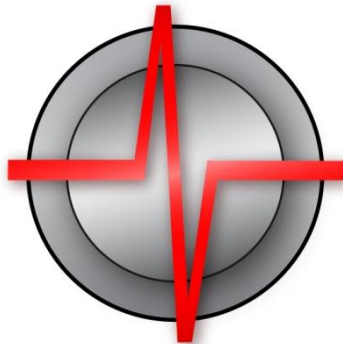


ROTAS

Noise Analysis System



Manual and Introduction

© 2019 Discom Industrial Measurement and Test Systems Inc.

Maschmühlenweg 81, 37081 Göttingen, Germany

Tel.: +(49) 551 548 33 10

Fax: +(49) 551 548 33 43

Email : info@discom.de

www.discom.de

ROTAS

Noise Analysis System

Contents

Safety Instructions.....	5
Introduction	10
About This Manual.....	10
Components of the Analysis System.....	11
The Measurement Computer	13
The TAS Box.....	15
Concepts and Basics	16
Important Terms	16
Limit Values	22
Noise Analysis Theory	27
The TasAlyser Program.....	35
The Project Directory	35
Test Bench Connection.....	38
Wave Audio Recording and Playback.....	39
Parameter Administration with TasForms.....	40
The Database in the Overall System.....	40
Creating and Erasing a Type	41
General Form Functions	46
Setting Limits	48
Learn Parameters	50
Defining Error Codes	52

Trigger	54
Measurement Archives and Evaluation	57
Archiving in the TasAlyser.....	57
The Presentation App / Marvis	58
Help from Discom.....	60
When the TasAlyser Does Not Work.....	61
Strange Noises	62
Unwanted Test Results	63



Safety Instructions

Appropriate use

The TAS device and the connected transducers may be used for measurement and directly related control tasks only. Any other use is not appropriate.

To ensure safe operation, the device may only be used as specified in the operating manual. It is also essential to follow the respective legal and safety regulations for the application concerned during use. The same applies to the use of accessories.

Maintenance and cleaning

The device is maintenance-free. Please note the following when cleaning the housing:

Before cleaning, disconnect the equipment completely.

Clean the housing with a soft, slightly damp (not wet) cloth. Never use solvents, since these could damage the labeling on the front panel and the display.

Do not apply high water pressure to the unit for cleaning.

Conditions on site

For all devices:

Observe the maximum permissible ambient temperatures given in the specifications.

Minimize device exposure to direct sunlight in hot operating environments.

General dangers of failing to follow the safety instructions

The device is a state of the art device and, as such, is fail-safe. The module may give rise to further dangers if it is inappropriately installed and operated by untrained personnel. Any person instructed to carry out installation, commissioning, maintenance or repair of the module must have read and



understood the User Manuals and in particular the technical safety instructions.

Remaining dangers

The scope of supply and performance of the module covers only a small area of measurement technology. In addition, equipment planners, installers and operators should plan, implement and respond to the safety engineering considerations of measurement technology in such a way as to minimize remaining dangers.

Prevailing regulations must be complied with at all times. There must be reference to the remaining dangers connected with measurement technology.

Product liability

In the following cases, the protection provided for the device may be adversely affected. Liability for device functionality then passes to the operator:

The device is not used in accordance with the operating manual.

The device is used outside the field of application described in this chapter.

The operator makes unauthorized changes to the device.

Working safely

The equipment complies with the EMC standards of EN 61326-1.

These standards define emission limits and immunity requirements for multiple environments.

With respect to emissions, the standards contain limits for industrial (class A) and residential / commercial (class B) environments. The standard herein references CISPR 11:2009+A1:2010.

With respect to immunity, the standards contain limits for electromagnetic protected (lowest requirements), general and industrial (highest requirements) environments.

The TAS28 device listed in the declaration of conformity is conformal to the requirements for:

Emissions: Class A

Immunity: Industrial



The TAS28 device and its modules are intended for use in an industrial environment. When used in residential or commercial environments, additional arrangements may be required to limit electromagnetic emissions.

Conversions and modifications

The TAS hardware must not be modified. Any modification shall exclude all liability on our part for any resultant damage.

In particular, any repair or soldering work on any printed circuit boards or replacement of components is prohibited. When exchanging complete modules, use only original parts from Discom.

The device is delivered from the factory with a fixed hardware and configuration. Changes can only be made within the possibilities documented in the manuals.

Qualified personnel

Qualified persons means persons entrusted with the installation, fitting, commissioning and operation of the product who possess the appropriate qualifications for their function. This TAS device is only to be installed and used by qualified personnel, strictly in accordance with the specifications and the safety rules and regulations.

This includes people who meet the following requirements:

As test plant operating personnel, you have been instructed how to handle the machinery and are familiar with the operation of the devices and technologies described in this documentation.

As commissioning engineers or service engineers, you have successfully completed the training to qualify you to repair the test systems. You are also authorized to activate, to ground and label circuits and equipment.

It is also essential to comply with the legal and safety requirements for the application concerned during use. The same applies to the use of accessories.

General Installation Instructions

The system must be connected to a power ground via the power cable for operation. The system shall only be opened by electrically trained personal. The power needs to be disconnected before opening the system.

Installation should be done by personnel trained to do electrical installations. Please refer to your local security guidelines. To avoid failures and to



Safety Instructions

increase lifetime of the analysis system, make sure that the installation site meets the following criteria:

- It is not near other heat sources,
- It is not near a magnetic device
- It is not in a damp and / or dusty environment
- It provides sufficient air supply for the front-side fan

Cable Installation Instructions

Proper grounding of the unit needs to be accomplished. The ground connection shall be done

- For the PC: using the power cable ground
- For a frontend TAS28, TAS48 or TASNano in connection with a ground free PC or laptop: Connect a ground wire to the frontend case. Use the TIS speed connector as a ground.

The signal cables need to follow these guide lines:

- Analog BNC and power supply cables need to be installed in grounded metallic cable trays for cable length of more than 2 m.
- The cable trays need to have a minimum distance of 50 cm to power cables with currents over 5 A and/or pulse content as in E-Drive power.
- For the TIS speed cables only use Discom supplied cables. If these are not available, make certain that the signal wires are twisted pairs for the RS422 inputs and that the whole cable has a ground connection to the shield.
- The maximum permissible cable length for any signal connection to the front end is 30 m.

Frontend Power Supply Requirement

The frontend needs to be connected to a dedicated power supply capable of driving 12V @ 1Amp. This power supply shall have no other loads connected.



Service notes

This applies to systems delivered together with a Rittal panel with air condition and a UPS unit.

The proper function of the air condition unit should be checked quarterly. Clean the heat exchanger as necessary.

The battery of the Uninterruptable Power Supply has a limited lifetime of about 3 years. Replace it when the UPS software tells you to do so.



Introduction

About This Manual

This manual describes the Rotas Noise Analysis System, focusing on the measurement program and the parameter database. The goal is to enable you, as noise analysis users, to operate the system in everyday situations and to deal with the tasks which typically occur.

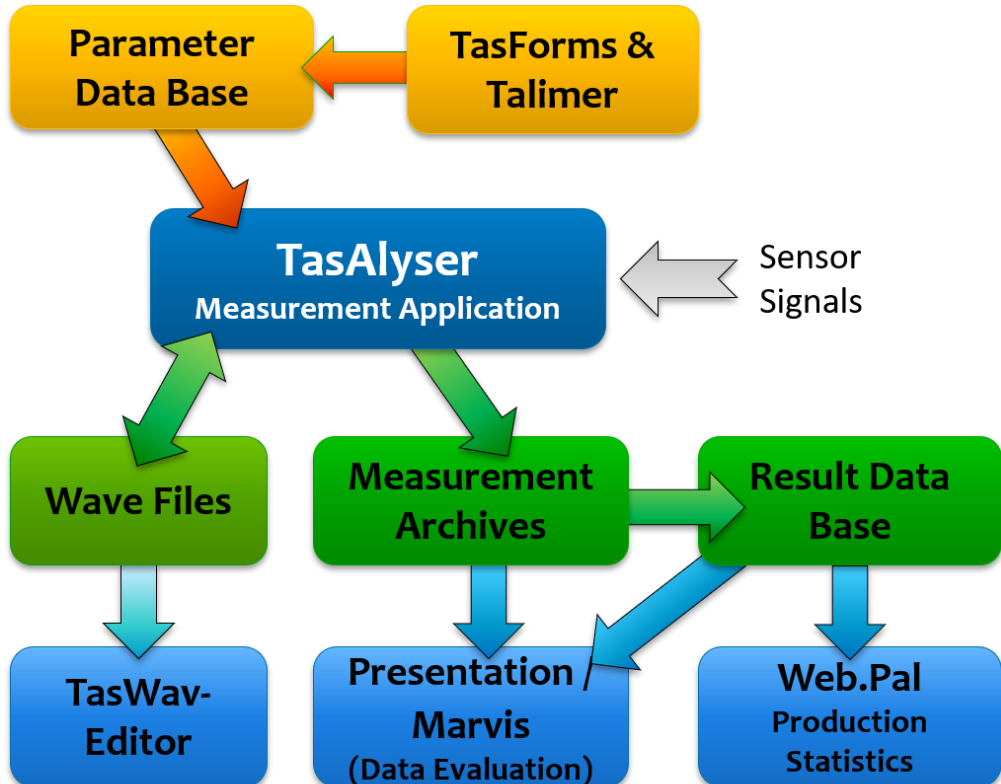
The noise analysis system consists of several components (see the next section). Each of these components is very efficient and offers numerous possibilities for widely diverse applications and tasks. Therefore this manual can only serve as an introduction and is not able to cover each detail – this task is reserved for the specialized manuals on the individual components.

This manual assumes a “typical” application situation in noise analysis, the routine testing of aggregates (e.g. gearboxes) on an end-of-line test bench. On the test bench, different types of aggregates (like gearboxes with different gear ratios) are tested. Noise analysis helps to find loud aggregates and thereby identify different kind of defects. One of the main tasks for the operator is to take care of the limit values which means drawing the line between good (O.K.) and bad (not O.K.).

The Rotas system can also be used for mobile noise measuring, for instance in car test drives, or in the continuous operation testing of individual aggregates at test benches. In principle, the routine test and the mobile measurement are very similar.

Components of the Analysis System

This manual describes the Rotas noise analysis system and its central components. These components are shown in the picture and explained below:



- The “TasAlyser“ measurement program: The TasAlyser runs on a PC which is connected to a “TAS Box” data collector. The TasAlyser program operates in real time, generates measured values and variables, compares these with limit values, evaluates them accordingly and stores the results in a *measurement data archive*.
- The parameter database *TasForms* and *Talimer*: The user interface of the parameter database *TasForms* is used to administer the design data of the candidates and types, so that the TasAlyser program can calculate order position and gear ratios. Furthermore, the parameter database sets which measurement procedure shall be applied and which measured values shall be generated. Finally, the parameter database contains the settings for the generation of limit values. For



editing limit curves, you can use the tool Talimer (=TAs LIMit Editor) which is especially designed for this task.

- *TasWavEditor* can display the wave files recorded by TasAlyser, including all channel information, test run marks and other metadata. You can also play back and listen to individual sensors.
- The *result database*: The TasAlyser stores the results and data of each measurement in a file, the measuring data archive. An auxiliary program, the Collector, sorts these files into a central database. This database serves as the basis for statistic analyses to set limit values, as well as for answering questions on the characteristics of individual aggregates, which were, in some cases, measured some time ago.
- The production statistics tool *Web.Pal*: This Intranet-based tool uses the result database to make production and error statistics available. In the analysis of the distribution and the time series of measured values, Web.Pal also offers an early warning function, which recognizes possible failure issues, before there are real failuers at the test stand.
- The evaluation program *Marvis / Presentation*: The information stored in the archives and measurement database can be made visible and evaluated using Marvis (Measurement ARchive VISualisation). The program also allows the automatic generation of reports, statistical analyses as well as detailed analyses of noise phenomena.

In addition to these major components, there are other elements, such as the Collector mentioned above or the TasWavEditor. These auxiliary programs are also explained in this manual.

The TasAlyser program runs on the measurement computer that is connected to the test bench. Both the parameter administration TasForms and the measured value database can be installed independently of each other, either on the measurement computer or on another computer (server). If several measurement computers (lines) are used in parallel, then installation on a server is preferable since all the test benches can be administered by one parameter database and the results can be filed in a common measured value database.

The evaluation programs (WebPal, Marvis) access the data via network. Both can therefore be run locally on the measurement computer or on the server, as well as on any other workstation which can access the database. Similarly, TasForms and Talimer, the user interfaces of the parameter database, can be run on another computer connected by network.



“Getting started”

Usually, there are links on the measurement computer desktop to start the measurement program, the parameter administration and the evaluation:



TasAnalyser



TasForms



Talimer



Presentation

Usually, the desktop shows also a folder “Rotas for Experts” containing the links above (which are the most important tools on the measurement computer) and more.

Normally, a reboot of Windows will automatically start the measurement program.

The Measurement Computer

The measurement computer is a Windows PC which is equipped with the TAS hardware for data collection. The TAS Box is of modular design and equipped according to the requirements of the test. More details on the TAS Box can be found in the section “

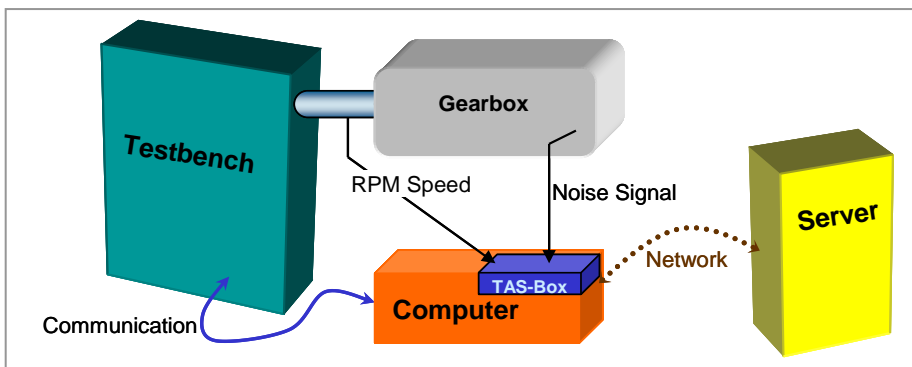


The TAS Box” below.

