	Version for English language check.
ver.	0.9	97-03-17	Final Draft.
ver.	0.95	97-05-30	Corrections and comments from Approval Meeting inserted.
ver.	0.99	97-06-12	Edition for final Approval by Himsa
ver.	1.00	97-07-01	First version approved.

1 Introduction

1.1 A few words about programming with REMHIT.H

This document intends to explain the use of the NOAH ver 2.0 standard for storing Real Ear Measurements and Hearing Instrument Tests according to the REMHIT.H header file. This header file written in the programming language "C" defines the RemData (Rem stands for Real Ear Measurement) and HitData (Hit stands for Hearing Instrument Test) structures. The RemData structure for Real Ear Measurement contains the results of the following measurements:

REM	**Real Ear Measurement**
Target Gain	A calculated optimum frequency response curve for a Hearing Instrument.
REUR	Real Ear Unaided Response
REOR	Real Ear Occluded Response (Insertion Loss)
REIR	Real Ear Insertion Response
REAR	Real Ear Aided Response (In situ aided response)
IOMeas	Input / Output Measurement
HrmDistortion	Total Harmonic Distortion Measurement
Occlusion	Comparison of Un-occluded / Occluded ear response.
RECoupler	Real Ear to Coupler Difference Measurement

The HitData structure for Hearing Instrument Tests contains the following measurements:

HIT	**Hearing Instrument Test**
SPL90	Saturation Response
FullOnGain	Full On Gain
FreqResp	Frequency Response
Battery	Battery Measurement
HarmDistortion	Second and Third Harmonic Distortion Measurement
InterDistortion	Intermodulation Distortion Measurement
EquivInputNoise	Equivalent Input Noise Measurement
IOMeas	Input / Output Measurement
AttackRecover	Attack / Recover Curve measurement
InductionCoil	Induction Coil Measurement

The aim of this document is to explain the correct use of the RemData and HitData structures. This is done by reading the header file REMHIT.H “upside down” starting with the “outer” definition of RemData and HitData, continuing with the aforementioned measurements’ Measuring Conditions structure (rhMeasCond) attached to each measurement. This is followed by the type definition of the measurements, their Measuring Conditions and their associated curve points, ending with the definition of all “inner” types, all defined as integers (some unsigned, some enumerated types).

This document is written as the second part of documentation for software developers of the NOAH Framework Programming Interface. The first part explains auddef.h, the standard for storing audiograms. Data can be exchanged across this interface among the NOAH modules. In this way data can be shared among different Hearing Instrument- and Audiological Equipment-manufacturers. This document describes the REM and HIT measurement formats and can be read independently of other NOAH documentation. It is intended as a starting point for interested, prospective licensees.

1.2 Contents

1.3 References

[Framework] NOAH Framework ver. 0.85. System Architecture Specification. Pallas Informatik A/S.

[Mueller] H. Gustav Mueller, David B. Hawkins and Jerry L. Northern: " Probe Microphone Measurements. Hearing Aid Selection and Assessment". Singular Publishing Group, Inc. San Diego, CA. 1992. ISBN 1-879105-68-3.

[HOCA-4] Handbook of Clinical Audiology, edited by Jack Katz. Williams & Wilkins, 1994, 4. Edition.

2 The NOAH standard for REM / HIT

2.1 Data Structure

In order to describe the data structure as it is defined in REMHIT.H, an extended version of the language Abstract Syntax Notation No. 1 (ASN.1) is used [1]. This is done for the following reasons:

1. Explanation of the data structure in REMHIT.H starting with "the basic REM and HIT formats", RemData and HitData and the structures defining them. From these "outer", all-embracing types all constituent types are defined as we go by. (In effect, the header file 'upside down'). The definition in ASN.1 ends in the case of this header file by defining all the fundamental types as integers.

2. ASN.1 contains a few useful distinctions, used in this chapter to explain important places in REMHIT.H, where the order of variables matters, and where it does not. Note, that variables are called 'components' when in an outer structure:

SEQUENCE	Ordered collection of component types.
SEQUENCE OF	Ordered collection of variables of the component type.
SET	Unordered collection of component types, all distinct.
SET OF	Unordered collection of variables of the component type

[1] ASN.1 is defined by ISO and the International Telecommunication Union (ITU) (see ISO 8824) with a set of so-called Basic Encoding Rules which we shall NOT use here. Instead, a "Direct Encoding Rule" can be formulated: Data are encoded exactly as they are shown, down to the definition of the INTEGER as consisting of two byte, low-order transmitted first (placed at lower address).

2.1.1 The Integer type used in RemHit.H

minInt	-32768 #8000 hex	Lowest negative value represented in two byte using standard "2's complement" representation. According to [Framework] , this value is **illegal** for the integer types defined in REMHIT.H.
undefInt	-32767 #8001 hex	Used to indicate that the value is **undefined,** a value which is assigned to the constant undefInt. Ref. [Framework]
minParmInt	-32766 #8002 hex	Lowest negative value **legal** in parameters defined as integer types in REMHIT.H according to [Framework].
Unknown	0 #0000 hex	*In Parameters:* The parameter is **defined,** however to an **unknown** value**.** *In Curve points:* Use logic here! For the types TdB10, TPct100, TMa100 and Tmm100 the value 0 is of course **defined** and **valid,** however for the THertz type, the value means **undefined**.
NoParam	1 #0001 hex	*In Parameters:* The parameter is **defined** Not Used (channel, parameter…), see REMHIT.H for different explanations of the different types: ct_none, bt_none, bp_none and so on.
MaxInt	32767 #7FFF hex	Highest positive value. Ref. [Framework].

2.1.2 Definition of RemHit standard

NOTE: When adding a rule name, a single byte needs to be used for alignment. For example, if a field is defined to have 51 characters, where each character is 1 byte, then an extra byte needs to be added for alignment purposes. This is an empty byte, set aside to serve as a placeholder.

```
RemHit DEFINITIONS ::=

IMPORTS ALL FROM Noahdef -- noahdef.h


--  Real Ear Measurement
--
--  RemData:
--  Defines structure for storing a Real Ear Measurement
--
--  REUR (Unaided Response):
--  Input is the input level, output is uncompensated probe level.
--  Must be valid for use as an output curve as well as a gain curve, so if only output has
--   been measured, fill the input part with the stimulus level value.
--
--  REOR (Occluded Response):
--  Input is the input level, output is uncompensated probe level.
--  Actually like Aided Response (REAR) but the HI is off or detached during the
--   measurement.
--
--
```

-- REIR (Insertion Response):
-- Input is the input level, output is the probe level with or without REUR compensation.
--

```
-- Measurement = meas_InsertGain_U ->
-- The output values are dB SPL values without any REUR compensation.
--
-- Measurement = meas_InsertGain_C ->
-- The output values are dB SPL values minus the REUR's gain, i.e. with REUR
--  compensation.
--
-- Gain curves may be stored here. In this case the Input part must be set to zero and the gain
--  value must be stored in the Output part.
-- REAR (Aided Response):
--
-- Input is the input level, output is uncompensated probe level.


RemData ::= SEQUENCE {
    targets                 SET OF 3 TTargetCurve,
    rEUR                    TFreqMeas,
    rEOR                    TFreqMeas,
    rEIR                    SET OF 5 TFreqMeas,
      rEAR                  SET OF 5 TFreqMeas,
    iOMeas                  SET OF 5 TIOMeas,
    hrmDistortion           SET OF 3 TTHDDistMeas,
    occlusion               SET OF 3 TOcclMeas,       -- See TOcclMeas description
      reCoupler             TRECMeas                   -- See TRECMeas description
}
```

HitData :	Defines the structure for storing a Hearing Instrument Test

```
HitData ::=             SEQUENCE {
    spl90               SET OF 2 TFreqMeas,
    fullOnGain          SET OF 2 TFreqMeas,
    freqResp            SET OF 2 TFreqMeas,
    battery             SET OF 2 TBatMeas,
    harmDistortion      SET OF 2 TTHDDistMeas,
    interDistortion     SET OF 2 TTIMDistMeas,
    equivInputNoise     SET OF 2 EINMeas,
    ioMeas              SET OF 2 TIOMeas,
    attackRecover       SET OF 4 TARMeas,
    inductionCoil       SET OF 2 TFreqMeas
}
```

TARMeas:	Attack / Recovery Measurement. A complete curve set containing a full Attack/Release test.

```
TARMeas ::= SEQUENCE {
   measCond          TRHMeasCond,          -- Generic meas. conditions.
   levelStep         TdB10,                -- Level step size from the start
                                           -- level defined in MeasCond.
   attackCrv         TARCurve              -- The curve where the
                                           -- stimulus level increases.
   releaseCrv        TARCurve              -- The curve where the
                                           -- stimulus level decreases
}
```

EINMeas:	Equivalent Input Noise Measurement.
Input part:	Uncompensated level in the test chamber.
Output part:	Coupler output level compensated with the gain value.
	In a test box, the Hearing Instrument (HI) is connected to a 2CC coupler. A reference microphone is placed close to the HI's microphone. No signal is used. The noise floor in the test box is measured by the reference microphone and saved as input level, and the level delivered in the 2CC coupler by the HI is measured. The 2CC gain (measured earlier) is subtracted from the measured level and the result, saved as output, is called the Equivalent Input Noise. This noise is the noise floor plus the HI noise. The level should be at least at the input level, which is saved as an extra security.

```
EINMeas ::= SEQUENCE {
   measCond          TRHMeasCond,
   einCrv            SET OF 169 TFreqMeasPoint,
                     einRMS                TdB10   -- The calculated result
}
```

TARCurve:	A single curve for partial info of an Attack/Release measurement. The structure can contain either an attack or a release curve.

```
TARCurve ::= SEQUENCE {
   curve             SET OF 256 TARMeasPoint,
     result          INTEGER,              -- Result in ms: A / R
                     resolution            INTEGER,      -- The time resolution in
ms
     predelay        INTEGER               -- Time before level change
```

}

TARMeasPoint:	A measurement point for Attack/Release measurements. The time result (in milliseconds) is saved for the Attack and Release curve respectively. The predelay is saved as the number of milliseconds included in the curve before the level change.

```
TARMeasPoint ::= SEQUENCE {
    output                  TdB10
}


TARTime ::= INTEGER     -- Attack/Release time in milliseconds
```

TBatMeas:	A complete curve containing a Battery Current measurement.

```
TBatMeas ::= SEQUENCE {
    measCond                TRHMeasCond,
    batCrv                  SET OF 169 TBatMeasPoint
}
```

TBatMeasPoint:	A measurement point for Battery Current measurements.

```
TBatMeasPoint::= SEQUENCE {
    freq                    THertz,
    current                 TmA100
}


TmA100 ::= INTEGER      -- Current value in hundredths of a milliAmpere (unsigned)
```

TIOMeas:	A complete curve containing an Input / Output measurement.

```
TIOMeas ::= SEQUENCE {
    measCond                TRHMeasCond,
    ioCrv                   SET OF 61 TIOMeasPoint
}
```

TIOMeasPoint:	A measurement point for Input / Output measurements.
Note:	No frequency information in single points. This is common for all the Measurement Points and the value is stated in MeasCond.

```
TIOMeasPoint::= SEQUENCE {
   input                TdB10,
   output               TdB10
}
```

TTIMDistMeas:	A complete curve containing an Intermodulation Distortion measurement. The Intermodulation Distortion (IM) (difference-frequency) is the ratio of the power of the output signal at frequencies other than those delivered to the Hearing Instrument (HI) to the power of the signals that were applied to the HI. It includes the tones Freq2-Freq1 and 2*Freq1 - Freq2.

```
TTIMDistMeas ::= SEQUENCE {
   measCond             TRHMeasCond
   distCrv              SET OF 161 TTIMDistMeasPoint
}
```

TTIMDistMeasPoint:	A single measurement point for Total InterModulation Distortion Measurements.

```
TTIMDistMeasPoint ::= SEQUENCE {
   freq1                THertz          -- First stim freq
   freq2                THertz          -- Second stim freq
   input1               TdB10           -- Level of first stim
   input2               TdB10           -- Level of second stim
   output1              TdB10           -- Output at Freq1
   output2              TdB10           -- Output at Freq2
   outputDif1           TdB10           -- Output at Freq2-Freq1
   outputDif2           TdB10           -- Output at 2*Freq1-Freq2
   tIMPct               TPct100         -- The calculated result
}
```

TTHDDistMeas :	A complete curve containing a Harmonic Distortion measurement. The harmonic distortion is saved in this structure based on measurements of the distortion level at 2 * base frequency and 3* base frequency. It is calculated as the ratio of the total value of the harmonics and the base frequency. Ref. [HOCA]. Harmonic Distortion measured in a 2CC coupler has long been an element of ANSI standards of hearing aid performance. When measured in a Real Ear the distortion level rises considerably, although it is not fully understood why. Ref. [Mueller].

```
TTHDDistMeas ::= SEQUENCE {
    measCond                TRHMeasCond
    distCrv                 SET OF 161 TTHDDistMeasPoint
}
```

TTHDDistMeasPoint :	A single measurement point for Total Harmonic Distortion Measurements.

```
TTHDDistMeasPoint ::= SEQUENCE {
    freq                    THertz          -- Signal frequency
    input                   TdB10           -- Signal level
    output1Harm             TdB10           -- The output value at Freq
    output2Harm             TdB10           -- The output value at 2*Freq
    output3Harm             TdB10           -- The output value at 3*Freq
    thdPct                  TPct100         -- The calculated THD based
                                            --   on OutputXHarm
}


TPct100 ::= INTEGER
```

TTargetCurve :	A full target curve including the description. Use the signalLevel field to state the signal level if the target curve relates to a specific signal level. Use the ruleName to add a text containing a name and description of the target curve.

```
TTargetCurve ::= SEQUENCE {
    manufCode           TManufCode,
    devTypeCode         TDevTypeCode,
    fittingRule         TFittingRule,
    hiType              THIType,
    ventDiam            Tmm10,                  -- The diameter of vent canal
    ventLen             Tmm10,                  -- The length of the vent canal
    resGain             TdB10,                  -- The reserve gain included
                                                -- in the target curve
    couplerType         TCouplerType,
    signalLevel         TdB10,
    target              SET OF 24 TTargetPoint, -- The target curve
    ruleName            SET OF 51 CHARACTER STRING
}
```

TTargetPoint :	A single element/point of the target curve.

```
TTargetPoint ::= SEQUENCE {
    targetFreq          THertz,
    targetGain          TdB10
}
```

```
Tmm10 ::= INTEGER   -- Length measured in mm x 10 or tenths of a mm.
```

THIType :	The type of hearing instrument being tested.

```
THIType ::= ENUMERATED {
    hit_ITE             (1),
      hit_BTE,          (2),
      hit_ITC,          (3),
      hit_MITC,         (4),
      hit_Body,         (5),
      hit_User1,        (6),
      hit_User2,        (7),
      hit_User3,        (8),
      hit_User4,        (9),
      hit_User5,        (10),
      hit_Undefined     (undefInt)
}
```

TFittingRule :	The fitting rule used for calculation of a target curve

```
TFittingRule ::= ENUMERATED {
    fr_POGO             (1),     -- McCandless and Lyregaard 1983.
      fr_POGOII,        (2),     -- Schwartz, Lyregaard and Lundh, 1988.
      fr_NAL,           (3),     -- Original NAL by Byrne and Tonnison, 1976
      fr_NALProf,       (4),     -- Byrne, Parkinson and Newall, 1990, 1991.
      fr_Berger,        (5),     --Original Berger by Berger, Hagberg and Rane 1977.
      fr_HalfGain,      (6),     -- Lybargers Half-Gain rule as described by Brooks, 1973.
      fr_ThirdGain,     (7),     -- A simple rule for mild hearing losses.
      fr_DSL,           (8),     -- Desired Sensation Level. Seewald, 1992.
      fr_LIBBY,         (9),     -- Modified POGO by Libby, 1986.
      fr_Byrne,         (10),   -- Byrne and Tonnison, 1976.
      fr_CoxMSU,        (11),    -- Memphis State University, Cox, 1988.
      fr_User1          (100), -- (Suggested for Berger BTE 1988)
      fr_User2,         (101), -- (Suggested for Berger ITE 1988)
      fr_User3,         (102),
      fr_User4,         (103),
      fr_User5,         (104),
      fr_User6,         (105),
      fr_User7,         (106),
      fr_User8,         (107),
      fr_User9,         (108),
      fr_User10,        (109),
      fr_Undefined      (undefInt)
}
```

TRECMeas:	Real Ear to Coupler Difference measurement.
RECDCrv	This curve is to be read as gain (output - input), where the result gives the difference between a measurement in a coupler and a measurement in the Client's ear. As both parts of this measurement are performed with the same measurement conditions, the curve can be used in a manner with the input part holding the coupler SPL output curve, and the output part holding the Real Ear SPL output curve. This will yield the correct gain value.

```
TRECMeas ::= SEQUENCE {
   measCond                TRHMeasCond,
   recdCrv SET OF 169      TFreqMeasPoint
}
```

TOcclMeas:	Occlusion Effects, comparison between the response of the unoccluded ear with the response of the occluded ear.
OpenEarCrv:	The response of the unoccluded ear, no compensation in input or output.
OccEarCrv:	The response of the occluded ear, no compensation in input or output. The steps for measurement of the occlusion effect, ref. [Mueller]: 1. Disable the loudspeaker of system. 2. Place the tube in the open ear canal 25-30 mm from the tragal notch. 3. Have the patient vocalise, e.g. vowels such as /ee/ 4. Let the patient find, e.g. 80 dB SPL by using a small sound level meter . 5. When the level is right, conduct the open-ear measurement. 6. Place HI or earmold and repeat the above steps with HI turned off. 7. The difference between the two measurements constitutes the occlusion effect.

```
TOcclMeas ::= SEQUENCE {
   measCond                TRHMeasCond,
   openEarCrv              SET OF 169 TFreqMeasPoint,
   occlEarCrv              SET OF 169 TFreqMeasPoint
}
```

TFreqMeas:	Generic two-channel frequency response type measurement. Refer to the two structures: RemData and HitData in remhit.h.

```
TFreqMeas ::= SEQUENCE {
   measCond                TRHMeasCond,
   freqCrv                 SET OF 169 TFreqMeasPoint
}
```

TFreqMeasPoint:	A curve point that contains a frequency plus input and output levels as measured at the stored frequency. The type is referenced in the structures TFreqMeas, TOcclMeas, TRECMeas and EINMeas.

```
TFreqMeasPoint ::= SEQUENCE {
   freq                 THertz,          -- The frequency at which
                                         -- input/output was recorded
   input                TdB10,           -- Input value
   output               TdB10            -- Output value
}

END
```

2.1.3 Reading and writing curve points

REMHIT.H defines the following curves:

Structured type	**Curve**	
Name of outer structured type	*Curve Identifier*	*Type of curve points*
TBatMeas	batCrv	SET OF 169 TBatMeasPoint
TIOMeas	ioCrv	SET OF 61 TIOMeasPoint
TTIMDistMeas	distCrv	SET OF 161 TTIMDiustMeasPoint
TTHDDistMeas	distCrv	SET OF 161 TTHDDistMeasPoint
TRECMeas	recdCrv	SET OF 169 TFreqMeasPoint
TOcclMeas	openEarCrv occlEarCrv	SET OF 169 TFreqMeasPoint SET OF 169 TFreqMeasPoint
TFreqMeas	freqCrv	SET OF 169 TFreqMeasPoint

The reading of curve points in a RemHit measurement from NOAH ver 2.0 is per definition done in the following way:

The freq (or freq1) is read first *(see exception below).* The curve points might be ordered, but since they are defined as a set, they also might be *unordered* with respect to frequency. Read the curve points while checking that Freq (freq1) belongs to the interval [20..20 000].

Exception: The points that constitute the Input/Output measurement are measured for fixed frequency. A selection of curves is made for various frequencies and the result is a 3-dimensional relation: The output measurement varies as a function of both input level and frequency. Instead of reading freq1 first, the input level must be read first. This procedure must be repeated for each curve, i.e. for each frequency. The frequency can be found in rhMeasCond.

Curve points are read until the namedValue endCurve occurs:

-- Do not overlook this end of curve marker !!!

endCurve <curve point>::=

```
                        -- The curve point can be any of
                        -- the types listed in the table above
{
   undefInt,            -- freq1 = undefInt defines the endCurve
   undefInt,            -- or any other value
   undefInt,            -- or any other value
   undefInt,            -- or any other value
   undefInt             -- or any other value
   …
}
```

After endCurve, Curve points with freq1 = 0 or freq1 = undefInt (-32 767) are discarded. Curve points with such unreasonable frequency should be discarded at any time during the reading.