The TAS Box is connected to the measurement computer via USB. If the TAS hardware is built into the measurement computer, the USB wire can be reached from the outside. With mobile systems the TAS Box is run as a separate piece of equipment. For demanding applications several TAS Boxes can be used at one computer.

With the Tas Box, the TasAlyser program records sensor data like noise, rotational speeds, sometimes torque, temperature etc. In order to carry out the test the TasAlyser also requires information about the test cycle, especially type and serial number of the aggregate or the current test step (gear). This information is transmitted to the measurement computer by the test bench control. The test bench control on the other hand can enquire evaluation results, as well as further information, such as error reports.

Usually, the measurement computer is connected with a server over a network. The measurement data archives are sent to this server for sorting into the database.



The figure shows the measurement computer in its environment:

The network connection to the server is optional and can only be available intermittently. A permanent network connection, however, opens up the possibility of remote maintenance of the measurement computer.

In case of a mobile system, there is no test bench. Communication occurs between the measurement program and the driver. If desired, the server can be connected before and after mobile measurements.

Communication with the Test Bench

The connection with the test bench can be done in many different ways, such as, for example, an “antiquated” serial line, Profibus or UDP network protocol. In most cases, the TasAlyser program and the test bench software



communicate over a command-oriented protocol with plaintext instructions. A window in the TasAlyser program allows you to monitor communication.

The communication volume depends on the requirements of the task and also on test bench capabilities. Usually, at the begin of a test cycle, the test bench transmits the aggregate type and a serial number, during the test the name of the next test step (e.g. gear when testing gearboxes), and, at the end, the information that the test cycle is completed. The test bench then queries the evaluation result.

The test bench can also query intermediate data, detailed error reports and even measured values. You can find further details on the communication protocol in another section of this manual.



The TAS Box

The special data collection hardware of the TAS System is connected with the measurement PC via USB (think of an external sound card). It consists of individual modules which are built into a separate housing. (PCI card version is also available.) The current standard version is the Tas28, which offers up to 8 A/D channels and 4 input channels called ‘TIS’ for pulsed speed signals. The Tas48 has up to 16 A/D inputs.



Tas28 Box

The A/D converter modules and the TIS module are designed for extremely low power consumption. This allows that a TAS Box with up to four A/D modules and TIS modules can be supplied with USB power only. However, you should use external power supply in a test stand environment. For a mobile system with four A/D Channels and a TIS (four microphones or structure borne noise sensors), USB power is sufficient.



Concepts and Basics

Important Terms

When you work with the noise analysis system, you will keep coming across some terms again and again. Some are from the data base background, others from acoustics or from gearbox design. These terms are described here briefly.

Test Step (Mode)

A complete test, a *test cycle*, consists of a sequence of sections. A possible section in gear box testing, for example, is “3rd gear, rotational speed increasing”. These sections are called *test steps* or *modes*. In each test step the defined measure values are recorded and evaluated individually. Each test step has its own limits. Also for a variety of other settings, there exists one dataset for each test step.

When errors are found, the error message contains information during which test step an error arose. Results are also classified in the measurement data archives according to the test step.

Location, Rotor and Order Source

The noise analysis system tries to specify a noise source as exact as possible. For this, the test candidate is “separated into its parts” for the analysis. These parts of a gearbox for example appear as “Locations” in the system. Most test candidates have turning parts like shafts as well as other parts which make noise such as meshing gears. In the Rotas world, a shaft is a *Rotor* and a meshing gear is an *order source*. The term “order source” shows that this part produces a characteristic frequency (the “base order”) which is expected to show up in the spectrum. On the other hand, the characteristic frequency of a “rotor” is its rotational speed: Everything which turns with the same rotational speed belongs to the same rotor.

The Rotas noise analysis system can be used to test a wide variety of aggregates. The number and characteristics of order sources and rotors in these aggregates is as different as the aggregates themselves. Sometimes, a part is a rotor and an order source at the same time, for example if you test a single gear wheel.

Channels: Synchronous and “Mix”

A central step in noise analysis is the rotationally synchronous analysis (for details see in the relevant section starting from page 27). The rotationally synchronous processing allows to separate the noise contribution of different rotors. The measured values obtained here are known as *synchronous* values



(e.g. synchronous spectra), abbreviated as *Sync*. However, since the noises in an aggregate are not all necessarily linked to a rotor, non rotor synchronous measured values are also generated, which are known as *mix* measured values (e.g. mix spectrum), because they are based on a mixture of all sources of noise.

Depending upon its type, a production error is found either more in synchronous or more in mix. For example, a gear wheel nick is found in the synchronous measured values of the appropriate rotor, a loud bearing, on the other hand, more in mix.

As an option, there can also be another type of processing channel, a *fixed frequency* or *fix channel*. This channel does not refer to the speed of a rotor, but uses a fixed sampling rate. Fixed channels are used, for example, in analyzing background noises (such as gear shift noises in gearboxes).

Once more, to make this quite clear: all processing channels - synchronous channels, mix channel, fixed channel - are processed copies of *one* sensor signal. If the noise analysis system is equipped with several sensors, then each sensor has its own set of synchronous channels, its own mix channel and, if necessary, its own fixed channel.

Instruments

The measurement program calculates (in each test step, see above) a wide range of very different measured values and curves. In order to organize these, we use the term *instrument*: each type of measured variable is formed by an appropriate instrument. For example, there is the instrument “order spectrum”, the instrument “Rms”, or the “Crest” instrument (used for nick detection; see “Crest & Co” on page 29). Many instruments have parameters, which specify the desired calculation in detail. You find the instruments in the parameter database, in the result display of the TasAlyser program, and, of course, in the measurement result archives. The instruments are divided into two main categories: single values and curves. As the name says, the result of a single value instrument consists of a single number. So, for example, the Crest instrument can produce “3.49” as result. Single values are very user friendly: the limit value is also only one number, which allows the results to be shown in a table. Furthermore, single values can easily be evaluated statistically (by generating distributions and time series). Curve instruments, however, produce a curve as their result. Examples are Spectra or level tracks over rotational speed (“order track”). The limits for such instruments are curves as well. The visualization is more complicated for such results, and statistical evaluations are more difficult.

Measurement values and instrument parameter

A Measurement value is a special kind of analysis which a specific instrument does in a specific channel for a specific location (see Instruments



and Channels: Synchronous and “Mix” beginning at page 17). Usually, it depends on the test step whether a specific kind of analysis is possible or not. Some locations are useless in some test steps (if a planetary gear set is blocked, for example, it does not make sense to try an analysis of its gear mesh noise). Most instruments can carry out more than one analysis at one time. In this case, these different kinds of analysis are identified by the instrument parameter. Depending on the instrument, the measurement values can be “single values”, “curves” like spectra and tracks, or curve arrays (“spectrograms”).

Clavis

Some key entries are already used when the measurement program queries the database for settings, for example test bench and type. Still, the measurement program has to distinguish test step, location, channel, sensor, instrument and measurement value. Since this 6 parts are explicitly important for the measurement program to distinguish data, the key which consists of these parts has been given a separate name and is called *clavis*. A *clavis* in the measurement program uniquely identifies a measurement value as well as the corresponding result or its limit. (“Clavis” is Latin and means “key.”)

Key and data set

Each entry in the parameter and result data base needs a unique address, its *key*. Each completely specified key designates a unique data set (parameter set, measurement result), and there cannot be two data sets with the same key. Therefore, a data set is a collection of different data which belong to the “object” specified by the key. For example, the object “Human Being” has first name, name, address, etc.

Type and Base Type

The noise analysis system is designed to deal with several different *types* of aggregates, e.g. gearbox types which differ in their gear ratios, or motor types, which differ in their additional components.

Not all differences between types are relevant for noise analysis. If a manufacturer has two identical types with different names because they are designed for different cars to be built into, this usually is not a difference for noise analysis. Relevant differences are when rotational frequencies differ or when additional noise sources are present. Because you want as little work as possible to set parameters for these types, types which do not differ for aspects of noise analysis find their parameters in the same dataset.

To be more exact, the *types* are the names which the test stand sends to the measurement program for the different analysis objects (e.g. gearboxes). For each type name, there is a *base type* which refers to the dataset. In other



words, the base type is the key to the datasets. Usually, a base type has more than one type name for the test stand to “call” it.

Family or model

Sometimes you have the situation that you want to test aggregates at a test stand which do not only differ in transmission ratios. One such difference is that gear boxes sometimes have 5 and 6 speeds variants. Such a difference is covered by the concept of families or models.

Each aggregate type belongs to a defined family from the beginning. This gives the measurement program the opportunity to notice the differences between families. A 6 speed gear box will have parameters for the 6th gear, a 5 speed transmission will have none.

If you want, you can even have completely different objects in one database. They then belong to different families. The consequence is, that you have to take special care when you make settings. The more the different objects differ, the easier it is to enter useless entries into the database.

Test bench, test bench group

The idea to have as less datasets as possible does also have effect on test benches. Each *test bench group* represents one data set which can be used for different *test benches*. This has the same effect as with types and base types: Each test bench of a test bench group uses the same dataset.

Limits and Error Codes

The real use of noise analysis becomes clear when the different measured values are evaluated. The Rotas system does this by comparing each individual measured value (whether single value or curve) with an individual limit. If the measurement value failed on that limit¹, then an error is announced and the test aggregate is declared “n.O.K.” (“not O.K.”). (Measured value = limit is only just “O.K.”).

The Rotas system does not use school grades or results such as “nearly n.O.K.”. Either an aggregate is good and can be used (sold), or it is not good and must be repaired.

For special applications, classification into several categories is possible. This means, however, a significantly higher parameterization effort for the user (i.e. for you) and should only be used when there are really good reasons for it. In our experience, we have found that an aggregate is either good or bad.

¹ Depending on the kind of analysis, this can mean “limit exceeded” (most limits are evaluated that way), or it can mean “limit not reached” (e.g. for a sensor signal check).



In the parameter database, an error code and limit settings are assigned to every measured value. (Different measure values can, of course, use the same error code if they indicate the same defect). For each error code, there is an error message. If a measure value fails on its limit, you get a message in TasAlyser's output window, consisting of the error code, the error message and further details (such as the test step, the instrument and the location causing that error). Normally, you need relatively few error codes – only as many as you want to have different message texts.

In addition, the error codes can be transmitted to the test bench and then stored on a data medium at the aggregate. In this case, you will probably like to set up more error codes. You can find more about error codes in the chapter on the parameter database.

Command Variables and Triggers

A *Command variable* is a measured value which is used to control the measurement or as reference. The typical command variable is the rotational speed, and for each application of noise analysis at least one rotational speed is required. It may be that your aggregate has several independent rotational speeds. Another common command variable is the torque. Time is also a reference variable but a special one: It always exists. You do not need a special sensor for it.

A typical test cycle consists of a sequence of command variable ramps. For example, that the rotational speed is increased steadily from an initial 1000 rpm. to 4000 rpm. and then lowered back to 1000 rpm. That way, two ramps (one rising and one falling) have been run. For noise analysis this results in two test steps (see above.).

In order to specify a measurement range within a ramp (for example from 1500 to 3500 rpm) and to record value tracks with reference to a speed within this measuring range, the measurement program has a *trigger* function. The trigger settings are specified in the parameter database. In the measurement program, they are used to control measurements (start/ stop) and to generate tracking curves.

The Test Cycle

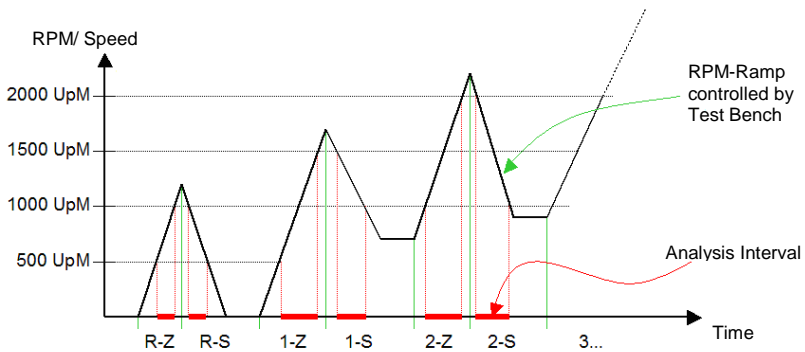
A typical test cycle for an aggregate in end-of-line testing is as follows:

1. The aggregate is clamped onto the test bench. The test bench transmits the aggregate type and serial number to the TasAlyser, whereupon the TasAlyser loads the parameter and limits which are currently valid for this type. This step is called *Insert*. It is the start of the test cycle.



2. The test bench transmits the name of the first test step to the TasAlyser. The TasAlyser begins to monitor the rotational speed (or another reference variable as specified in parameterization).
3. The rotational speed (or other reference variable) reaches the initial value specified in the trigger settings. From now on, measure values are captured. This moment is called *measurement start*.
4. When the reference variable reaches the defined target value, the trigger determines *measurement end*. Measure value capturing is completed now, and the results for this test step are evaluated and displayed.
5. The test bench transmits the name of the next test step. Repeat with step 2.
6. At the end of the test cycle, the test bench transmits the *Remove* instruction to the TasAlyser. At this moment, the final result of the test is evaluated and can then be queried by the test bench. The TasAlyser stores all measurement data in an archive file, which can be sent later to the result database if required.

The following diagram illustrates a typical test cycle for a gearbox: rotational speed is increased (“drive”) and decreased (“coast”) in the ramps. The analysis intervals within the ramps are specified by the trigger settings. The duration of these intervals (in seconds) depends on the steepness of the ramps.



The sequence of the test steps is random, at least as far as the TasAlyser is concerned. A test step may also be repeated (immediately or later), whereby all the results and error messages from the first measurement² are discarded

² It is possible to generate the mean or the maximum of repeated measurements.



and new ones are recorded. You do not have to use all of the test steps which are contained in the database in a test cycle³.

If the test bench announces a new test step (step 5), before the measurement end condition of the previous test step has been reached (step 4), then the results of this test step are discarded and it is regarded as being not measured.

Beside the regular measurements within the test steps, other measurements can be done, which are not connected to the normal test steps. An example is gear shift noises with gearboxes, which typically occur during the *transition* between test steps. Another example is the gear ratio test, with which the TasAlyser tests the correct gear ratio of a gearbox on the basis of two rotational speeds. The gear ratio test is started and terminated by the test bench with separate commands.

The test cycle can also be controlled manually, which is necessary when you use the mobile system. The TasAlyser has a corresponding control window for this purpose (see the following chapter).

It is also possible to cancel a test cycle either by a test bench command or manually. In this case no evaluation result will be generated, all the measured values are discarded and no result data archive is produced.

Limit Values

As already described in the previous section, the noise test uses limit values to separate good and bad. Every measured value has a limit value (or limit curve), which can be influenced individually.

The following comments refer to *upper limits*, i.e. limit values which, when exceeded, result in a not O.K. evaluation. This is, by far, the most common case. However, the basic principles are the same for measured values which are tested against a lower limit or for deviation from a target value.

How Limits Are Generated

Every limit value is generated by combining learned values with fixed settings from the parameter data base.

“Learning” consists of calculating the mean and the standard deviation (variance) for the measured value – refer also to the following section "How

³ However, it is possible to set up the TasAlyser in a way that an error is reported if one of the essential test steps have not been measured or if measurement values are missing.



Limits Are Learned". Using mean and standard deviation, the learned limit is calculated as follows:

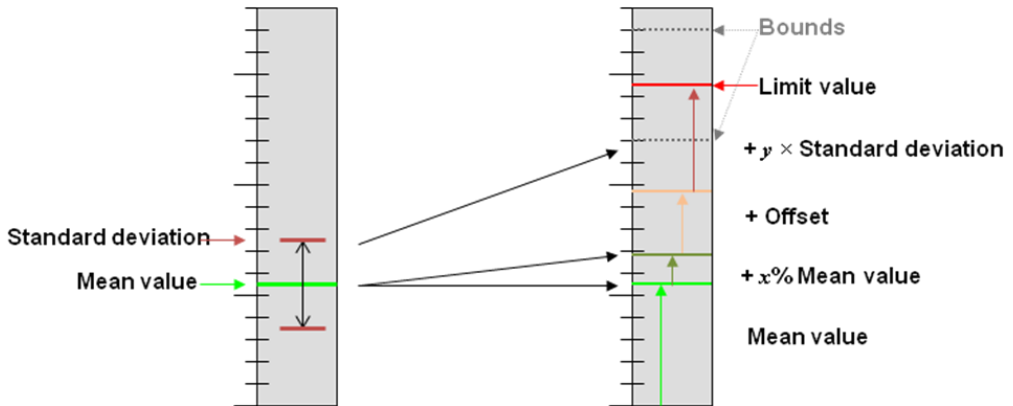
Limit value = *base value* ("offset") + mean + $x\%$ mean + *factor* × std. deviation

The numbers for *offset*, x and *factor* are set in the parameter database. An example: A mean of 77.5 and a standard deviation of 2.8 with the parameters *offset* = 5, x = 10% and *factor* = 3 results in:

$$\text{Limit value} = 5 + 77.5 + 7.75 + 3 \times 2.8 = 98.65$$

In addition to *Offset*, x and *Factor* the parameter database also contains a lower and an upper bound for each limit value. The actual limit value is not permitted to lie outside these bounds. If, in the above example, a lower bound of 100 and an upper one of 120 are entered in the parameter database, then the limit value used is 100 and not just 98.65.

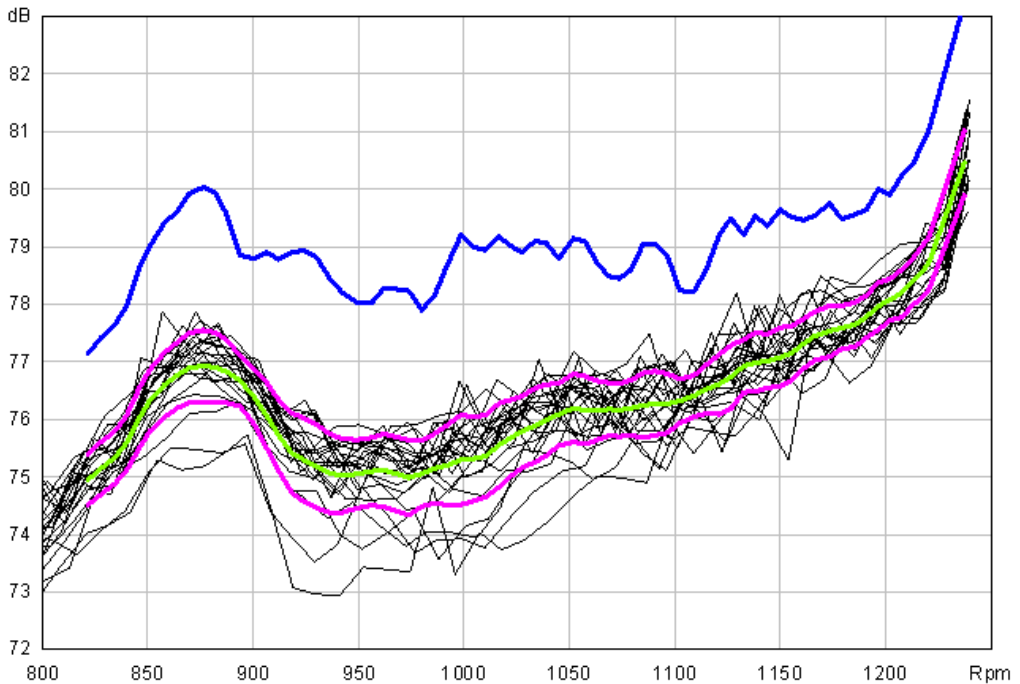
The following diagram illustrates the generation of the limit value:



If the lower and the upper bound are set equal in the parameter database, then learning is completely overdriven: the limit value will be fixed to the bound.

The description above refers to a single value. For spectra and curves each point of the curve is learned separately. (This means: mean and standard deviation are calculated for each point). *Offset*, x and *factor* are values which are valid for the whole curve. The lower and upper bounds are again curves, which are defined in the database as polygons. They are known as minimum and maximum polygons. It is also true here that if minimum polygon = maximum polygon (also possible in sections), then the polygons determine a fixed limit curve in this range.

The figure below shows a number of measurements (black), their mean value (green), the band ± 1 x standard deviation (magenta) and a possible learned limit from mean + 5 x standard deviation (blue).



Spectral Values in Spectral Limits

The Rotas noise analysis system has an extra feature for spectral limit curves: the *spectral values*. It can be specified in the database that for the characteristic frequencies (orders, see also “Frequency, Order, Harmonics” on pages 30ff.) of order sources separate single values (such as “meshing order of gear wheel A”) are extracted, evaluated and learned. The limits of these single values can also be set as fixed values (by means of minimum bound = maximum bound), although the rest of the spectrum is learned normally. The learned limit curve is turned off at the positions of these “spectral values”.

The limits of these spectral single values are inserted into the (otherwise) learned limit curve, which then shows these “hats” at the corresponding positions.

The reason for these spectral values is that the behaviour of order sources (concerning standard deviation) is different to the rest of the spectrum. This is the reason why you want to specify independant limits for their frequencies in the spectrum. Sometimes, you even want only to turn off evaluation at these positions (e.g. when the frequency of an order source shows up in the spectrum of a shaft where it does not belong to). Order sources usually appear in more than one synchronous spectrum (which



belong to the two or more meshing wheels). The noise level should be the same in each spectrum, therefore you do not need to set tight limits in each spectrum. You set the tight limit for only one spectrum and set high limit for the others to avoid double evaluation.

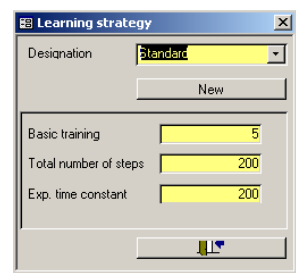
Since the positions of these frequencies in the spectrum depend on the aggregate type (especially on the number of teeth for the components), it is not possible to simply integrate the spectral values in the minimum and maximum polygons. Instead, the TasAlyser takes over the task of calculating the correct frequencies and positioning the spectral values accordingly, depending on aggregate type and order source.

The “hats” use the single value instrument “spectral value” in the parameter and in the result database.

How Limits Are Learned

For calculating the mean and the standard deviation, which correspond to learned limits, a number of measurements are necessary. What does the TasAlyser do if the first aggregate of a type is waiting to be tested?

Learning is divided into two phases: Base learning and additional learning. Base learning involves a small number of aggregates (5 to 20), additional learning a much larger number (e.g. 200). Both numbers are set in the parameter database as shown in the figure on the right.



During base learning the aggregates are evaluated using the upper bound from the parameter database. If one of the first aggregates tested is *very* loud, then it will already be identified as being n.O.K.

After base learning has been completed, the first learned limit is generated using the mean value and the standard deviation calculated from the measurements of these first aggregates. The additional learning phase begins with the next aggregate.

Each subsequent aggregate is tested against the learned limit valid at that moment. If the result is n.O.K., then the aggregate is segregated out. However, if the result is O.K. then the data of this gearbox contributes to the whole and a new limit is generated.

Measured gearboxes (after restart of learning)



As the number of tested gearboxes increases with each additional gearbox, mean and standard deviation become more stable. Once the maximal number



of aggregates has been reached, learning stops and the limits remain where they are. If you want, you can have neverending learning by entering the value “-1” for the maximal number of aggregates. Then the “additional learning” phase is never left.

The Time Constant

In addition to the number for base learning and the total learning number, you find a third number in the parameter database. This is the exponential time constant.

To be exact, the mean values are not calculated evenly over all the learned measurements. In contrast, the later (younger) measurements are weighted higher than older measurements.

The reason for this is that when testing more recent measurements there is a much better learned limit available than there was with earlier measurements. And, since the first aggregates during base learning were only tested against the maximal bounds, it could well be that these would not be O.K. when tested against the current limits – one would, therefore, like to lessen the influence of these first measurements.

The amount of weighting is determined using the time constant. The greater the time constant is when compared with the learning target, the more even will be the weighting of all the measurements. Small time constants weight the recent measurements more strongly.

Assume that you have a learning target of 200. With a time constant of likewise 200 the weighting of the first measurement compared to the last is only about 37%. With a time constant of 100 the weighting of the first measurement is about 14%, with a time constant of 500, however, 67%.

You can re-initiate learning at any time, either completely or selectively for only individual limit values. You will find more information in the chapter on Learn Parameters on page 50 ff.



Noise Analysis Theory

This section describes the “scientific background” to rotationally synchronous noise analysis. Therefore, it is less relevant for operating the TasAlyser or for setting limits at first. Nevertheless, if you want to understand what the individual measured values mean and how you can deduce the causes from these numbers, then you should read on.

Rotationally Synchronous Analysis

The precise error detection of the Rotas noise analysis is based essentially on the rotationally synchronous analysis of the noises. This makes it possible to extract the noise contributions of the aggregate’s different internal shafts and rotors from a sensor signal.

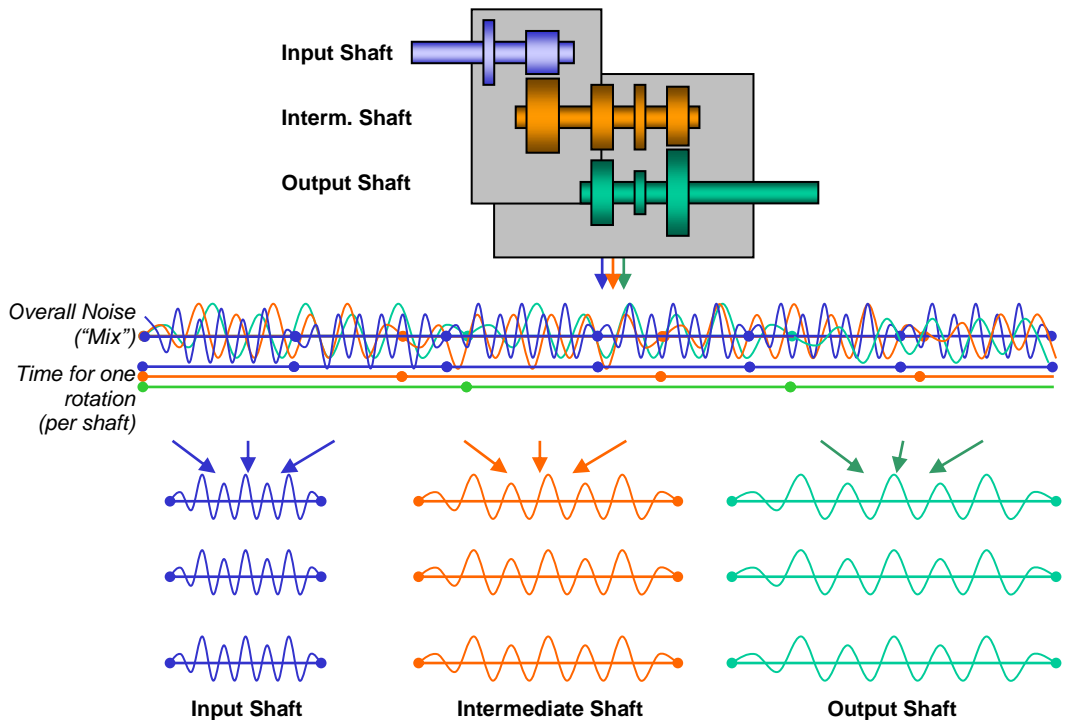
The parameter database contains construction data for all aggregate types. Given the necessary speed information (usually the speed of the input shaft), the TasAlyser is able to calculate the rotational speed of each gear wheel or rotor because it knows the internal transmission ratios.