For the input / output exception read:
After endCurve, Curve points with input = 0 or input = undefInt are discarded.

When writing curve points, you have to make a choice: You can -

a) place the curvepoints in the order you prefer (for example the order in which they were measured) or you can

b) place them sorted with freq1 in ascending order

In both cases you will end with an endCurve marker and fill the rest of the array with endCurve markers (undefInts). This filling is not mandatory but is considerate to fellow programmers.

Note 1: In either case, valid codepoints should placed together. "Holes" in curves are not allowed.

Note 2: In order to retain compatibility with the existing RemHit modules, curves should start in curvepoint [0] with a valid frequency.

For the input / output exception read:
When writing curve points, you can -
a) Place the curvepoints in the order you prefer (for example the order in which they were measured) or you can
b) Place them sorted with inputs in ascending order

In both cases you will end with an endCurve marker and fill the rest of the array with endCurve markers (undefInts). Note 1 and 2 (above) applies.

2.1.4 RHMeasCond (Rem / Hit MeasCond)

REM HIT Measurement Conditions. The measurement conditions are placed as the first component in the following structured types: TOcclMeas, TRECMeas, TTHDDistMeas, TIOMeas, TBatMeas, TARMeas and EINMeas.

Set the signalFreq to undefInt in measurements where the curve points contain different frequencies.

Note the boolean value useRECoupler. It indicates whether the measurement is done using a coupler instead of a Real Ear. This method can be valuable for example if the patient is a child. Use the value TRUE, if the measurement is done using Real Ear to Coupler Difference. In this case the structure TRecMeas will contain the Real Ear to Coupler difference.

-- Information about each recorded curve: [17 x 2 x 2 = 68 bytes]

MeasuringConditions DEFINITIONS ::=

BEGIN

```
TRHMeasCond ::= SEQUENCE {
    manufCode           TManufCode,
    devTypeCode         TDevTypeCode,
    signalType          TSignalType,
    signalOutput        TSignalOutput,
    signalLevel         TdB10,
    signalFreq          THertz,
    battType            TBattType,
    battPill            TBattPill,
    battVoltage         TBattVoltage,
    battImp             TBattImp,
    useRECoupler        TBOOL,          -- type imported from Noahdef
    measMode            TMeasMode,
    measurement         Tmeasurement
}
```

2.1.5 Defined values

DefinedValues DEFINITIONS ::=

BEGIN

TSignalType:	The signal type used during the measurement.

```
TSignalType ::= ENUMERATED   {
    st_Tone             (1),
    st_Warble           (2),
    st_NarrNoise        (3),
    st_TwoTone          (4),
    st_WhiteNoise       (5),
    st_PinkNoise        (6),
    st_SpeechNoise      (7),
    st_Patient          (8),
    st_User1            (9),
    st_User2            (10),
    st_User3            (11),
    st_NU               (undefInt)
}
```

TSignalOutput:	The channel/media used to present the signal/stimulus.

```
TSignalOutput ::= ENUMERATED   {
    so_InternalBox          (1),
    so_ExternalBox          (2),
    so_FF                   (3),
    so_InternalBoxCoil      (4),
    so_ExternalBoxCoil      (5),
    so_FFCoil               (6),
    so_AC                   (7),
    so_User1                (8),
    so_User2                (9),
    so_User3                (10),
    so_NU                   (undefInt)
}
```

TCouplerType :	The device in which the probe microphone is placed; In other words; this is where the output is read.

```
TCouplerType ::= ENUMERATED   {
    ct_None             (1),
    ct_RealEar          (2),
    ct_711              (3),
    ct_2cc              (4),
    ct_FreibKK          (5),          -- Freiburg Konischer Kuppler
    ct_FreibKKK         (6),          -- Freiburger Konischer Kinder
                                      -- Kuppler
    ct_User1            (7),
    ct_User2            (8),
    ct_User3            (9),
    ct_NU               (undefInt)
}
```

TBattType :	The battery type used.

```
TBattType ::= ENUMERATED {
    bt_None             (1),
    bt_Mercury          (2),
    bt_ZincAir          (3),
    bt_OtherType        (4),
    bt_User1            (5),
    bt_User2            (6),
    bt_User3            (7),
    bt_NU               (undefInt)
}
```

```
TBattVoltage ::= TV1000     -- The voltage of the battery in milliVolts
TBattImp ::= TOhm1000       -- The impedance of the battery in milliOhms
```

```
TV1000 ::= Word             -- ASN.1 does not contain a "WORD" built-in type.
TOhm1000 ::= Word           -- For the "Direct Encoding Rules" is formulated:
                            -- The INTEGER is encoded 2-byte 2's
                            -- complement as in "C"
                            -- The Word is encoded 2-byte binary as in "C"
```

TBattPill :	The type of battery pill.

```
TBattPill ::= ENUMERATED {
    bp_None             (1),
    bp_Bat312           (2),
    bp_Bat13            (3),
    bp_Bat230           (4),          -- often referred to as size 10
    bp_Bat675           (5),
    bp_User1            (6),          -- suggested for size 5/5A
    bp_User2            (7),
    bp_User3            (8),
    bp_NU               (undefInt)
}
```

TMeasMode:	The measurement mode was used during the measurement

```
TMeasMode ::= ENUMERATED   {
    mm_Sweep            (1),
    mm_FFT              (2),
    mm_TimeMeas         (3),
    mm_Battery          (4),
    mm_User1            (5),
    mm_User2            (6),
    mm_User3            (7),
    mm_NU               (undefInt)
}
```

TMeasurement:	Identification of the measurement; mainly for use with data export and storage of not yet publicly defined measurements at the end of the public buffer.

```
Tmeasurement ::= ENUMERATED {
    meas_AudHand          (1),      -- Audiometry
    meas_TargHand         (2),      -- Target Curve
    meas_UnAided          (3),      -- Unaided Response
    meas_OcclResp         (4),      -- Occluded Response
    meas_InsertGain_C     (5),      -- Insertion Response
                                    --(Compensated for REUR)
    meas_AidedResp        (6),      -- Aided Response
    meas_InputOutput      (7),      -- Input Output
    meas_HarmDist         (8),      -- Harmonic Distortion
    meas_OcclEff          (9),      -- Occlusion Effect
    meas_RECD             (10),     -- Real Ear to Coupler Difference
    meas_SPL90            (11),     -- XSPL-90, i.e. OSPL-90 for IEC and
                                    -- SSPL-90 for ANSI
    meas_FOG              (12),     -- Full On Gain
    meas_FreqResp         (13),     -- Frequency Response
    meas_BattCurr         (14),     -- Battery Current
    meas_InterDist        (15),     -- Intermodulation Distortion
      meas_EquivNoise     (16),     -- Equivalent Input Noise
      meas_AttackRec      (17),     -- Attack/Recovery
      meas_IndCoil        (18),     -- Induction Coil
      meas_User1          (19),     -- User specific #1
      meas_User2          (20),     -- User specific #2
      meas_User3          (21),     -- User specific #3
      meas_InsertGain_U   (50),     -- Insertion Response
                                    -- (Uncompensated for REUR)
    meas_NU               (undefInt)
}
```

END – of defined values

2.2 Reading and writing RemHit Measurements

In the previous chapter, the RemData and HitData structures were explained. This chapter will give some hints to the actual reading and writing of a Rem- or HitData structure as defined in the NOAH standard version 2.0.

The basic principle is that a whole structure has to be saved although perhaps only one measurement has actually been performed. It can be one or it can be several different measurements, the result has to be saved in a complete structure. Unfortunately, this means that only a small fraction of the Rem- or HitData structure is filled by usable data. The NOAH database caters for this by compressing data before adding it to its database / expanding it before supplying the data to an external software module. The price paid in other words is slowed down communication, the gain is a uniform structure of data.

2.2.1 Reading the RemHit Measurements

The NOAH ver. 2.0 specification attaches a comprehensive measurement condition structure to each recorded curve called Measuring Conditions (Type definition TMeasCond).

In order to find the measurements that contain useful data when reading a Rem- or Hit Data structure, your program should read the Measuring Conditions attached to each measurement.

In this chapter a namedValue[2] called rhInitialCond is introduced. Most of the measurement conditions will be equal to this namedValue: rhInitialCond. Subsequent chapters describe the minimum changes in rhInitialCond that make RHMeasCond valid for each of the measurements that constitute a complete Rem- or Hit measurement.

Note 1: If the Rem Hit Measuring Conditions for a measurement are completely identical to rhInitialCond, this means that the associated measurement is empty.

Note 2: The definitions for Integer values written in the beginning of this chapter apply. However, the value zero can be found in empty measurements where the correct value should have been undefInt.

2.2.2 Writing the RemHit Measurements

When writing a Rem or Hit Measurement, use the following method:

1) Initialise all the measurements in the structure by setting all Rem Hit Measuring Conditions to the initial conditions rhInitialCond (see below). The codepoints should be initialised with endCurve. Refer to paragraph Reading and writing curve points.

2) Insert the appropriate values in the actual Rem Hit Measuring Conditions for the measurements that you want to save. Start with the minimum settings shown in the two subsequent chapters and modify according to the measuring conditions that were actually applied when recording the measurement.

The curvepoints are then inserted. Their insertion follows the directions mentioned in paragraph Reading and writing curve points.

[2] ASN.1 defines namedValues as structures of an indicated type with a defined content.