From the relative rotational frequency of a rotor it can then be calculated how long a whole revolution will take at the current rotational speed of the reference shaft. For different rotors rotating at different speeds, the time taken for a revolution is also different. The TasAlyser cuts a copy of the total signal for each rotor into sections, which exactly cover one revolution of this rotor at a time.

Finally, after averaging over several revolutions of a rotor, a rotationally synchronous time signal is obtained, in which the noise components, which are not synchronous with this rotor (and therefore come from other rotors), are suppressed.



The diagram below illustrates graphically the principles of rotationally synchronous analysis:



Synchronous Channels and Mix

After the rotationally synchronous analysis step there are several copies of the sensor signal. In each case, these are synchronous to a rotor and are further analyzed in parallel and independently. These processing strands are known as *synchronous channels*.

Not all the noises in an aggregate are necessarily synchronous to a rotor in the construction (an example is bearing noise). Since these noises are not synchronous to a rotor, they are suppressed in all synchronous channels. In order to prevent these noises from escaping analysis, there is another processing channel, which is known as the *mix* channel. Although the input signal is also cut into blocks for this channel which contain one or several revolutions of a reference shaft, it contains all the noise components because the averaging is omitted for these blocks.

Optionally, a further processing channel is available, a *fixed frequency* or *fix channel*. This does not refer to the revolutions of a rotor, but has a fixed scanning rate. Fixed channels are used, for example, to analyze background noises (such as gear shift noises in gearboxes). Once more, to make this quite clear: all processing channels - synchronous channels, mix channel, fixed



channel - are processed copies of one sensor signal. If the noise analysis system is equipped with several sensors, then each sensor has its own set of synchronous channels, its own mix channel and, if necessary, its own fixed channel.

Crest & Co

The time domain analysis is the first step after calculating a synchronous (or mix) channel. In this step, different parameters are obtained from the signal of one revolution. The most important of these parameters are *RMS*, *Peak* and *Crest*.

The RMS value corresponds to the total energy of the signal – so to speak, the volume⁴. A high RMS value means that the aggregate is loud. If the RMS value of a synchronous channel is high, then the noise comes from this rotor. A higher RMS value in the mix channel suggests a generally loud aggregate or a cause beside the rotors. Typical RMS values lie between 1 and 10, depending upon the aggregate type and size, the rotational speed and other factors.

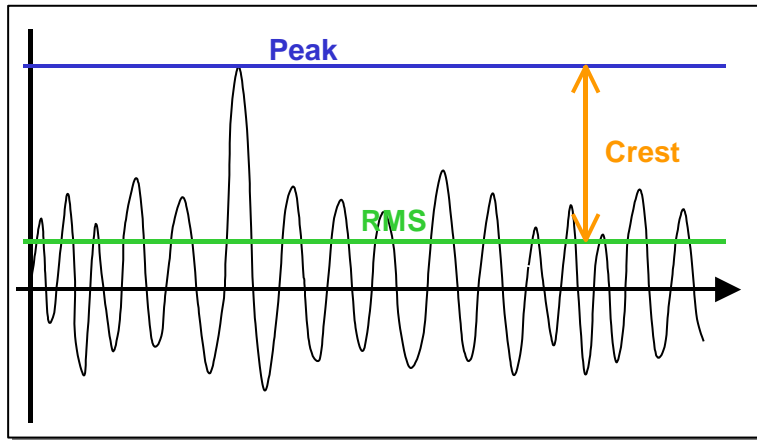
Occasionally the RMS value is also converted to the logarithmic dB scale, so that it is directly comparable with levels arising in the spectra. This value is known as the *master volume*

The Peak value is simply the highest occurring value, i.e. the signal spike. A single loud crashing noise during the measurement produces a high Peak value. In addition, a high Peak value will result if there is a clear “tick” during each revolution of a rotor. The signal spike will then simply appear more than once.

Because of this, the Peak value can hint to a nick of a rotor, such as a defective tooth in a gear wheel. The height of the signal spike, however, also depends on the background noise: an altogether louder aggregate or rotor (= higher RMS value, see above) will usually yield higher Peak values. On the other hand, the height of the signal spike does not necessarily have to increase with increasing rotational speed. All in all, the Peak value’s suitability for nick detection is limited.

The *Crest value* is much more reliable. It is calculated (for each revolution) as the ratio of peak value to mean value, i.e. Peak/ RMS:

⁴ technically speaking, volume and total energy are completely different things



The Crest value indicates how strongly the signal spike steps out of the ground noise. A high Crest value is, therefore, a very much clearer indication of “ticking” than a high peak value. Typical Crest values lie between 4 - 8, depending upon the aggregate type.

The Crest value is calculated separately for each rotor (synchronous channel). A high Crest value in a synchronous channel hints to a nick on one of the order sources (gear wheels) on this rotor.

The Kurtosis is related to the Crest value. It increases if the signal contains many spikes. In noise terms this corresponds to a “crackle”. Defective needle bearings, for example, can cause a crackling noise.

Frequency, Order, Harmonics

For each revolution, a spectrum is calculated from the (rotationally synchronous) time signal of each synchronous channel. (Occasionally you will see the term “FFT” = “Fast Fourier Transform”.) The characteristic frequencies of different order sources can be found in a spectrum. If the spectrum deviates from the (learned) standard, it is possible to deduce different defects from the kind of deviation.

If a time signal is processed directly in the spectral analysis, you get a frequency spectrum. If, for example, a distinctive component with 160 oscillations per second occurs in the time signal, then a line appears in the frequency spectrum at 160 Hz.

If, however, the spectral analysis is used on the rotationally synchronous time signal, then multiples of the revolution frequency are obtained instead of frequencies with the unit Hz. For example, if a distinctive component occurs with 16 oscillations per revolution in the time signal, then a line appears at 16, which means 16 times the rotational frequency or the 16th

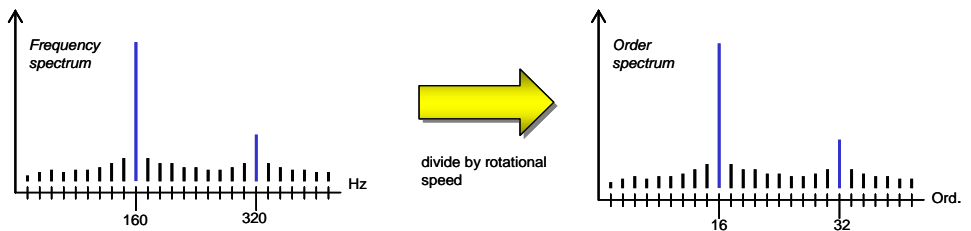


order. For this reason, the spectrum of the rotationally synchronous time signal is known as the *order spectrum*.

If the noise of a gear wheel with 16 teeth is analyzed, then 16 small “clicks” will be heard for each revolution when the teeth of the gear wheel mesh with the paired gear wheel. These 16 “clicks” produce a line in the order spectrum at the 16th order. This line is independent of the rotational frequency (rotational speed). It does not matter whether the gear wheel turns with 10 or 20 revolutions per second. The 16 “clicks” per revolution always remain and, therefore, the 16th order. That is different for the frequency in the frequency spectrum: with 10 revolutions per second the 16 “clicks” generate a frequency of 160 Hz, with 20 revolutions per second, however, 320 Hz.

This example demonstrates the advantage of the order spectrum compared with the frequency spectrum. The order spectrum is independent of the number of revolutions. You can assign the spectral components quite simply to the sources (like the 16th order to the 16 teeth of a the gear wheel).

Simple noise analysis systems produce an order spectrum by generating a frequency spectrum and then dividing the frequency axis by the rotational speed:



However, the rotationally synchronous analysis of the time signal in the Rotas system produces order spectra with very much finer resolution and can calculate a spectrum for each rotor. The result of the “simple” order analysis, however, can be compared to the “mix” channel of the Rotas system (see “Synchronous Channels and Mix” above).

Harmonics

As described in the example, the dominating noise sources are, in particular, meshings, i.e. the noise which occurs when the teeth of gear wheels engage. Similar to a guitar string, meshing does not produce a pure sine tone with only one frequency, but, like the musical instrument, consists of root and harmonics.

The root frequency or base order (e.g. the 16th order) in particular is found in the spectrum with its multiples (32nd, 48th, 64th order etc.). In the context of Rotas noise analysis we call the base order the “first harmonic” or “H1”, the double base order “second harmonic” and/or “H2”, etc.



Harmonics can be clearly recognized in a typical gear wheel spectrum. Whether or not H1 is higher than H2, or whether H4 is still clearly recognizable, depends on the geometry and surface of the particular gear wheel. This means that no general defaults (with regard to the limit values) can be set for harmonic patterns, instead the circumstances of the project must act as a guideline.

Beside meshing orders and harmonics, usually also *sidebands* appear. High sidebands can refer to eccentricity or ovality (see “**Fehler! Verweisquelle konnte nicht gefunden werden.**” below).

The instrument “Spectral Value”

On the whole, the spectrum shows the general characteristics of a noise. In addition, individual positions in the spectrum, particularly in an order spectrum, have special meaning and deliver important information on the component to be tested. Such positions include the harmonics and their sidebands, which have already been discussed, but other positions can also be significant depending on the aggregate.

The “spectral value” tool delivers a single value, which corresponds to the value of the spectrum at a certain position – thus, for example, the height of the first harmonic as a single value.

In reality, positions in the order spectrum, which correspond to parts of the meshing order (e.g. half gear mesh order) are sometimes conspicuous. Damaged or worn out grinding wheels in gear wheel manufacturing can literally “grind in” such peculiarities on a gear wheel. These are referred to as circular pitch errors.

Compared with the spectrum in general, the spectral value has the advantage that you can specify the limit separately and independantly (see “Spectral Values in Spectral Limits“ on page 24) as well as independant error codes. Furthermore, single values can be evaluated statistically more easily.

Moreover, the “spectral value” instrument is not limited to only extracting an individual order from the spectrum. It can also determine the maximum value of an order band, or the total energy of an order band.

Spectral Tracks (= Order Tracks)

All the measured variables looked at so far have one thing in common: during testing time (for instance, over the rotational speed ramp) the values are maximized, minimized or averaged (depending upon parameterization) and deliver a final result, e.g. a spectrum. What is usually ignored here is the measured variable’s course over rotational speed, time or torque. Often irregularities do not appear during the whole testing time, but only at specific



rotational speeds or under specific torque conditions. It then disappears if only one final value is generated over the whole testing time.

In order to fill this gap, different “tracking” instruments for peak, RMS, Crest, kurtosis, spectral value and spectra exist. These instruments are used to record the measured value’s course relative to its reference signal as a curve, allowing it to be evaluated. The spectral value’s course against the reference variable for example is known as the “order track”.

When spectra courses are recorded against a reference variable, the result is called a spectrogram. With regard to the data volume, spectrograms are the “heaviest” representation, but they show an quite exact picture of noise behavior during the test.

Second Level Instruments

The measure values covered up to now have been determined directly through averaging, minimizing, maximizing or other calculations performed directly on the data stream. The spectral value is the only exception here, because this can only be determined when the corresponding peak hold spectrum has been completed.

Instruments whose results are based on the processing of the results of other tools, are known as second level instruments. Beside the spectral value, there are other such second level instruments which can be processed only after another measure value has been calculated. The “curve interval” and the “curve polygon” belong to this category.

Both second level instruments need a tracking curve as input datum. From this curve, the curve interval calculates a single value for a specified section of the curve (maximum, minimum, mean). By applying the curve interval on a tracking curve, you have the option to divide the whole track into sections which show characteristic noise behaviour and calculate a single value for each section reflecting these characteristics. The advantage of this is that you can also evaluate track characteristics statistically (as you can for all single values).

The curve polygon is used to compare a curve with a polygon and then generate a characteristic value. In a simple case the minimum or the maximum can be determined within the polygon’s validity interval (similar to the curve interval). The new thing is that you can also calculate the area between the polygon and the curve. This kind of evaluation is carried out, for example, in the analysis of curves which represent gearshift force against displacement. The measure value, which is produced here, characterizes the gearshift work.

In parameterization, generation of limit values and evaluation, second level instruments do not differ from other instruments. The only important thing is

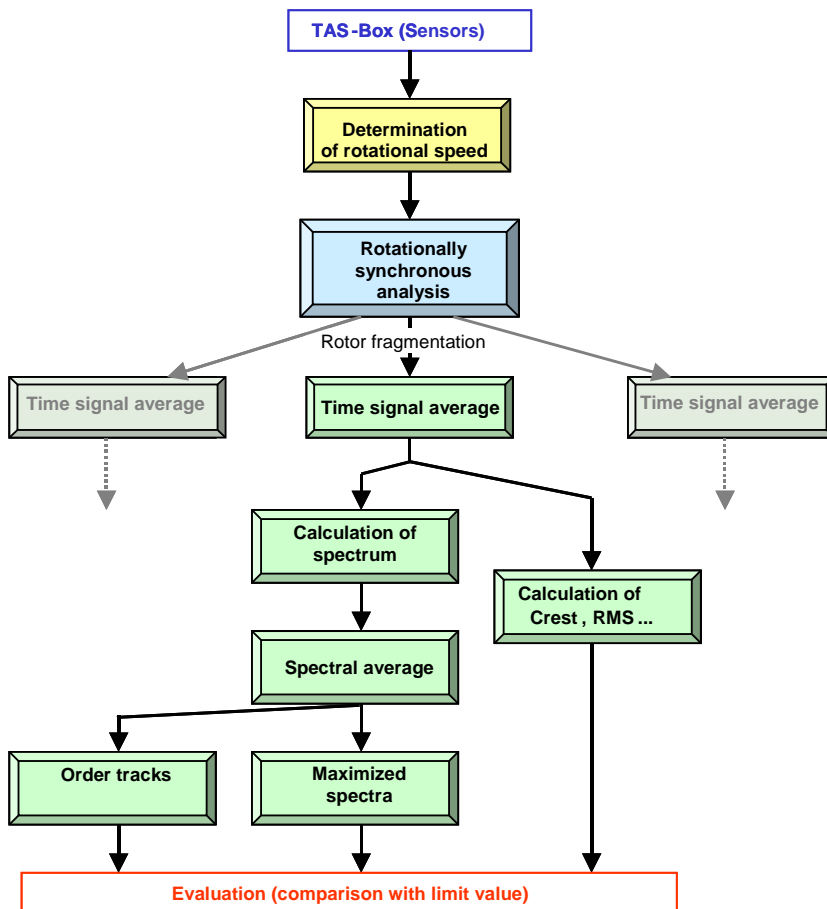


that in order to use a second level instrument (such as the curve interval), the source instrument (the measured value track) must be available as well.

Analysis Steps

A rough operational procedure in noise analysis has already emerged from the previous remarks: rotationally synchronous analysis and separation of synchronous channels, calculation of Crest & Co, spectral analysis, secondary instruments, etc.

The following flow diagram shows the typical steps in noise analysis:



Further analysis steps can, of course, occur, depending upon your aggregate or project. This diagram can serve as orientation and also as clarification of the three important sources of evaluable measured variables: single values from the time signal, maximized order spectra and order tracks over the measuring ramp.



The TasAlyser Program

The TasAlyser program, known simply as TasAlyser or measurement program, processes sensor signals, calculates the acoustic measure values from these signals and evaluates them against limits. The TasAlyser, therefore, carries out the actual noise analysis.

Depending on the measurement project and customer requirement, the TasAlyser is individually configurable with regard to the project's analysis components and the layout of the windows. This chapter presents a "typical" measurement project with the displays, windows and control functions that most often occur.

The Project Directory

The TasAlyser program opens a measurement project in the same way that you would open a Word document in the Microsoft Word program. As in Windows the TasAlyser program is usually installed in the folder `C:\Program Files (x86)\Discom`⁵. Without a measurement project, however, the TasAlyser is only an empty shell.

Unlike Word, a project is not, however, contained in only one file (and you cannot produce a new project as simply as a new Word document). Instead, a whole number of files belong to the project, which are all contained in a common project directory.

The project directory is usually a sub-directory of `C:\Discom\Measurement\...`, such as `C:\Discom\Measurement\MultiRot\MyProject`. The project directory contains a set of sub-folders and usually also a link to start the TasAlyser with this measurement project.

If you do not know where the measurement project directory is stored, you can locate it using the **Project Directory** instruction in the **File** menu of the TasAlyser program.

Test Cycle, Command Center

The picture below shows the "Command center" window. The section "The Test Cycle" on page 20 describes how a test cycle is divided into several test steps. The *command center* window displays which aggregate type is being tested and the test step in which it is being tested at that moment in time.

⁵ The exact installation path can be determined via the environment variable `%DiscomSoftwareRoot%`.



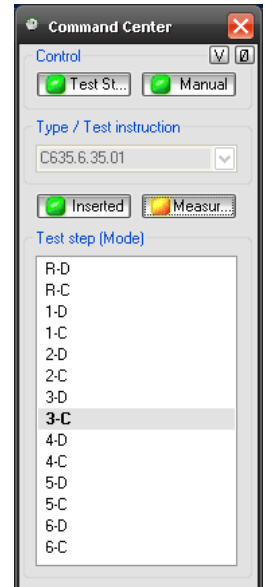
The *command center* window has another visual representation than shown above (see below) in which all the intended test steps are displayed as a list:

The display format can be changed using the small button with the \emptyset symbol at the upper right hand corner.

In its large variant, the command center window can also be used to emit commands to the measurement program (“manual operation”), hence the name “command center“.

In the menu under **Type/Test instruction**, you first select the aggregate type to be tested, and press **Insert**. Then you select a test step in the list. To start the measurement, you press **Measurement** and to terminate the measurement, you press **Measurement** again.

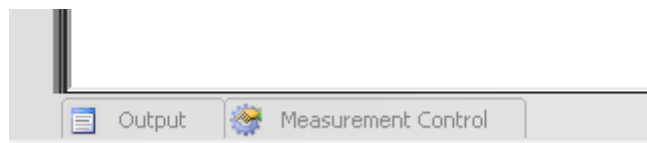
While the test bench control is sending commands, you can do this manually at the same time. This means that you can complete incomplete test cycles, for example, during maintenance or start-up, or supplement missing test bench commands. During normal test operations it is better to deactivate this function by turning off the **Manual** button and revert the command center to its small variant (button \emptyset).



Docking Windows

In addition to the various display windows, there are some docking windows. These are not used to display measured values or results, but for operating the measurement program.

The docking windows **Output** and **Measurement Control** can be found at the bottom border of the program’s main window:



Docking windows are normally hidden except for a tab. To unhide the docking window either move the mouse to the tab and wait, or click the tab. If a docking window is unhidden, you will find control elements at the upper right corner: Via these elements you can control the behavior of the docking window. In particular, by using the “pin” (middle symbol) you can “pin down” the docking window to prevent it to be hidden automatically when you click outside of the window.





Output

Messages and status reports appear in the **Output** docking window. The output window is divided into several sections (by appropriate tabs on the bottom window border). In **Communication** you can see, for example, a log of the control commands, which are being exchanged between test bench and measurement program.

Measurement Sequence Control

This window contains some large buttons via which the test cycle can be controlled. These large buttons are used if the measurement program is used in mobile measurement (e.g. during a car drive) or runs on a computer with Touch screen.

System Configuration

The tab for the **System Configuration** docking window is located at the left-hand side of the program's main window, together with the tab for the **Favorites** window.

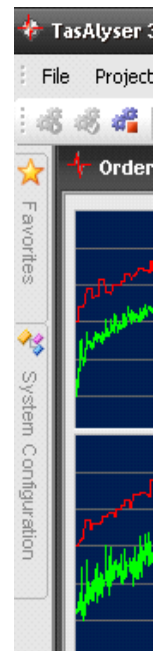
The measurement program is composed of a large number of individual software modules. Most of these modules are working in the background, and, normally, you do not have to bother about them. If, however, for some reason you do need to, then the configuration window permits access to each individual software module.

Favorites

The most important software modules are contained in the favorites window. Here, for example, you will find the module for the order spectra scope, the report window or the wave file recorder. If you have closed the Scope window, for example, and then want to open it again, simply drop down the favorites and double-click on the appropriate entry.

You can add any module from the system configuration to the favorites and sort the list of favorites using the buttons in the Favorites window's toolbar.

If you have accidentally closed one of the docking windows (not just hidden), and then want to re-open it, you can do this using the **View** menu of the measurement program. The docking windows are listed in the **Tool Windows** submenu.

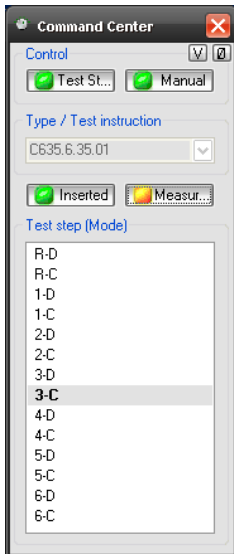





Test Bench Connection

In a test bench environment the TasAlyser is controlled by instructions from the test bench control (see “The Measurement Computer” and “Communication with the Test Bench “ on page 13ff.). In most cases, measurement computers and test bench will be connected by a classical serial line, although a connection over network using appropriate protocols (UDP, TCP/IP) is also possible. The measurement computer can be equipped with a profibus interface card, or even communicate with the test bench using a “low level” parallel bit interface.


The advantage of instruction-based communication is that, on the one hand, it can be monitored and understood easily, and, on the other hand, it is relatively simple to extend the instruction set. With bit-based communication it must always be kept in mind which bit stands for what. Adding more instructions proves to be quite difficult, due to a lack of bits.



All kinds of test bench connections are translated within the TasAlyser into internal standard commands. This task is performed by a decoder module, which can be equipped with supplements, “plug-ins”, in order to implement additional instructions. Communication with the test bench is logged in the **Output** window, section **Communication**, (see “Docking Windows” on page 36).

The interface settings can be accessed in the system configuration (see “System Configuration” on page 37). Here you can see a tree diagram of all the modules in the measurement program. Open the section **Evaluation** and in it the node **Control**  **Serial Interface** **Center** (or **Command Center**). Here you will find one or more interface modules, for example, a module designated **Serial Interface**. Open the setting dialog by double-clicking on the icon in the system configuration tree.

If communication is functioning normally, you can observe the test steps during a test cycle, the aggregate type being loaded, the test steps being activated, etc., in the command center window. Note: Here in the command center you can deactivate test bench control! (In which case the button at the upper left is red.)

If you have closed the window of the command center, you can re-open it by double-clicking on the command center entry in the system tree (You have already met this entry as **Command Center**  **Command Center** super-module of the interface modules.) In addition, you can usually find the command center under the favorites.



Wave Audio Recording and Playback

The TasAlyser can record complete test cycles or concatenated cuts as Wave files. Control commands, such as test step selection, are embedded in the Wave file, so that later, during playback with the TasAlyser, not only is the noise signal reproduced, but the complete test cycle is repeated. The modules for recording (*Recorder*) and playback (*Player*) are found in the favorites (or in **Source** in the system configuration):



Wave Recorder

The Wave recorder writes the complete signal current of all sensors and channels into a file in WAV format. In addition, information about the channels, such as calibration data, is filed in the header data (the meta data) so that, when read with the Wave player, the signal current is restored exactly the same as when it originally came from the TAS Box. The wave recordings can also be opened in the TasWavEditor application or in third-party tools like “Audacity”.

Maximum Directory Size

In the Wave Player options, “Files” section, you can specify a maximum directory size. The Wave recorder ensures that the total amount of Wave files in the target directory does not exceed the indicated size in gigabytes. If necessary, the Wave recorder will delete the oldest files, if the directory becomes too large.

If you activate the option **Subdirectory for NOK/OK measurements** and give a name for this subdirectory, all “not OK” measurements are stored in the according directory and the “OK” ones in the other. The maximum directory size is checked for these folders separately, meaning that the subdirectories together can then reach twice the indicated size.

Wave Player

The Wave player complements the Wave recorder: it plays back the recorded Wave files and, at the same time, reproduces (if activated) recorded test cycle events, such as a test step change.

Read more about the wave recorder and wave playback options in the TasWavEditor quick start manual.

Parameter Administration with TasForms

The Rotas System uses a database-supported parameter administration. Without database, the Rotas System cannot work. Limit value administration, in particular, takes place in the database (c.f. section “How Limits Are Generated” on page 22). Therefore, it is important to be acquainted with the basic concepts of parameter administration.

This chapter shows how new aggregate types are created, existing ones administered, and also how to set limit values.

The Database in the Overall System

Database and User Interface

The parameter database is an Access database and consists of a single file (file ending .mdb), which contains the parameter tables. To set the parameters, you do not open the database itself (which is possible with Access) but make use of a user interface. This chapter describes this “TasForms” user interface – which is also an Access database.

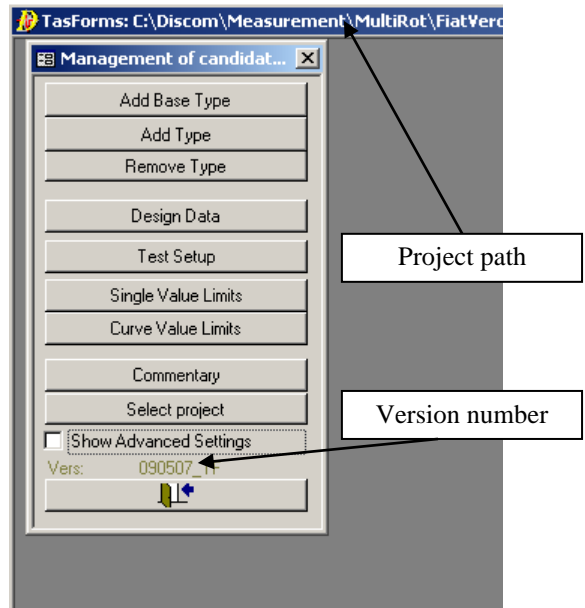
Starting TasForms

A link, symbolized by a yellow “D”, is set up on the desktop to start TasForms:



This link is either to be found on the desktop or in a folder on the desktop with the name “Rotas for Experts”. A double-click on the symbol opens the parameter administration with the start-up screen shown on the right:

In the header of the main window you can see the path locating the parameter database file which is currently being accessed. The smaller window consists almost exclusively of buttons which open further windows for data



processing. Above the exit button with which you can leave the program, the version number is given. If you have questions on parameter administration, you need to know the program version. This avoids any misunderstandings, should there be newer versions with changed functionality.