REM HIT Initial Measurement Conditions: The namedValue rhIinitialCond		
Data Type	*Field*	*Value*
TManufCode (noahdef.h)	ManufCode	undefInt (-32767)
TDevTypeCode (noahdef.h)	DevTypeCode	undefInt (-32767)
TSignalType	SignalType	st_NU = undefInt
TsignalOutput	SignalOutput	so_NU = undefInt
TdB10 (noahdef.h)	SignalLevel	undefInt (-32767)
THertz	SignalFreq	undefInt (-32767)
TbattType	BattType	undefInt (-32767)
TbattPill	BattPill	undefInt (-32767)
TbattVoltage	BattVoltage	undefInt (-32767)
TbattImp	BattImp	undefInt (-32767)
TBOOL	UseRECoupler	undefInt (-32767)
TmeasMode	MeasMode	undefInt (-32767)
Tmeasurement	Measurement	undefInt (-32767)

2.2.3 Minimum settings for a Real Ear Measurement (REM)

Target Gain:		
Exception: This measurement does not use rhMeasCond. Instead the TTargetCurve structure contains a number of components from rhMeasCond plus a few extra components necessary for a complete description of the measurement.		
Component of TtargetCurve	*Value*	*Explanation*
manufCode	choose the correct mfc_<Manufacturer code>	Mandatory Field. Use the correct #defined value as defined in noahdef.h. The value identifies the manufacturer of the measuring equipment.
devTypeCode	An Integer value in the range [1..maxInt] should be defined by the manufacturer.	Mandatory field. Defined individually by NOAH modules. Ref. [Framework].
fittingRule	choose the correct fr_<Fitting Rule>	Mandatory field. Enumerated values are defined in remhit.h.
hiType	choose the correct hi_<HI Type>	For some Fitting Rules the type of Hearing Instrument is not important. In these cases this field can be undefInt. Enumerated values are defined in remhit.h.
couplerType	choose the correct ct_<Coupler Type>	Mandatory field. Enumerated values are defined in remhit.h.
signalLevel	An Integer value in the range [0..1200]	For some Fitting Rules the signal Level is not important. In these cases this field can be undefInt. Otherwise measured in dB x 10 or centiBel.

REUR: Real Ear Unaided Response ***REOR: Real Ear Occluded Response*** ***REIR: Real Ear Insertion Response*** ***REAR: Real Ear Aided Response***		***IOMeas: Input / Output Measurement*** ***HrmDist: Harmonic Distortion*** ***Occlusion: Occluded ear response*** ***RECoupler: Real Ear to Coupler Difference***
Field in MeasCond	*Value*	*Explanation*
manufCode	Choose the correct mfc_<Manufacturer code>	Mandatory Field. Use the correct #defined value as defined in noahdef.h. The value identifies the manufacturer of the measuring equipment.
devTypeCode	An Integer value in the range [1..maxInt] should be defined by the manufacturer.	Mandatory field. Defined individually by NOAH modules. Ref. [Framework].
signalType	Choose the correct st_<Signal Type>.	Mandatory Field. Enumerated values are defined in remhit.h.
signalOutput	Choose the correct so_<Signal Type>	Mandatory Field. Enumerated values are defined in remhit.h.
signalLevel	An Integer value in the range [0..1400].	Mandatory Field. Measured in dB x 10 or centiBel. *Exception:* I/O measurement. Set this field to undefInt, since the input/output levels will be stored in the curve-points in this case.
signalFreq	An Integer value in the range [20..20 000].	Mandatory Field. *Exception:* Set this field to undefInt in case of multifrequency measurements. Otherwise state the frequency here.
useRECoupler	TRUE / FALSE.	Mandatory Field. TRUE if the measurement is done using Real Ear to Coupler Difference. In this case the structure TRecMeas will contain the Real Ear to Coupler difference.
measurement	Choose the correct meas_<Measurement>	Mandatory Field. Identifies the measurement.

2.2.4 Minimum settings for a Hearing Instrument Test (HIT)

SPL90:	***Saturation Response***	***HarmDistortion:***	***Harmonic Distortion***
FullOnGain:	***Full On Gain***	***InterDistortion:***	***Intermodulation Distortion***
FreqResp:	***Frequency Response***	***AttackRecover:***	***Attack / Recover Curve Measurement***
Attack/ Recover:	***Attack / Recover Curve***	***ICoil:***	***Induction Coil***

Field in MeasCond	*Value*	*Explanation*
manufCode	Choose the correct mfc_<Manufacturer code>	Mandatory Field. Use the correct #defined value as defined in noahdef.h. The value identifies the manufacturer of the measuring equipment.
devTypeCode	An Integer value in the range [1..maxInt] should be defined by the manufacturer.	Mandatory field. Defined individually by NOAH modules. Ref. [Framework].
signalType	Choose the correct st_<Signal Type>	Mandatory Field. Enumerated values are defined in remhit.h.
signalOutput	Choose the correct so_<Signal Type>	Mandatory Field. Enumerated values are defined in remhit.h.
signalLevel	An Integer value in the range [0..1400]	Mandatory Field. Measured in dB x 10 or centiBel. *Exception:* I/O measurement. Set this field to undefInt, since the input/output levels will be stored in the curve-points in this case.
signalFreq	An Integer value in the range [20..20 000]	Mandatory Field. *Exception:* Set this field to undefInt in case of multifrequency measurements, otherwise state the frequency here.
useRECoupler	TRUE / FALSE	Mandatory Field. TRUE if the measurement is done using Real Ear to Coupler Difference. In this case the structure TRecMeas will contain the Real Ear to Coupler difference.
measurement	Choose the correct meas_<Measurement>	Mandatory Field. Identifies the measurement.

Battery: Battery Measurement		
Field in MeasCond	*Value*	*Explanation*
manufCode	Choose the correct mfc_<Manufacturer code>	Mandatory Field. Use the correct #defined value as defined in noahdef.h. The value identifies the manufacturer of the measuring equipment.
devTypeCode	An Integer value in the range [1..maxInt] should be defined by the manufacturer.	Mandatory field. Defined individually by NOAH modules. Ref. [Framework].
signalType	Choose the correct st_<Signal Type>	Mandatory Field. Enumerated values are defined in remhit.h.
signalOutput	Choose the correct so_<Signal Type>	Mandatory Field. Enumerated values are defined in remhit.h.
signalLevel	An Integer value in the range [0..1400]	Mandatory Field. Measured in dB x 10 or centiBel. *Exception:* I/O measurement. Set this field to undefint, since the input/output levels will be stored in the curve-points in this case.
battType	Choose the correct bt_<Battery Type>	Mandatory Field. Mercury / Zinc-Air / Other Type as appropriate. Enumerated values are defined in remhit.h.
battPill	Choose the correct bp_<Battery Pill>	Mandatory Field. “312”, “13”, “230” or “675” as appropriate. Enumerated values are defined in remhit.h.
battVoltage	An unsigned Integer in the range [0.. 65535].	Mandatory Field. Measured in mV. It can be interesting, e.g. to measure the output level if the battery voltage is, e.g. 200 mV below the nominal level.
battImp	An unsigned Integer in the range [0.. 65535].	Mandatory Field. Measured in milliOhms. The measuring equipment simulates the battery impedance.

EquivInputNoise: Equivalent Input Noise Measurement		
All fields as for the SPL90 measurement except the two signalLevel and signalFreq fields.		
Field in MeasCond	*Value*	*Explanation*
signalLevel signalFreq	Values from the Gain measurement.	The stimulus level and frequency describes the situation when measuring the Gain. The gain is subtracted from the Output Noise Level in order to obtain the equivalent Input Noise Level.

IOMeas: Input / Output Measurement		
All fields as for the SPL90 measurement except the two signalLevel and signalFreq fields.		
Field in MeasCond	*Value*	*Explanation*
signalFreq	An Integer value in the range [20..20 000]	The I/O Measurement is performed at a fixed frequency.
signalLevel	Not applicable, use undefInt.	The signal levels are saved in the codepoints.

Appendix A: Vocabulary and Abbreviations

A

ASN.1	Abstract Syntax Notation No. 1. ITU and OSI defined language for specification of protocol message content.
Aided Response	Aided Response or Real Ear Aided Response (REAR) is the Sound Pressure Level (SPL) as a function of frequency, at a specified measurement point in the ear canal for a specified soundfield with the hearing aid in place and turned on. This can be expressed either in SPL or as gain in decibels relative to the stimulus level.
AttackCrv	A component of the structure TARMeas of type TARCurve. The component contains the result of an Attack Measurement.
AttackRecover	This test examines the response time of a Hearing Instrument for a change in sound pressure level. In general, narrow-band signals are used as stimulus. Component of the outer type HitData.

B

BatCrv	A Battery Curve consisting of 169 TBatMeasPoints, each consisting of a (frequency, current) measurement.
Battery	Battery is a component of the structure Hearing Instrument Test (HitData), where two battery measurements are saved. Each battery Measurement consists for Measurement Conditions and a Battery Curve.
BattImp	Battery Impedance (BattImp) in Ohm x 1000 (milliOhms). Component of type TBattImp in the REMHIT Measurement Conditions (RHMeasCond).
BattPill	Battery Pill (BattPill):"312", "13", "230" or "675". Component of type TBattPill in the REMHIT Measurement Conditions (RHMeasCond).
BattType	Battery Type (BattType): Mercury or Zinc-Air. Component of type TBattType in the REMHIT Measurement Conditions (RHMeasCond).
BattVoltage	Battery Voltage (BattVoltage) in Volt x 1000 (milliVolt). Component of type TBattVoltage in the REMHIT Measurement Conditions (RHMeasCond).
Bp_Bat13 *See* TBattPill	The Battery Pill Type (BP) is defined as type "13".
Bp_Bat230 *See* TBattPill	The Battery Pill Type (BP) is defined as type "230".

Bp_Bat312 *See* TBattPill	The Battery Pill Type (BP) is defined as type "312".
Bp_Bat675 *See* TBattPill	The Battery Pill Type (BP) is defined as type "675".
Bp_None *See* TBattPill	The Battery Pill Type (BP) is **defined** however, to an **unknown** value.
Bp_NU *See* TBattPill	The Battery Pill is undefined. This value is assigned to the constant undefInt.
Bp_User[1..3] *See* TBattPill	The Battery Pill is User Defined Battery Pill No. [1..3]. Not NOAH defined.
Bt_Mercury *See* TBattType	The Battery Type (BT) is Mercury.
Bt_None *See* TBattType	The Battery Type (BT) is **defined** however, to an **unknown** value.
Bt_NU *See* TBattType	The Battery Type is undefined. This value is assigned to the constant undefInt.
Bt_OtherType *See* TBattType	The Battery Type is different from Mercury or Zinc-air. Reserved for future NOAH definition.
Bt_User3 *See* TBattType	The Battery Type is defined as User Battery Type No. [1..3]. Not NOAH defined.
Bt_ZincAir *See* TBattType	The Battery Type is Zinc-Air.