With the checkbox “Show Advanced Settings” you can extend the list of selection buttons. Since these settings assume advanced knowledge, they will not be discussed in detail at this early stage.

Creating and Erasing a Type

New Type or New Base Type?

A frequent task in connection with the parameterization of the noise analysis system is the creation of a new type. As already discussed in section “Type and Base Type” on page 18, small modifications of the gearbox housing can cause the test bench to transmit a new type identification. Completely new gearboxes for a new vehicle platform are also common in reality.

The reason for a new type is not important: the first question to ask is whether or not a new base type is required for the new type. From measurement system’s point of view, a new base type is necessary if, among the existing base types, none uses exactly the same order sources with the same base orders (numbers of teeth). If the new type is identical to an existing type concerning number of teeth, then, usually, only a new type name for this base type is needed. In exceptional cases, it may be that you want to create a new base type, although the new type has the same number of teeth as an existing base type. This will be the case when you expect the new type to have major acoustic differences to the existing base type.

It is also possible to convert a type, which was only a name up to now, into an own base type later on.

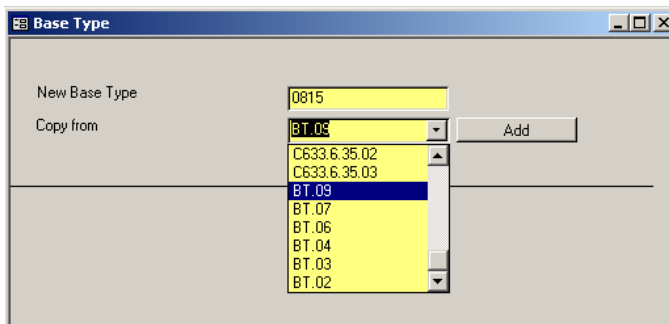
Names

The names of types, base types (and also of all other designatable database objects) can contain letters and numbers as well as hyphens, underscores and periods. Spaces and colons are not allowed; upper and lower cases are differentiated.

Creating a New Base Type

Let’s first assume that the new type differs in the number of teeth from all existing base types. This means that we need to create a new base type. The appropriate function is accessed using the upper button in the start form: **Add New Base Type**.

This opens the form shown below:



In the input field “New Base Type” you can enter a name for the base type. In this example, the base types are all designated with names like BT.xx. The new

base type is now to receive the name 0815.

Since it is not a good idea to start with a completely empty data record for the new base type, you are forced to select an existing base type as copy source. Its data will then be taken over for the new base type. You should select a base type which is as similar as possible to the new base type for this process. The more similar the two types are, the less the new base type needs to be adapted afterwards.

When you have chosen an existing base type and selected it in the list, then click on the **Add** button. The new base type is created and the existing base type’s data is copied. The program sends a message, **Done**, when the copy action has been finished. After the message has been confirmed with **OK**, the message window and the input form for the new base type will close.

Since a base type alone is not enough to make the data accessible for the measurement program, a type name is also created for the base type at the same time (i.e., to base type 0815, a type 0815 also exists immediately, which is assigned to that base type). Since it is possible that the name of the base type already exists as type name for another base type, this action can fail. In this case, the base type has been created but is not accessible over a type name at that time. In particular, the measurement program cannot access this base type. The problem is solved when a new type is created and assigned to this base type.

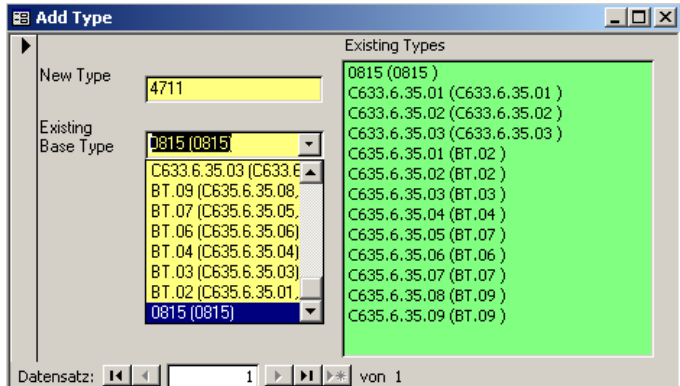
Please note that the form shown above is not only used by the program to create a new base type, but also when other new objects need to be added. Behavior and operation are then similar to the above.

If you create other objects, it can be useful to change the option “Copy all connected data” to “Copy only basic information”. If you change this option, only the really basic informations are copied from the copy source, but not, for example, connected measurement values. By default, the option “Copy all” is selected. Then *all* data connected to the copy source are also copied.

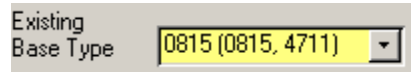
Creating a New Type

If an existing base type can be used for the new type, then you only need to assign the new type to this base type. The appropriate function can be reached using the button **Add New Type** in the parameter administration start form. Clicking this button opens the following form:

The input field and the selection field on the left hand side resemble the form for creating a new base type. In addition, there is a list on the right hand side which shows the existing types with assigned base type in brackets.



The procedure for creating a new type is very simple now: Enter the new type in the input field, select the base type from the list (the types already assigned are listed in brackets after the base type), and press the **Add Type** button. If the action is successful, the new type appears in the list of the existing types. In addition the new type is supplemented in the selection list in the brackets behind the base type.

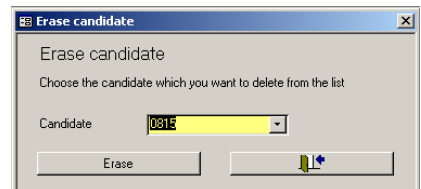


This action can fail if the name for the **New Type** already exists. The program will then point this out with the appropriate error message.

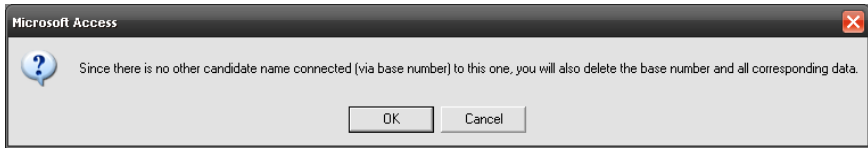
Erasing Types and Base Types

Occasionally, aggregates are designed for production but then not built as planned. If types have already been created for these aggregates you want to get rid of them. The appropriate function is reached using the **Erase** button in the parameter administration start form. Pressing this button, opens the form:

With the help of the selection list, select a type (not a base type!). Clicking on **Erase** removes this type from the parameter database and it then becomes unknown, especially in the measurement program.



If the type to be erased is the last and only type which is assigned to a specific base type, then the following message appears:

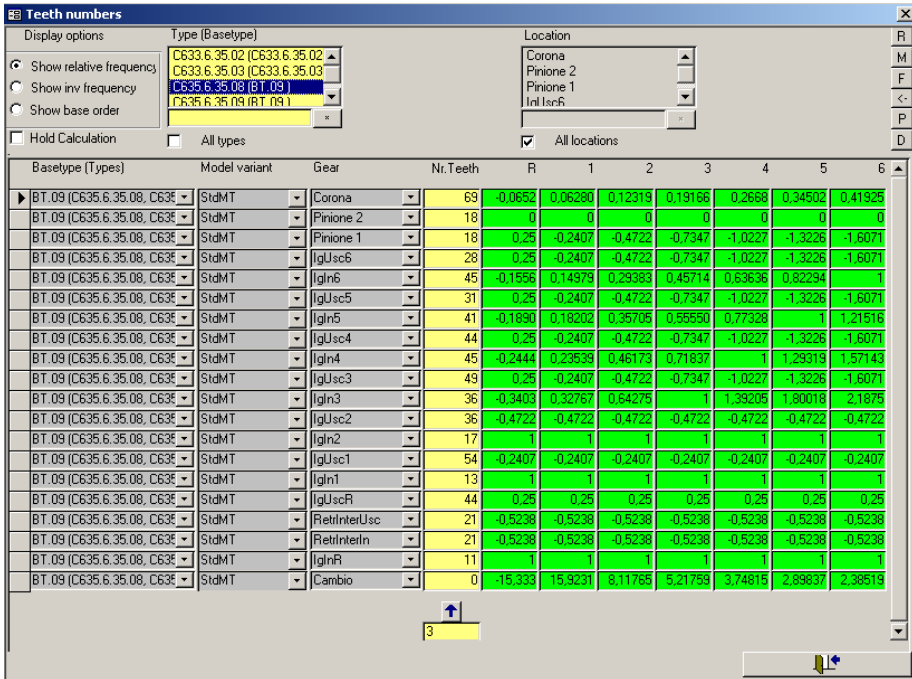


If this prompt is confirmed with **OK** then the type *and* its base type will be erased. Any settings which have been made for this base type will then be lost (limits, number of teeth, etc.) If the prompt is confirmed with **Cancel** then nothing happens. Both the type and the base type remain in the database.

This behavior makes sense since usually you do not want parameters in the database which cannot be accessed by the measurement program. However, as described above, this can occur when creating a base type. In order to get rid of such “invisible” base types, you first have to create a type name (see Creating a New Type), after which you can erase it with the “Erase Type” form.

Changing the Number of Teeth

One of the first things that needs to be done after creating a new base type is to modify the number of teeth, since the new base type usually has - at least some - other teeth numbers. The appropriate function is accessed by clicking on the **Construction Data** button in the parameter administration start form. This opens a variant of the following form:



Basetype (Types)	Model variant	Gear	Nr. Teeth	R	1	2	3	4	5	6
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	Corona	69	-0.0652	0.06280	0.12319	0.19166	0.2668	0.34502	0.41925
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	Pinione 2	18	0	0	0	0	0	0	0
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	Pinione 1	18	0.25	-0.2407	-0.4722	-0.7347	-1.0227	-1.3226	-1.6071
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgUsc6	28	0.25	-0.2407	-0.4722	-0.7347	-1.0227	-1.3226	-1.6071
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgIn6	45	-0.1556	0.14979	0.29383	0.45714	0.63636	0.82294	1
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgUsc5	31	0.25	-0.2407	-0.4722	-0.7347	-1.0227	-1.3226	-1.6071
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgIn5	41	-0.1990	0.18202	0.35705	0.55550	0.77328	1	1.21516
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgUsc4	44	0.25	-0.2407	-0.4722	-0.7347	-1.0227	-1.3226	-1.6071
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgIn4	45	-0.2444	0.23538	0.46173	0.71837	1	1.29319	1.57143
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgUsc3	49	0.25	-0.2407	-0.4722	-0.7347	-1.0227	-1.3226	-1.6071
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgIn3	36	-0.3403	0.32767	0.64275	1	1.39205	1.80019	2.1875
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgUsc2	36	-0.4722	-0.4722	-0.4722	-0.4722	-0.4722	-0.4722	-0.4722
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgIn2	17	1	1	1	1	1	1	1
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgUsc1	54	-0.2407	-0.2407	-0.2407	-0.2407	-0.2407	-0.2407	-0.2407
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgIn1	13	1	1	1	1	1	1	1
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgUscR	44	0.25	0.25	0.25	0.25	0.25	0.25	0.25
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	RetInterUsc	21	-0.5238	-0.5238	-0.5238	-0.5238	-0.5238	-0.5238	-0.5238
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	RetInterIn	21	-0.5238	-0.5238	-0.5238	-0.5238	-0.5238	-0.5238	-0.5238
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	IgInR	11	1	1	1	1	1	1	1
BT.09 (C635.6.35.08, C635.6.35.08, C635.6.35.08)	StdMT	Cambio	0	-15.333	-15.3231	8.11769	5.21799	-3.74819	2.89937	-2.38519

The order sources specified in the “Gear” column depend on the respective project, as do the columns “R”, “1”, “2”, etc. for the physical gears of a gearbox.

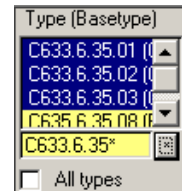
In the “Type (Basetype)” column you must select one or more base types in order to display the appropriate construction data.

In the form above only the column with the number of teeth is open for input. The relative frequencies of the gear wheels for each gear are displayed in the green fields for information. The frequencies are given with reference to a “calculation” speed of 1. This calculation speed of 1 often corresponds with the input or output speed of the gearbox. With the gearbox in the above screen shot the relative frequencies are given with reference to the gearbox input speed.

General Form Functions

Operating the Key Selections Fields

With the help of the key selection fields, you can make a selection in a number of ways. The effect of the check box **All Types** is clear: a check mark here deactivates all the other possibilities of the corresponding key selection field and the data area shows all data records without restriction for that key field.



If only individual entries are required, then no checkmark is set at “all types”. Subsequently, one or more entries in the list can be marked (several at one time by pressing the Ctrl or Shift key, as usual in Windows) or with the help of the input field and the “*” button. In the above example the asterisk button was used to mark all the entries in the list which fitted the format “C633.6.35*”. To specify the format, the usual substitute symbols “*” and “?” can be used. (For MS Access Profi: The asterisk button calls up the “Like” function (SQL).)

Valid for all setting buttons in selection lists: If the expected result does not occur with the first click then press the button again. Under certain circumstances, which are not clear, MS-Access needs a second trigger.

Changing the Entries of a Whole Column

Often one would like to change all the entries of a column at one go. This function is provided by the field on the right, whose function is quite simple: You enter something, click the arrow button and the corresponding column of the current selection is filled with this value.



If the column contains a numerical value, an auxiliary function provides the option of changing values *relatively*. That means that the column is not filled with a fixed value, but calculated with the existing value. If you want to do this, you enter the arithmetic operation which you want to execute in the input field. The following options are available: +X (add value X), -X (subtract value X), *X (multiply value by X). If you want to subtract a value using “-X” then the program will query to specify more exactly which is the action to be carried out because the minus sign can mean either the arithmetic operator or a prefix operator.