C

components	Used in ASN.1 for the fields in a structured type (a "C" structure). The components are given Identifiers, i.e. a field name, in "C" referred to as the member.
CouplerType	The device in which the probe microphone is placed. In other words: this is where the output is read. >The type TCouplertype is defined Enumerated, the values are named ct_<coupler type>.
Ct_2cc *See* TCouplerType	The Coupler Type (CT) is a 2cc coupler.
Ct_711 *See* TCouplerType	The Coupler Type (CT) is an IEC 711 coupler.
Ct_FreibKK *See* TCouplerType	The Coupler Type (CT) is a Freiburger Konischer Kuppler.

Ct_FreibKKK *See* TCouplerType	The Coupler Type (CT) is a Freiburger Konischer *Kinder* Kuppler.
ct_None *See* TCouplerType	The Coupler Type (CT) is defined however, to an unknown value.
ct_NU *See* TCouplerType	The coupler type is undefined. This value is assigned to the constant undefInt.
ct_RealEar *See* TCouplerType	The Coupler Type (CT) is a real ear. The measurement is carried out in the client's ear.
ct_User[1..3] *See* TCouplerType	The Coupler Type (CT) is User Defined Coupler Type No. [1..3]. Not NOAH defined.
Current	Hearing Instrument Battery Current as simulated in a REMHIT test equipment, measured in mA x 100 (Hundredths of milli-Ampere). Component of the TbatMeasPoint structure of type TmA100. Defined as unsigned integer.
curve	A component of the Attack Release Curve structured type (TARCurve). The curve can be Attack or it can be Release. One curve consists of 256 points of type TARMeasPoint.

D

data structure	REMHIT.H describes the data structure for interchange of data with the NOAH ver. 2.0 database.
DevTypeCode	Defined as Integer in Noahdef.h. Identifies a particular device or instrument type to a NOAH module. Defined individually by NOAH modules. Ref. [Framework].
DistCrv	DistortionCurve. Curve of InterModulation or Harmonic Distortion curve points: SET OF TTIMDistMeasPoint or TTHDDistMeasPoint.

E

EINCrv	Equivalent Input Noise Curve points of the type SET OF 169 TFreqMeasPoint.
EINMeas	Equivalent Input Noise Measurement: 1. In a test box, the Hearing Instrument (HI) is connected to a 2CC coupler. 2. A reference microphone is placed close to the HI microphone. No signal is used. 3. The noise floor in the test box is measured by the reference microphone and saved as input level, and the level delivered in the 2CC coupler by the HI is measured. 4. The 2CC gain (measured earlier) is subtracted from the measured level and the result, saved as output, is the

Equivalent Input Noise.

5. This noise is the noise floor plus the HI noise. The level should be at least at the input level, which is saved as an extra security.

EINRMS — The curve points of the einCrv output curve are squared, summed and divided with the number of defined curve points. The square root of the result is saved as EINRMS (Root Mean Square).

endCurve — The set of curve points in a Rem- or HitData structure is not necessarily filled with data. It is recommended to save an endCurve marker after the curve points with actual data. The unused curve points can be endCurve or null-filled. See Reading and writing curve points.

EquivInputNoise — Equivalent Input Noise Measurement. A measurement in the Hearing Instrument Test (HitData) structure. Defined as SET OF 2 EINMeas.

F

FittingRule — When selecting gain and frequency response for a HI the question often is: “How much gain does the HI wearer desire for everyday listening"? The idea is that the **use gain** tends to be some percentage of the degree of hearing loss. Samuel Lybarger proposed the first fitting rule in the 1940’s. He proposed that gain should be equal to one half of the hearing loss. Ref. [Mueller]

fr_Berger (5) *See* TFittingRule — FittingRule = Berger. The original version of the Berger rule is Berger, Hagberg and Rane, 1977. The rule is based on gain, frequency response and Saturation Response. Requires only pure-tone thresholds. Ref. [Mueller].

fr_Byrne (10) *See* TFittingRule — FittingRule = Byrne and Tonnison, 1976. First version of the National Acoustic Laboratories (NAL) method, Australia. Ref. [Mueller].

fr_CoxMSU (11) *See* TFittingRule — Fitting Rule = Memphis State University, Cox 1988. Ref. [Mueller].

fr_DSL (8) *See* TFittingRule — FittingRule = Desired Sensation Level. The most recent paper on this method is Seewald, 1992. Ref. [Mueller]. The idea is that the long-term spectrum of speech is amplified in order to reach a DSL. Ref. [Mueller].

fr_HalfGain (6) *See* TFittingRule — FittingRule = Lybargers Half-gain rule as described by Brooks, 1973. Ref. [Mueller].

fr_LIBBY (9) *See* TFittingRule — FittingRule = Libby, 1986. A modification of the POGO procedure that varies the amount of REIG depending on the degree of hearing loss. A mild hearing loss leads to a preferred REIG equal to one third of their hearing loss. Ref. [Mueller].

fr_NAL (3) *See* TFittingRule	FittingRule = Byrne and Tonnison, 1976 National Acoustics Laboratories (NAL) of Australia.Ref. [Mueller].
fr_NALProf (4) *See* TFittingRule	FittingRule = Byrne, Parkinson and Newall, 1990, 1991. Modified NAL formula for persons with servere sensorineural hearing losses. Ref. [Mueller].
fr_POGO (1) *See* TFittingRule	FittingRule = McCandless and Lyregaard, 1983: Prescription of Gain and Output Procedure (POGO). POGO is a simple procedure based on the half-gain rule and REIG, but with a low-frequency reduction. Ref. [Mueller].
fr_POGOII (2) *See* TFittingRule	FittingRule = Schwartz, Lyregaard and Lundh, 1988. A modified POGO procedure. Ref. [Mueller].
fr_ThirdGain (7) *See* TFittingRule	FittingRule = One third of REIG. A simple rule for mild hearing losses. Ref. [Mueller].
fr_Undefined (-32767) *See* TFittingRule	FittingRule = undefined. The value is assigned to the constant undefInt (-32767).
fr_User[1..10] ([100..109]) *See* TFittingRule	The Fitting Rule is User Defined Fitting Rule No. [1..10]. Not NOAH defined.
Freq	In the structured type TfreqMeasPoint, TTHDDistMeasPoint and TBatMeasPoint: The frequency at which the measurement was done.
Freq1	First stimulation frequency. A component of the structured type TTIMDistMeasPoint of type THertz.
Freq2	Second stimulation frequency. A component of the structured type TTIMDistMeasPoint of type THertz.
FreqCrv	A component of the TfreqMeas structure, a generic, two-channel frequency response type.
FreqResp	The Hearing Instrument Frequency Response is a frequency / Level curve representing the amplification of the HI.
FullOnGain	Full On Gain. Component of the outer structure HitData of the type SET OF 2 TFreqMeas. It represents a calculated level in a 2CC coupler for servere hearing losses.

H

HarmDistortion	Harmonic Distortion. Component of the type SET OF 3 TTHDDistMeas in the outer structure RemData. Also SET OF 2 TTHDDistMeas in the HitData. The component represents the Total Harmonic Distortion Measurement.
hit_Body	Hearing Instrument Type = “Body Worn Hearing Instrument”.

See THIType

hit_BTE
See THIType
Hearing Instrument Type = “Behind the Ear”.

hit_ITC
See THIType
Hearing Instrument Type = “In the Canal”.

hit_ITE
See THIType
Hearing Instrument Type = “In the Ear”.

hit_MITC
See THIType
Hearing Instrument Type = “Mini In The Canal”.

hit_Undefined
See THIType
The Hearing Instrument Type = undefined. The value is assigned to the constant undefInt (-32767).

hit_User[1..5]
See THIType
The Hearing Instrument Test (hit) is User Defined Hearing Instrument Test No.[1..5]. Not NOAH defined.

HitData
Defines structure for storing a Hearing Instrument Test.

HIType
The type of the hearing instrument being tested.

I

InductionCoil
Induction Coil. Component of the outer structure HitData. Audio Induction Coil Loop is one of the oldest forms of assistive listening technology in use today. These systems provide large area access to Hearing Instruments equipped with telecoil.
Ref. [HOCA-4].

Input
In the structured types TFreqMeasPoint, TTHDDistMeasPoint and TIOMeasPoint the Signal Input Level. Type: TdB10 (dB x 10 or CentiBel).

Input1
In the structured type TTIMDistMeasPoint, the Signal No. 1 Input Level. Type: TdB10 (dB x 10 or CentiBel).

Input2
In the structured type TTIMDistMeasPoint, the Signal No. 2 Input Level. Type: TdB10 (dB x 10 or CentiBel).

Insertion Response
See Real Ear Insertion Response (REIR).

InterDistortion
Intermodulation Distortion Measurement. A component of the outer structure Hitdata of the type SET OF 2 TTIMDistMeas.

IOCrv
Input or Output curve of type SET OF 61 TIOMeasPoint. A

component of the structured type TIOMeas .

IOMeas — Input / Output measurement (ioMeas) component of the outer type RemData (Real Ear Measurement). Type SET OF 5 TIOMeas.

L

LevelStep — Input Level step size from the start level defined in MeasCond. The input used is a sine wave. At Attack is used Level Increase, at Release is used Level Decrease. The Level Step has to be performed at a zero crossing.
LevelStep is a component of type TdB10 (dB x 10 or centiBel) in the structured type TARMeas (Attack Release Measurement).

M

ManufCode — A Component in the TRHMeasCond and TTargetCurve structured types.

maxInt — Highest positive value for the Integer Type = 32767 (#7FFF hex)

meas_AidedResp — The Measurement is Aided Response (REAR).
See Tmeasurement

meas_AttackRec — The Measurement is Attack/Recovery.
See Tmeasurement

meas_AudHand — The Measurement is Audiometry.
See Tmeasurement

meas_BattCurr — The Measurement is Battery Current Measurement.
See Tmeasurement

meas_EquivNoise — The Measurement is Equivalent Input Noise.
See Tmeasurement

meas_FOG — The Measurement is Full On Gain (FOG).
See Tmeasurement

meas_FreqResp — The Measurement is Frequency Response.
See Tmeasurement

meas_HarmDist — The Measurement is Harmonic Distortion.
See Tmeasurement

meas_IndCoil — The Measurement is Induction Coil.
See Tmeasurement

meas_InputOutput *See* Tmeasurement	The Measurement is Input Output.
meas_InsertGain_C *See* Tmeasurement	Insertion Gain measurement with compensation for the REUR curve.
meas_InsertGain_U *See* Tmeasurement	Insertion Gain measurement without compensation for the REUR curve.
meas_InterDist *See* Tmeasurement	The Measurement is Intermodulation Distortion.
meas_NU *See* Tmeasurement	The Measurement is **undefined**. This value is assigned to the constant undefInt.
meas_OcclEff *See* Tmeasurement	The Measurement is Occlusion Effect.
meas_OcclResp *See* Tmeasurement	The Measurement is Occluded Response.
meas_RECD *See* Tmeasurement	The Measurement is Real Ear to Coupler Difference (RECD).
meas_SPL90 *See* Tmeasurement	The Measurement is xSPL-90 i.e. OSPL-90 for IEC and SSPL-90 for ANSI.
meas_TargHand *See* Tmeasurement	The Measurement is Target curve.
meas_UnAided *See* Tmeasurement	The Measurement is Unaided Response (REUR).
meas_User[1..3] *See* Tmeasurement	The Measurement (Meas) is User Defined Measurement No.[1..3]. Not NOAH defined.
measCond	Measuring Conditions. Component of type TRHMeasCond found in the structured types TFreqMeas, TOcclMeas, TRECMeas, TTHDDistMeas, TTIMDistMeas, TIOMeas, TBatMeas, TARMeas and EINMeas.
MeasMode	Measurement Mode (MeasMode) used during Measurement. A Component of the REM HIT Measurement Conditions (RHMeasCond) of type TMeasMode.
Measurement	Identification of the measurement, mainly for use with data export and storage of not yet publicly defined measurements at the end of the public buffer. Measurement is a Component of the REM HIT Measurement Conditions (RHMeasCond) of type TMeasurement.
Measuring Conditions	See MeasCond.
Minimum Settings	The recommended minimum of Measurement Conditions that must be saved with a measurement in order to make it valuable when retrieved at a later stage. See chapters 2.2.3 and 2.2.4.

minInt	Integers are stored using 2's complement in a two-byte store. This means that minInt = -32768 or #8000 hex. See chapter 2.1.1.
mm_Battery *See* TMeasMode	The Measurement Mode is Battery Current.
mm_FFT *See* TMeasMode	The Measurement Mode is Fast Fourier Transform.
mm_NU *See* TMeasMode	The Measurement Mode is **undefined**. This value is assigned to the constant undefInt.
mm_Sweep *See* TMeasMode	The Measurement Mode is Frequency Sweep.
mm_TimeMeas *See* TMeasMode	The Measurement Mode is Time Domain.
mm_User[1..3] *See* TMeasMode	The Measurement Mode (MM) is User Defined Measurement Mode No.[1..3]. Not NOAH defined.

O

OcclEarCrv	The Occlusion Ear curve is the response of the occluded ear, no compensation in input or output. Defined SET OF 169 TFreqMeasPoint.
Occlusion	Hearing Instruments, headphones, earmolds, etc. create occlusion effects (the effect of closing one or two ear canals). Ref. [Mueller].
OpenEarCrv	Open Ear Curve: The response of the unoccluded ear: No compensation in input or output.
Output	Output Level Attack or Release curve. Component of the structure TARMeasPoint of type TdB10 (or centiBel).
Output1	Output Level at Freq1. Component of the structure TTIMDistMeasPoint of type TdB10 (or centiBel).
Output1Harm	Output level at base frequency. Component of the structure TTHDDistMeasPoint of type TdB10 (or centiBel).
Output2	Output Level at Freq2. Component of the structure TTIMDistMeasPoint of type TdB10 (or centiBel).
Output2Harm	Output level at 2 * base frequency. Component of the structure TTHDDistMeasPoint of type TdB10 (or centiBel).
Output3Harm	Output level at 3* base frequency. Component of the structure

	TTHDDistMeasPoint of type TdB10 (or centiBel).
OutputDif1	Output Level at Freq2 - Freq1. Component of the structure TTIMDistMeasPoint of type TdB10 (or centiBel).
OutputDif2	Output Level at 2 * Freq1 - Freq2. Component of the structure TTIMDistMeasPoint of type TdB10 (or centiBel).

P

Predelay	The predelay is saved in the Attack Release Curve as the no. of milliseconds included in the curve before the level change.

R

REAR	The Real Ear Aided Response (REAR) is the Sound Pressure Level, as a function of frequency, at a specified measurement point in the ear canal for a specified sound field with the hearing aid in place and turned on. This can be expressed either in SPL or as gain in decibels relative to the stimulus level. NOAH uses SPL always. Ref. [Mueller].
RECDCrv	The Real Ear to Coupler Difference curve represents the gain (output minus input) where the result gives the difference between a measurement in a coupler and a measurement in the client's ear. Defined SET OF 169 TFreqMeasPoint.
RECoupler	A component of the outer structure RemData. The measurement contains the Real Ear to Coupler gain.
REIG	The Real Ear Insertion Gain is calculated as follows: REIG = REIR.output - REIR.input The REIG curve is not stored in RemData.
REIR	The Real Ear Insertion Response is the mathematical difference between the REUR (Unaided Response) and the REAR (Aided Response): REIR = REAR - REUR. Ref. [Mueller]. The formula applies for measurement = meas_InsertGain_C (Insertion Response: REIR compensated for REUR) The REIR curve may in DataFmtCodeStd 200 be used as both compensated and uncompensated (for REUR) identified by the value of the RHMeasCond component measurement. The REIR curve may also be used as a gain curve by setting the input part to zero and storing the gain in the output part. If uncompensated, REIR becomes equal to REAR.
ReleaseCrv	A component of the structure TARMeas of type TARCurve. Contains the result of a release measurement.
RemData	Real Ear Measurement Data. One of the two outer structures in

	remhit.h containing 9 different measurements.
REOR	The Real Ear Occluded Response (REOR) is the Sound Pressure Level, as a function of Frequency, at a specified point in the ear canal for a specified sound field, with the hearing aid in place and turned off. This can be expressed either in SPL or as gain in decibels relative to the stimulus level. NOAH uses SPL always. Ref. [Mueller]. The measurement is similar to REAR but the HI is on in REAR.
ResGain	The reserve gain included in the target curve. A component in the TTargetCurve structure, type TdB10.
Resolution	A component of the structure TARCurve. It expresses the time resolution of the Attack / Release Measurement. (In milliseconds).
Result	The result of an Attack / Release Measurement is defined as the point in time where the output of the Hearing Instrument is stabilised at a new level. As stimulus is used a sinus wave.
REUR	The Real Ear Unaided Response (REUR) is the Sound Pressure Level, as a function of frequency, at a specified point in the unoccluded ear canal for a specified sound field. This can be expressed in SPL or as gain in decibels relative to the stimulus level. NOAH uses SPL always. Ref. [Mueller].
RuleName	Text containing the target curve's name/description. Character String of length 51.

S

SignalFreq	Signal Frequency (SignalFreq). A component of the structured type TRHMeasCond. The SignalFreq is set to undefInt for multifrequency measurements, or where the Signal Frequency has no significance.
SignalLevel	Use this field to state the signal level if target curve relates to a specific signal level. A component of the structure TtargetCurve of Type TdB10 (dB x 10 or centiBel).
SignalOutput	The channel/media used to present the signal/stimulus.
SignalType	The signal type used during the measurement.
so_AC. *See* TSignalOutput	The Signal Output (SO) / Stimulus is presented in an Air Conduction (AC) transducer.
so_ExternalBox. *See* TSignalOutput	The Signal Output (SO) / Stimulus is presented in an external test box.
so_ExternalBoxCoil. *See* TSignalOutput	The Signal Output (SO) / Stimulus is presented in an Induction Coil placed in an external test box.

so_FF. *See* TSignalOutput	The Signal Output (SO) / Stimulus is presented in a Free Field (FF) loudspeaker.
so_FFCoil. *See* TSignalOutput	The Signal Output (SO) / Stimulus is presented in a Free Field (FF) induction coil.
so_InternalBox. *See* TSignalOutput	The Signal Output (SO) / Stimulus is presented in a Built-in test box.
so_InternalBoxCoil. *See* TSignalOutput	The Signal Output (SO) / Stimulus is presented in an Induction Coil placed in a built-in test box.
so_NU. *See* TSignalOutput	The Signal Output is **undefined**. This value is assigned to the constant undefInt.
so_User[1..3]. *See* TSignalOutput	The Signal Output (SO) is defined as User Signal Output No. [1..3]. Not NOAH defined.
SPL90	Sound Pressure Level 90 dB re 20 microPascal. (µPa). A component in the outer structure HitData. A 90 dB SPL tone or warble tone is delivered as input to a test box In NOAH, SPL90 stands for the IEC measurement OSPL-90 or the ANSI measurement SSPL-90.
st_NarrNoise. *See* TSignalType	The Signal Type (ST) is 1/3 octave bandwidth filtered noise.
st_NU. *See* TSignalType	The Signal Type (ST) is **undefined**. This value is assigned to the constant undefInt.
st_Patient. *See* TSignalType	The Signal Type (ST) is Speech Weighted Noise . The client's own voice is used.
st_PinkNoise. *See* TSignalType	The Signal Type (ST) is Pink weighted noise, i.e. noise with equal energy content for equal relative bandwidths, e.g. 1/3-octave bands. An averaged FFT-analysis of pink noise measured in 1/3-octave bands (or other relative bandwidths) shows a flat spectrum. A pink noise signal is produced by low pass filtering white noise with -3 dB/octave.
st_SpeechNoise. *See* TSignalType	The Signal Type (ST) is Speech Weighted Noise (Refer ANSI 3.42 or other).
st_Tone. *See* TSignalType	The Signal Type (ST) is a Pure sine wave Tone.
st_TwoTone. *See* TSignalType	The Signal Type (ST) consists of two simultaneous pure tones.
st_User[1..3]. *See* TSignalType	The Signal Type (ST) is User Signal Type No.[1..3]. Not NOAH defined.
st_Warble. *See* TSignalType	The Signal Type (ST) is Modulated sine wave.

st_WhiteNoise. *See* TSignalType	The Signal Type (ST) is White Noise, i.e. noise with equal energy content for equal absolute bandwidths in Hz. An averaged FFT-analysis in of white noise shows a flat spectrum.