Sorting the View

It is not always sufficient to restrict the display to just specific data. Sometimes you also want to see sorted data. In order to do this, you can use an MS Access function: Click with the right mouse button in any field of the column to be sorted. Beside other available Access functions, entries can be

sorted in ascending or descending order. This function is available for all columns, whether key columns or data columns.

Copying, Printing, Comparing

There are six buttons on the right-hand side of the control area, which have been mentioned briefly above. With their assistance some powerful functions can be initiated, among which are copying, printing and comparing data.

The button **P** (for print) is used to print the current selection (in landscape format). Otherwise, the lettering of the buttons is analogous to the keys of a pocket calculator: With **M** a selection is memorized, and with **R** restored (press twice, if necessary).

With many forms a further form is opened when the **M** button is pressed. This form can also be opened separately using the **F** button (field selection). Here you can select different column fields e.g. for copying. Only the data of the selected columns are then copied.

With the **<-** button the copying action is finally performed as follows: The data corresponding to the memorized (using **M**) selection are read and inserted into the current selection (if possible).

The comparison function, which is initiated with the **D** button (difference) operates in a similar way. It compares the memorized data with the data of the current selection and shows the differences. With large data sets and low computing power this is to be used with caution, since the query which needs to be executed is quite complicated.

To show that the data area does not represent the data of the current selection, the control area changes its background color to brown. In this mode all the differences found between the selections are displayed.

Pressing the **D** button again, changes the color to purple. The lettering on the **M** and **D** buttons also changes to purple, in order to show that only the data records are now indicated, which are present in both selections and have different entries.

A further click on **D** displays background and **M** button in pink. In this mode the data area shows all data records which are in the memorized selection, but not in the current selection.

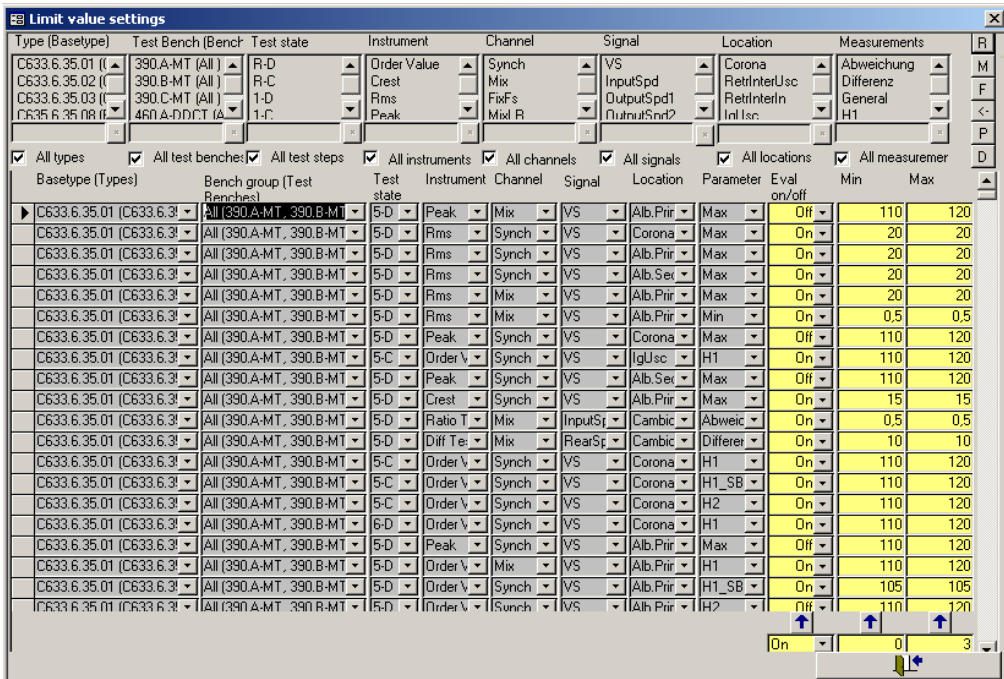
Finally, the reverse situation (all data records, which are in the current selection, but not in the memorized) is obtained with another click on **D**. Here the color code is a turquoise background and **D** lettering. Another click on **D**? No problem, we start again with brown...

Setting Limits

As already discussed in the section “How Limits Are Generated“ on page 22, limit values are generated by a combination of learned data and fixed defaults. In the parameter database, therefore, there is no direct setting of limits but only the rules for generating them.

Limits for Single Values

The limits for single values can be set using the following form, opened using the **Limit Single Value** button in the parameter administration start form. This form and the form for parameterizing the limit curves (see below) are the ones which have the most key selection fields in the control area.



Type (Basetype)	Test Bench (Bench)	Test state	Instrument	Channel	Signal	Location	Measurements	R	
C633.6.35.01	390.A-MT (All)	R-D	Order Value	Synch	VS	Corona	Abweichung	M	
C633.6.35.02	390.B-MT (All)	R-C	Crest	Mix	InputSpd	RetriInterUsc	Differenz	F	
C633.6.35.03	390.C-MT (All)	1-D	Rms	FixFs	OutputSpd1	RetriInterIn	General	P	
C633.6.35.04	460.A-DDCT (A)	1-C	Peak	Mix R	OutputSpd2	Inl. Isc.	H1	D	
<input checked="" type="checkbox"/> All types <input checked="" type="checkbox"/> All test benches <input checked="" type="checkbox"/> All test steps <input checked="" type="checkbox"/> All instruments <input checked="" type="checkbox"/> All channels <input checked="" type="checkbox"/> All signals <input checked="" type="checkbox"/> All locations <input checked="" type="checkbox"/> All measurer									
Basetype (Types)	Bench group (Test Benches)	Test state	Instrument	Channel	Signal	Location	Parameter Eval on/off	Min	Max
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Peak	Mix	VS	Alb.Priir	Max	Off	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Rms	Synch	VS	Corona	Max	On	20 20
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Rms	Synch	VS	Alb.Priir	Max	On	20 20
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Rms	Synch	VS	Alb.Ser	Max	On	20 20
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Rms	Mix	VS	Alb.Priir	Max	On	20 20
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Rms	Mix	VS	Alb.Priir	Min	On	0.5 0.5
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Peak	Synch	VS	Corona	Max	Off	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-C	Order V	Synch	VS	IgUsc	H1	On	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Peak	Synch	VS	Alb.Ser	Max	Off	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Crest	Synch	VS	Alb.Priir	Max	On	15 15
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Ratio T	Mix	InputSp	Cambic	Abweic	On	0.5 0.5
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Diff Te	Mix	RearSp	Cambic	Differer	On	10 10
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-C	Order V	Synch	VS	Corona	H1	On	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-C	Order V	Synch	VS	Corona	H1_SB	On	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-C	Order V	Synch	VS	Corona	H2	On	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	6-D	Order V	Synch	VS	Corona	H1	On	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Peak	Synch	VS	Alb.Priir	Max	Off	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Order V	Mix	VS	Alb.Priir	H1	On	110 120
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Order V	Synch	VS	Alb.Priir	H1_SB	On	105 105
C633.6.35.01	All (390.A-MT, 390.B-MT)	5-D	Order V	Synch	VS	Alb.Priir	H2	On	110 120

Whether or not evaluation should be carried out or what limits should be valid for learning (see above) is parameterized here as data (yellow fields). If the upper and lower limits are set as equal then learning is, to all intents and purposes, switched off.

Limit Curves

The 'Limit Curve Settings' dialog box contains a table with the following columns: Type [Basetype], Test Bench (Benc), Test state, Instrument, Channel, Signal, Location, and Measurements. Below the table, there are checkboxes for 'All types', 'All test benches', 'All test steps', 'All instruments', 'All channels', 'All signals', 'All locations', and 'All measurements'. A detailed list of limit curves follows, with columns for Basetype (Types), Bench group (Test Benches), Test state, Instrument, Channel, Signal, Location, Parameter, Eval on/off, Min, and Max. The 'Datensatz:' field shows '1' and 'von 966'. A 'Polygons...' button is located at the bottom left.

The form for limit curves looks almost exactly like the one for single value limits. The only difference is that no individual values for learning delimitation can be entered here, only a polygonal traverse. In order to modify or create polygons, you must first select the instrument for which the polygon is valid in the appropriate key selection field (because of differing units in the x -axis of different instruments). This activates the **Polygons** button, located in the lower left corner of the form. Clicking on the button opens the polygon administration form. Similar to the lists mentioned above, the polygons are also parameters, which are independent of type and bench. They only have meaning for a test when they are used in the limit curves form.

Defining Polygons

The following form shows, as an example, the settings for the polygon “StdMinSpectrum“. This is defined for the spectrum evaluation instrument.

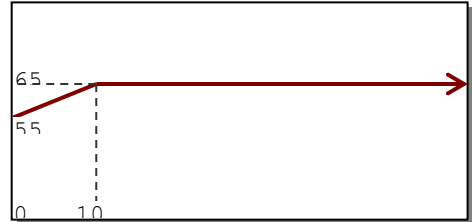
The settings are to be read as follows. In each case, a line with **X** and **Y** values belong together. The sequence depends on the **X**-

The 'Polygon settings' dialog box has a 'List name' dropdown set to 'StdMinSpectrum'. It includes 'New' and 'Delete' buttons. Below is a 'Location' dropdown. A table with 'X' and 'Y' columns contains the following data:

X	Y
80	80
10000	80
*	0

values (the smallest **X**-value is always at the top). The polygon is created in the measurement program by linearly connecting the registered bases according to this sequence. In the above example a horizontal line is defined as a polygon, which has the value 65 between the **X**-values 0 and 10000 (inclusively).

To clarify what we mean with the linear connection, we modify the above polygon as follows (**X/Y** values are noted in pairs): (0/55), (10/65), (10000/65). This polygon begins with **X** = 0 and **Y** = 55, rises at **X** = 10 to **Y** = 65 and then continues horizontally further.



Learn Parameters

Each measured value has its individual learned limit, and also its individual learning strategy. These are set in the Learn Parameter form:

Type (Basetype)	Test Bench (Bench)	Test state	Instrument	Channel	Signal	Location	Measurements
5GearProto (1)	SAIC-JAC	1-C	Command/Varial	FixFs	EXT	Counter Shaft	Abweichung
6GearProto (2)		1-D	Command/Varial	Mix	Speed	CS Gear	Differenz
		2-C	Crest	Mod	Time	Input Gear	General
		2-D	Crest Track	SPS	Vibration	Input Shaft	H1

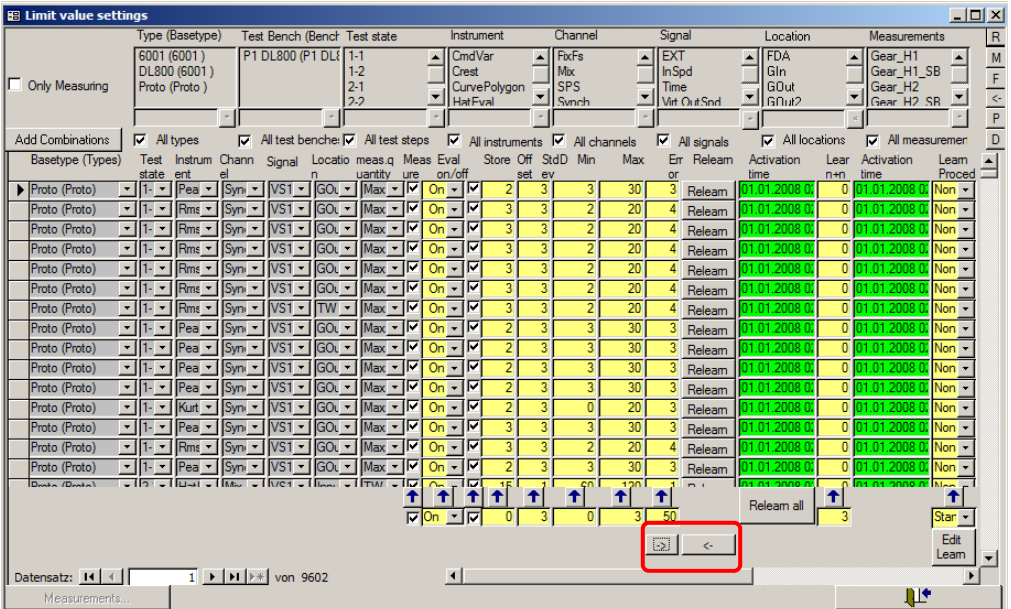
Basetype (Types)	Test state	Instrument	Channel	Signal	Location	meas. quantity	Learn	Relearn	Activation time	Learn+	Activation time
1 (5GearProto)	2-D	Order	Synch	Vibrati	CS Ge	H3	Stand:	(Relearn)	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	MS Ge	H1_SE	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	MS Ge	H2	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	MS Ge	H2_SE	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	MS Ge	H3	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	MS Ge	H3_SE	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	MS Ge	H4	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	CS Ge	H1	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	CS Ge	H1_SE	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Mix	Vibrati	CS Ge	H3	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	CS Ge	H2_SE	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	Secon	H3_SE	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	CS Ge	H3_SE	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Synch	Vibrati	CS Ge	H4	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Mix	Vibrati	Input	H1	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00
1 (5GearProto)	2-D	Order	Mix	Vibrati	Input	H2	Stand:	Relearn	15.06.2011 11:27	0	01.01.2008 02:00

Use the [Relearn] buttons in the rows to start a new learning of the according value. Press the [Relearn all] button to initiate a new learn for all currently listed values. (You can, for example, select only one type in the selection fields and then start a new learning only for this type.)