T

TARCurve	Attack / Release curve. A tone stimulus is used as input. At time = Tattack the level of the input is increased / decreased momentarily and the output from the Hearing Instrument is measured and saved in a TARCurve.
Target	The target curve. The curve consists of 24 curve points of type TTargetPoint. A component of the TTargetCurve structure. The target is the *calculated* optimum frequency response curve for a Hearing Instrument. Note, that the REIG curve represents a *measured* frequency response curve for a Hearing Instrument
TargetFreq	Target Frequency. A component of the TTargetPoint structure of the type THertz.
TargetGain	Target Gain. A component of the TTargetPoint structure of the type TdB10.
TARMeas	A complete curve set containing a full Attack / Release test.
TARMeasPoint	A measurement point for Attack / Release measurements. This structure only contains an output level value, the input curve is not saved.
TARTime	Attack / Release time measured in milliseconds. The TARTime is defined Integer.
TBatMeas	Battery Measurement. A Component of the outer structure HitData.
TBatMeasPoint	Battery Current Measurement Point. This structured type contains [Frequency, Current].
TBattImp	The impedance of a Hearing instrument Battery as simulated by the REM / HIT test equipment in Ohms x 1000 (milliOhms). Defined as unsigned Integer.
TBattPill	The Battery Pill type: “312”, “13”, “230” or “675”.
TBattType	The type of the battery: Mercury or Zinc-Air.
TBattVoltage	The voltage of a Hearing Instrument Battery as simulated by the REM / HIT test equipment in Volt x 1000 (milliVolts). Defined as unsigned Integer.

TCouplerType	Enumerated type. Represents the actual placement of the probe microphone.
TdB10	A Level (often Sound Pressure Level) expressed in dB x 10 or centiBel.
TDevTypeCode	Device Type Code, defined as Integer in noahdef.h.
TFittingRule	The fitting rule is used for the calculation of a target curve. Refer FittingRule.
TFreqMeas	A Generic, two-channel frequency response type measurement, referenced in the outer remhit.h data structures RemData and HitData.
TFreqMeasPoint	This curve point contains a frequency plus input and output level as measured at the stored frequency. The type is referenced in the structures TFreqMeas, TOcclMeas, TRECMeas and EINMeas.
THDPct	The calculated Third Harmonic Distortion based on the Output 1 and 3 levels.
THIType	The type of the Hearing Instrument (HI) being tested. Defined Enumerated, the values are named hit_<HI Type>.
TIMPct	The calculated result of the Total Intermodulation Distortion Measurement. Type TPct100 or 1/10 000.
TIOMeas	Input / Output Measurement. A Component of the outer structure HitData.
TIOMeasPoint	Input / Output Measurement Point. This structured type consists of Input and Output levels of type TdB10. The measuring frequency is found in the associated Measuring Conditions.
TmA100	Current expressed in milliAmpere x 100 or 1/100 000 A.
TManufCode	Manufacturer code and its allowed values defined as an Integer with defined values in noahdef.h. Referenced in the TRHMeasCond and TTargetCurve structures.
TMeasMode	The Measurement Mode used during measurement.
TMeasurement	Identification of the measurement, mainly for use with data export and storage of not yet publicly defined measurements at the end of the public buffer. The measurement found in the extension part of the public buffer might be a structure that is to be incorporated in the public part at a later stage. TMeasurement is used to provide additional information about the curve stored as Insertion Gain (REIR). This curve may be stored as a curve with or without Unaided Response (REUR). TMeasurement is able to specify this.
Tmm10	Length in tenths of a millimetre.

TOcclMeas	Occlusion effect measurement: A comparison between the respon-se of the unoccluded ear with the response of the occluded ear.
TOhm1000	Impedance in Ohm x 1000 (milliOhms). Defined as unsigned Integer.
TPct100	Percentage x 100. (or 1/10 000).
TRECMeas	The Real Ear to Coupler Difference measurement yields the gain (output minus input), where the result gives the difference between a measurement in a coupler and a measurement in the Client's ear. As both parts of this measurement are performed with the same measurement conditions, the curve can be used in a manner with the input part holding the coupler SPL output curve, and the output part holding the Real ear SPL output curve. This will yield the correct gain value.
TRHMeasCond	REM HIT Measurement Conditions. The measurement conditions are placed as the first component in the following structured types: TOcclMeas, TRECMeas, TTHDDistMeas, TIOMeas, TBatMeas, TARMeas and EINMeas.
TTargetCurve	Target Curve: A full target curve including the description. For saving the target gain calculated for a Hearing Instrument.
TTargetPoint	Target Point: A single element/point of the target curve. Contains frequency and calculated level of the target gain of a Hearing Instrument.
TTHDDistMeas	Second and Third Harmonic Distortion Measurement (THDDistMeas). The type is a component of the outer type RemData defined in remhit.h.
TTHDDistMeasPoint	Second and Third Harmonic Distortion Measurement Point (THDDistMeasPoint). A single measurement point for Harmonic Distortion Measurements. Referenced in the structure TTHDDistMeas.
TTIMDistMeas	InterModulation Distortion Measurement. A component of the outer structure HitData.
TTIMDistMeasPoint	A single measurement point for InterModulation Distortion Measurement. Referenced in the structure TTIMDistMeas. Contains two frequencies at which the curve point is recorded, two input levels and 4 output levels: Freq1, Freq2, Freq2-Freq1 and 2*Freq1 – Freq2.
TV1000	Voltage x 1000 (milliVolts). Defined as unsigned Integer.

U

undefInt	The Integer value -32767. (#8001 hex). Used to indicate that a value is undefined. It is a value that is assigned to the constant undefInt Ref. [Framework].
unknown.	The Integer value 0. (#0000 hex). When used as a parameter

	value it means that the parameter is **defined** however, to an unknown value.
UseRECoupler	TRUE if measurement done using Real Ear to Coupler Difference

V

VentDiam	The diameter of the vent canal. A component of the TTargetCurve structuredtype of type Tmm10. (measured in mm x 10, tenths of a millimetre)
VentLen	The length of the vent canal. Remarks as for VentDiam.

Appendix B: The header file REMHIT.H

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//                                                                                                                              //
// REMHIT.H                                                                                                                     //
//
    //
//  Based on REMHIT.PAS for NOAH version 2.0.                                                                                       //
//  Updated file for public format 200.                                                                                             //
//  Original data structure unchanged, interpretation and handling descriptions added.                                          //
//                                                                                                                              //
//  NOTE!   UndefInt can be used in any data field to indicate that a value is not                                              //
//                              applicable.
    //
//
    //
//  Peter Kossek, REXTON DANPLEX A/S, DENMARK
    //
//  August 1, 1996.
    //
//
    //
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

#include "noahdef.h"

#ifndef __REMHIT_H
#define __REMHIT_H

//
// TSignalType : The signal type used during the measurement.
//
enum TSignalType {
    st_Tone=1,
        st_Warble,
        st_NarrNoise,
        st_TwoTone,
        st_WhiteNoise,
        st_PinkNoise,
        st_SpeechNoise,
        st_Patient,
        st_User1,
        st_User2,
        st_User3,
    st_NU=UndefInt
};

//
//  TSignalOutput :         The channel/media used to present the signal/stimulus.
//
enum TSignalOutput {
    so_InternalBox=1,
        so_ExternalBox,
        so_FF,
        so_InternalBoxCoil,
        so_ExternalBoxCoil,
        so_FFCoil,
        so_AC,
        so_User1,
```

```
        so_User2,
        so_User3,
    so_NU=UndefInt
};

//
//  TCouplerType :      The device in which the probe microphone is placed,
//                      in other words: this is where the output is read.
//

enum TCouplerType {
    ct_None=1,
        ct_RealEar,
        ct_711,
        ct_2cc,
        ct_FreibKK,     // Freiburger Konischer Kuppler
        ct_FreibKKK,    // Freiburger Konischer Kinder Kuppler
        ct_User1,
        ct_User2,
        ct_User3,
    ct_NU=UndefInt
};

//
// TBattType : The battery type used.
//

enum TBattType {
    bt_None=1,
        bt_Mercury,
        bt_ZincAir,
        bt_OtherType,
        bt_User1,
        bt_User2,
        bt_User3,
    bt_NU=UndefInt
};

typedef     WORD        TV1000,                 TOhm1000;
typedef     TV1000      TBattVoltage;           // The voltage of the battery in milliVolts
typedef     TOhm1000    TBattImp;               // The impedance of the battery in milliOhms

//
// TBattPill : The type of battery pill
//

enum TBattPill {
    bp_None=1,
        bp_Bat312,
        bp_Bat13,
        bp_Bat230,
        bp_Bat675,
        bp_User1,
        bp_User2,
        bp_User3,
    bp_NU=UndefInt
};
```

```
//
//  TMeasMode:          Which measurement mode was used during the measurement
//

enum TMeasMode {
    mm_Sweep=1,
        mm_FFT,
        mm_TimeMeas,
        mm_Battery,
        mm_User1,
        mm_User2,
        mm_User3,
    mm_NU=UndefInt
};

//
//  TMeasurement:       Identification of the measurement, mainly for use with data export
//                              and storage of not yet publicly defined measurements at the end of the
//                              public buffer.
//

enum Tmeasurement {
    meas_AudHand=1,     // Audiometry
        meas_TargHand,  // Target Curve
        meas_UnAided,   // Unaided Response
        meas_OcclResp,  // Occluded Response
        meas_InsertGain_C,              // Insertion Response (compensated for REUR)
        meas_AidedResp, // Aided Response
        meas_InputOutput,// Input Output
        meas_HarmDist,  // Harmonic Distortion
        meas_OcclEff,   // Occlusion Effect
        meas_RECD,      // Real Ear to Coupler Difference
        meas_SPL90,     // XSPL-90, i.e. OSPL-90 for IEC and SSPL-90 for ANSI
        meas_FOG,       // Full On Gain
        meas_FreqResp,  // Frequency Response
        meas_BattCurr,  // Battery Current
        meas_InterDist, // Intermodulation Distortion
        meas_EquivNoise,// Equivalent Input Noise
        meas_AttackRec, // Attack/Recovery
        meas_IndCoil,   // Induction Coil
        meas_User1,     // User specific #1
        meas_User2,     // User specific #2
        meas_User3,     // User specific #3
        meas_InsertGain_U =             50,             // Insertion Response (uncompensated for REUR)
        meas_NU=UndefInt
};
```

```
typedef struct {
    TManufCode          ManufCode;
    TDevTypeCode        DevTypeCode;
    TSignalType         SignalType;
    TSignalOutput       SignalOutput;
    TdB10               SignalLevel;
      //
    // Set this to UndefInt for multifrequency measurements,
      // otherwise state the frequency here
      //
    THertz              SignalFreq;
    TBattType           BattType;
    TBattPill           BattPill;
    TBattVoltage        BattVoltage;
    TBattImp            BattImp;
      //
    // TRUE, if measurement done using Real Ear to Coupler
    // Difference
    //
    BOOL                UseRECoupler;
    TMeasMode           MeasMode;
    Tmeasurement        Measurement;
} TRHMeasCond;

typedef struct {
    THertz              Freq;       // The frequency at which Input/Output was recorded
    TdB10               Input;      // Input value
    TdB10               Output;     // Output value
} TFreqMeasPoint;

//
//  TFreqMeas:          generic, two-channel frequency response type measurement
//

typedef struct {
    TRHMeasCond         MeasCond;
    TFreqMeasPoint      FreqCrv[169];
} TFreqMeas;

//
//  TOcclMeas:          Occlusion Effects, comparison between the response of the unoccluded
//                      ear with the reponse of the occluded ear.
//                      OpenEarCrv is the response of the unoccluded ear, no compensation in
//                      input or output.
//                      OccEarCrv   is the response of the occluded ear, no compensation in input
//                      or output.
//

typedef struct {
    TRHMeasCond         MeasCond;
    TFreqMeasPoint      OpenEarCrv[169];
    TFreqMeasPoint      OcclEarCrv[169];
} TOcclMeas;
```

```
//
//  TRECMeas :          Real Ear to Coupler Difference measurement
//
//  RECDCrv is to be read as gain (output - input), where the result gives the difference between
//          a measurement in a coupler and a measurement in the client's ear. As both parts of
//          this measurement are performed with the same measurement conditions, the curve can be
//          used in a manner with the input part holding the coupler SPL output curve, and the
//          output part holding the real ear SPL output curve. This will yield the correct gain
//          value.
//

typedef struct {
    TRHMeasCond         MeasCond;
    TFreqMeasPoint      RECDCrv[169];
} TRECMeas;

//
//  TFittingRule :      The fitting rule used for the calculation of a target curve
//
enum TFittingRule {
        fr_POGO=1,
        fr_POGOII,
        fr_NAL,
        fr_NALProf,
        fr_Berger,
        fr_HalfGain,
        fr_ThirdGain,
        fr_DSL,
        fr_LIBBY,
        fr_Byrne,
        ft_CoxMSU,
        fr_User1=100,
        fr_User2,
        fr_User3,
        fr_User4,
        fr_User5,
        fr_User6,
        fr_User7,
        fr_User8,
        fr_User9,
        fr_User10,
        fr_Undefined=UndefInt
};
```

```
//
//  THIType :           The type of the hearing instrument being tested.
//

enum THIType {
    hit_ITE=1,
        hit_BTE,
        hit_ITC,
        hit_MITC,
        hit_Body,
        hit_User1,
        hit_User2,
        hit_User3,
        hit_User4,
        hit_User5,
        hit_Undefined=UndefInt
};

typedef int Tmm10;

//
//  TTargetPoint :      a single element/point of the target curve.
//

typedef struct {
    THertz    TargetFreq;
    TdB10     TargetGain;
} TTargetPoint;

//
// TTargetCurve : a full target curve including the description.
//

typedef struct {
    TManufCode          ManufCode;
    TDevTypeCode        DevTypeCode;
    TFittingRule        FittingRule;
    THIType             HIType;
    Tmm10               VentDiam;   // The diameter of the vent canal
    Tmm10               VentLen;    // The length of the vent canal
    TdB10               ResGain;    // The reserve gain included in the target curve
    TCouplerType        CouplerType;
      //
      // Use this field to state the signal level if target curve relates to a specific signal level.
      //
    TdB10               SignalLevel;
    TTargetPoint        Target[24];  // The target curve
    char                RuleName[51]; // Text containing the target curve's name/description
} TTargetCurve;
```

```
typedef int TPct100;


//
//   TTHDDistMeasPoint: A single measurement point for Harmonic Distortion Measurements.
//

typedef struct {
    THertz    Freq;                 // Signal frequency
    TdB10     Input;                // Signal level
    TdB10     Output1Harm;          // the output value at Freq
    TdB10     Output2Harm;          // the output value at 2*Freq
    TdB10     Output3Harm;          // the output value at 3*Freq
    TPct100   THDPct;               // the calculated THD based on OutputXHarm
} TTHDDistMeasPoint;

//
//   TTHDDistMeas:       A complete curve containing a Harmonic Distortion measurement.
//

typedef struct {
    TRHMeasCond         MeasCond;
    TTHDDistMeasPoint   DistCrv[161];
} TTHDDistMeas;

//
//   TTIMDistMeasPoint:          A single measurement point for InterModulation Distortion
//                               Measurements.
//

typedef struct {
    THertz    Freq1;        // First stim freq
    THertz    Freq2;        // Second stim freq
    TdB10     Input1;       // Level of first stim
    TdB10     Input2;       // Level of second stim
    TdB10     Output1;      // Output at Freq1
    TdB10     Output2;      // Output at Freq2
    TdB10     OutputDif1; // Output at Freq2-Freq1
    TdB10     OutputDif2; // Output at 2*Freq1-Freq2
    TPct100   TIMPct;       // The calculated result
} TTIMDistMeasPoint;

//
// TTIMDistMeas: A complete curve containing an Intermodulation Distortion measurement.
//

typedef struct {
    TRHMeasCond         MeasCond;
    TTIMDistMeasPoint   DistCrv[161];
} TTIMDistMeas;
```

```
//
//  TIOMeasPoint:       A measurement point for Input/Output measurements.
//
//  Note:               No frequency information in single points. This is common for all the
//                      measurement points and the value is stated in MeasCond.
//

typedef struct {
    TdB10    Input;
    TdB10    Output;
} TIOMeasPoint;

//
//  TIOMeas :           A complete curve containing an Input/Output measurement.
//

typedef struct {
    TRHMeasCond         MeasCond;
    TIOMeasPoint        IOCrv[61];
} TIOMeas;

typedef     unsigned    TmA100;             // Current value in hundredths of a milliampère

//
//  TBatMeasPoint :     A measurement point for Battery Current measurements.
//

typedef struct {
    THertz    Freq;
    TmA100 Current;
} TBatMeasPoint;

//
//  TBatMeas :          A complete curve containing a Battery Current measurement.
//

typedef struct {
    TRHMeasCond         MeasCond;
    TBatMeasPoint       BatCrv[169];
} TBatMeas;

typedef int TARTime;    // Attack/Release time in milliseconds

//
// TARMeasPoint : a measurement point for Attack/Release measurements.
//

typedef struct {
    TdB10    Output;
} TARMeasPoint;
```

```
//
// TARCurve : a single curve for partial info of an Attack/Release measurement.
//

typedef struct {
    TARMeasPoint        Curve[256];             // The curve itself
      //
      // Time result (in milliseconds) (result depending on whether this is AttackCrv or ReleaseCrv)
      //
    int         Result;
      //
      // The time resolution of the measurement (in milliseconds).
      //
    int         Resolution;
      //
      // The number of milliseconds included in the curve before level change.
      //
    int         Predelay;
} TARCurve;

//
//  TARMeas :           A complete curve set containing a full Attack/Release test.
//

typedef struct {
    TRHMeasCond         MeasCond;  // Generic measurement conditions.
    TdB10               LevelStep;   // The level step size from the start level defined in MeasCond.
    TARCurve            AttackCrv;   // The curve where the stimulus level increases.
    TARCurve            ReleaseCrv; // The curve where the stimulus level decreases.
} TARMeas;

//
//
//

typedef struct {
    TRHMeasCond         MeasCond;
    TFreqMeasPoint      EINCrv[169];
      //
      // Input part: uncompensated level from the test chamber
      // Output part: coupler output level compensated with the gain value
      //
    TdB10               EINRMS;    // The calculated result.
} EINMeas;
```

```
//
//  RemData :           Defines structure for storing a Real Ear Measurement
//

typedef struct {
    TTargetCurve        Target[3];
      //
      // Unaided Response: input is the input level, output is uncompensated probe level.
      //                   Must be valid for use as an output curve as well as a gain curve,
      //                   so if only output has been measured, fill the input part with the
      //                   stimulus level value.
      //
    TFreqMeas           REUR;
      //
      // Occluded Response: input is the input level, output is uncompensated probe level.
      // Actually like Aided Response(REAR) but the H.I. is off or detached during the measurement.
      //
    TFreqMeas           REOR;
      //
      //        Insertion Response:
    //                  Input is the input level, output is the probe level with or
      //                        without REUR compensation.
      //        Measurement =     meas_InsertGain_U -> the output values are dB SPL values without
any
      //                  REUR compensation.
      //        Measurement =     meas_InsertGain_C -> the output values are dB SPL values less the
      //                  REUR's gain, i.e. with REUR compensation.
      //                        Gain curves may be stored here. In this case the Input part must be set
to
    //                  zero and the gain value must be stored in the Output part.
      //
    TFreqMeas           REIR[5];
      //
      // Aided Response: input is the input level, output is uncompensated probe level.
      //
    TFreqMeas           REAR[5];
    TIOMeasIOMeas[5];
    TTHDDistMeas        HarmDistortion[3];
      //
      // See the TOcclMeas description
      //
    TOcclMeas           Occlusion[3];
      //
      // See the TRECMeas description
      //
    TRECMeas            RECoupler;
} RemData;
```

```
//
//  HitData :                   Defines structure for storing a Hearing Instrument Test
//

typedef struct {
    TFreqMeas           SPL90[2];
    TFreqMeas           FullOnGain[2];
    TFreqMeas           FreqResp[2];
    TBatMeas            Battery[2];
    TTHDDistMeas        HarmDistortion[2];
    TTIMDistMeas        InterDistortion[2];
    EINMeas             EquivInputNoise[2];
    TIOMeasIOMeas[2];
    TARMeas             AttackRecover[4];
    TFreqMeas           InductionCoil[2];
} HitData;

#endif
```

Index

H

I

L

M

O

P

R

S

T

U

V