Press the [Edit Learn] button to see and change the definitions of the different learn procedures.

Three in one

In some projects, the evaluation parameter list and/ or the list of learn parameters are integrated into the list for limit curves and limit values. In this case, the form for limit values looks as follows:



As described for the evaluation parameter list, now you find in this form the buttons “add combinations” and “Measurements”. They work in the same way as described above. In the same way, the learn control works as described for the list of learn parameter.

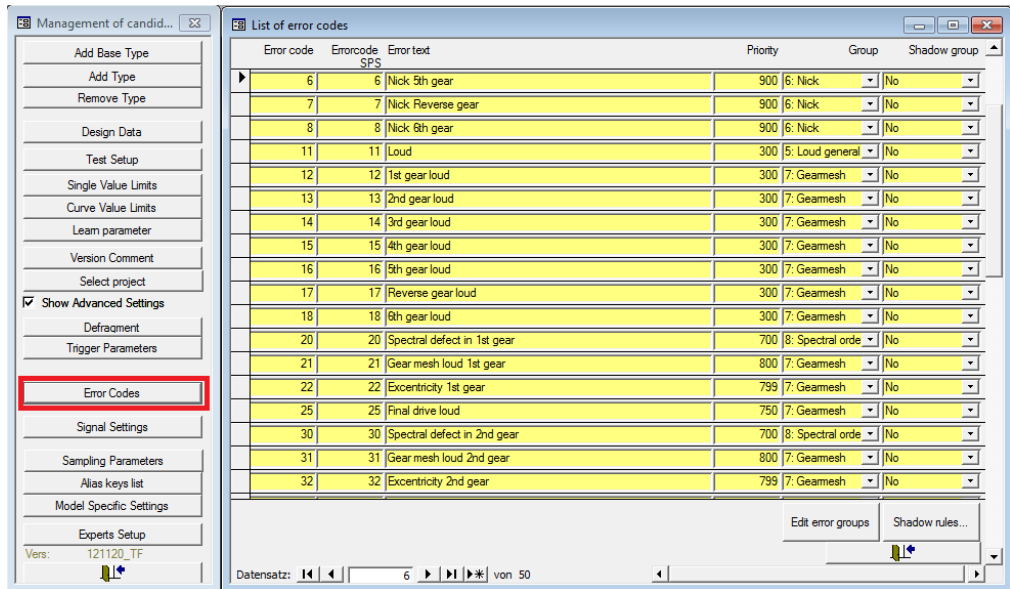
Since the amount of data as shown in the figure above requires a wide screen to be displayed adequately, you can partially hide some data for a better overview. This is being done with the two buttons marked with the red rectangle. If necessary, you can hide with these buttons the limit settings (then you only see the learn parameters) or the learn parameters (then you only see the limit settings). You need rarely both settings at the same time.

When you open the limit settings list with integrated learn parameters, the learn parameters are hidden by default, because re-learning is needed not as often as changing limits. Then you find one of the buttons on the bottom right corner to “un-hide” the learn parameters.

Defining Error Codes

Each measured value (be it a single value or a curve) is associated with an error code. When the measured value violates a limit, the associated error code is raised and the associated message is shown in TasAlyser's report window (and later in the evaluation software).

Error codes and error messages are defined in the parameter data base and can be changed as needed. They are listed in the Error Codes form:



Error code	Errorcode	Error text	Priority	Group	Shadow group
6	6	Nick 5th gear	900	6: Nick	No
7	7	Nick Reverse gear	900	6: Nick	No
8	8	Nick 6th gear	900	6: Nick	No
11	11	Loud	300	5: Loud general	No
12	12	1st gear loud	300	7: Gearmesh	No
13	13	2nd gear loud	300	7: Gearmesh	No
14	14	3rd gear loud	300	7: Gearmesh	No
15	15	4th gear loud	300	7: Gearmesh	No
16	16	5th gear loud	300	7: Gearmesh	No
17	17	Reverse gear loud	300	7: Gearmesh	No
18	18	6th gear loud	300	7: Gearmesh	No
20	20	Spectral defect in 1st gear	700	8: Spectral orde	No
21	21	Gear mesh loud 1st gear	800	7: Gearmesh	No
22	22	Excentricity 1st gear	799	7: Gearmesh	No
25	25	Final drive loud	750	7: Gearmesh	No
30	30	Spectral defect in 2nd gear	700	8: Spectral orde	No
31	31	Gear mesh loud 2nd gear	800	7: Gearmesh	No
32	32	Excentricity 2nd gear	799	7: Gearmesh	No

Changing the error messages

The error texts associated with the codes can be changed at any time. There is no limitation to the length of error texts (besides practical considerations about the readability and the on-screen display size). If you are using a language we Discom people are not fluent in reading (like e.g. Chinese), we recommend using dual-language texts like “齿轮啮合响 Gearmesh loud”. So when you ask us for advice and send us measured data (see “Help from Discom” on page 60), we will be able to understand the problem.

Adding Errors

You can add lines to the error code table as needed. (You can also delete error codes you do not need any more.) Each line consists of the following entries:

Error Code	The error code can be any positive number less than 2147483648. The used codes do not need to be sequential (you do not need to define error 99 if you wish to use error 100). Error code 0 (Zero) is not allowed.
External code (SPS code)	When the test stand control asks TasAlyser for the error messages, it gets the <i>external</i> code as a reply. In most cases, the external code will be equal to the error code. As an example, the external code can be used to map several errors (like all errors for one gear) to the same test stand code, because the test stand can handle less error codes than the measurement system.
Error text	The text displayed in the report window. See remarks above.
Priority	The errors from one test run are sorted by priority (highest priority goes first). For the production statistics, the first error (the one with the highest priority) is evaluated.
Group/Severity	Errors can be sorted into groups with ascending severity. The severity group can be used for additional statistics or for advanced techniques like re-typing of aggregates.
Shadow group	Shadow groups can be used to implement relationships between errors. So for example, nicks tend to elevate the spectrum and raise a number of spectral errors in addition to the nick error message. Shadow groups can be used to suppress the spectral errors in these cases.

To add an error, just select one line, copy it (**Ctrl+C**) and paste it at the end of the table (**Ctrl+V**) and then change the entries as needed. After you have added the error code, you can use it in the measurement value definition.

Test Stand Errors

The test stand can send error codes to the measurement system. These “test stand errors” are treated as normal errors: they are displayed together with the acoustical errors, they render a “not OK” overall result and they are stored in the result data base.

Any error code the test stand wants to send must be defined in the Error Codes table. (If the code sent by the test stand must be mapped to a different TasAlyser error code, this can also be achieved.) Please refer to the documentation about the Test Stand Commands for more details about sending test stand errors.

Trigger

The purpose of a trigger is to check control values (speed, torque, etc.) and add marks to the signal flow on certain circumstances. Such a circumstance usually is the passing of a certain value. The marks then can be used to initiate other actions like starting or stopping a measurement or the recording of a point for a tracking curve.

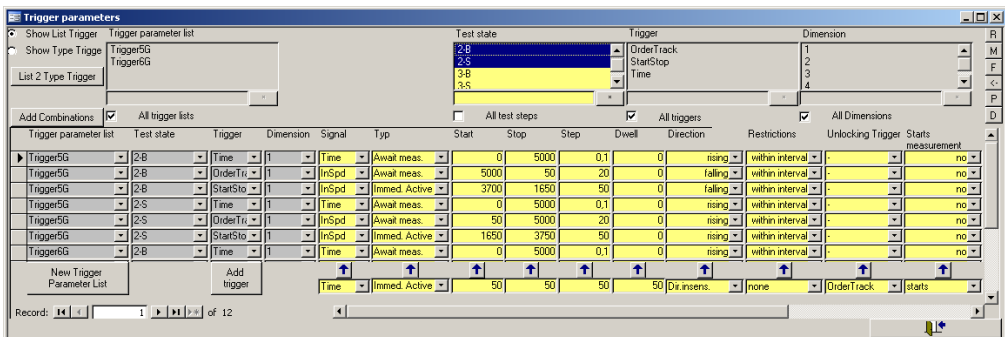
Where Triggers are used

The most frequent use of a trigger is for recording tracking curves depending on a control value, typically an order track or spectrogram relative to input speed. With reference to time, control values (like speed) are tracked during measurement to check the measurement conditions. In both situations, you must specify a trigger to get results.

In case the test stand does not send these commands, trigger are also used to start and stop measurements. If in doubt, the use of a trigger is preferable because the noise analysis usually checks control values with finer resolution than the test stand. Moreover, you do not need to take the command sending and processing time into consideration (which also may not be exactly the same at all times).

Trigger setup

The trigger parameters are set in the corresponding dialog in the parameter database section **Trigger Parameters**.



Trigger-Fields

By specifying **start**, **stop** and **step**, a segmented control value interval is being defined. Generally, when the control value enters the interval, a trigger is being started. When a segment boundary is passed, a mark is being sent, and leaving the interval stops the trigger. We call the interval *field* and the segments *cells*.

If the difference between **start** and **stop** is not dividable through the **step** size, the higher value is being rounded up.

Directions

The behavior of control values that leads to trigger events can be specified more detailedly with further settings. If a certain **direction** is specified, the trigger has to reach the interval following the specified direction in order to start the trigger. The same is true for the segment boundaries. Although the trigger can skip segments on sudden changes of the control value, if the control value leaves the segment in the wrong direction this does not lead to another trigger mark. Instead, the trigger waits until the control value enters another segment (relative to the last trigger point) in the right direction before sending the next mark.

Restrictions

The behaviour described above belongs to a trigger having the **restriction** “**within interval**”. If this restriction is changed to **none**, the trigger marks each change of the control value in the right direction. Basis for calculation of the segments are **start** and **stop** value as usual, although they are ignored as such.

Dwell time

Under certain circumstances, for example if the control value is not stable, it can be useful to specify a minimum **dwell** for a trigger cell. This is being seen in units of data blocks. The trigger counts the number of data blocks for which the value of the control value remains inside the cell. When the **dwell** time has been reached, the mark is being sent. If the control value leaves the cell (in whatever direction), no mark is being sent and the counter resetted. The cell is then considered as not reached.

Please note that you create a time shift when using a dwell time greater than zero. Moreover, some calculations may behave differently. Crest, Rms, etc calculate values on the basis of the predecessor signal blocks when using a dwell of zero. If you use a higher dwell, the calculation is restarted when entering a trigger cell. In this case, the calculation includes only those blocks where the control value was inside the corresponding cell.

More Than One Control Value

Although the trigger usually follows on control value, it is able to follow four of them. The trigger field then gets multidimensional and each **dimension** is set up in the database separately. The entries **typ**, **dwell**, **unlocking trigger** and **starts measurement** which are common for all dimensions are specified with the entries in the first dimension.

Trigger-Types

Basically, triggers have to be „unlocked“. Trigger starting measurements are **immediately active**, that means they follow their control values and send marks on corresponding events without the need of further activation.

On the other hand, the **Await measurement** type needs further activation. Its control values can have every possible value, it does not send marks before a measurement has been started (manually, by test stand or by another trigger).

The third type is the **Follower** trigger. This type is unlocked when the **unlocking trigger** has reached its end condition. With this trigger, trigger cascades can be built.

The last type is the **continuous** type. This type is used to record *each* measured value, disregarding trigger points. Therefore, this is just formally a trigger. Continuous trigger are basically working for spectrograms and have the following restrictions:

1. They always run between start and end measurement
2. They start always at ,0'
3. **Stop**, **Step** and **Direction** must specify a rising ramp
4. The control value for the trigger must be a *speed* signal for measurements in synchronous channels (or mix). Fixed frequency channels must use the signal *time*.
5. For trigger using time, the segments should correspond with the time resolution of the signal flow. Otherwise either data is being lost (too large segments) or you do not have data for each segment (too fine segments).

Trigger Using Signal Time

A trigger using time determines that time using the sampling rate in the signal flow. The 0-position is the point when the trigger is being unlocked. An **immediately active** trigger has the 0-point when the mode starts, trigger of the type **Await measurement** have their 0-point when the measurement starts. Finally, **Follower** have their 0-point when their unlocking trigger ends. 0 is a valid start time.

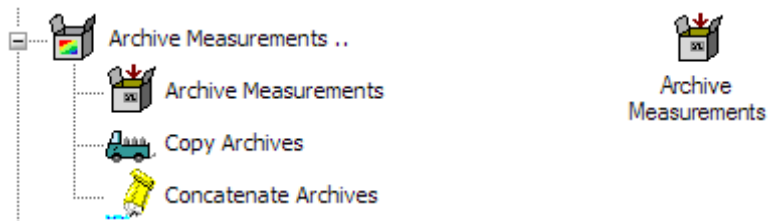
Measurement Archives and Evaluation

The measurement program stores the data of completed tests in *Archives* with a special file format and the file extension `rdt`. These archives can be stored in a central place (on a server or the measurement computer) and indexed by a database.

The complement to the measurement program which stores the archives is the evaluation program, known as *Presentation*. This program allows you to read the archive files, evaluate the content (the measurement data) and display the results graphically. While the measurement program can only be started on a computer equipped with a TAS Box, the presentation program can be used on any PC.

Archiving in the TasAlyser

In the TasAlyser one module is responsible for the creation of the archive files. You find it in the favorites or in the system configuration below **Evaluation**.



By double-clicking on the entry you open the dialog of the archive module. Here you can specify where the archive files are to be stored and how the name is constructed.

The **Write Files** control box at the top is particularly important. Here you can completely deactivate the archiving process.

Moving Archives to the Collector

An archive file is created by the archiving module for each test cycle, typically in the subdirectory **TempArchives** of the project directory (see dialog setting in the picture above).

Normally, the archives do not remain in this directory. Only with the mobile application they remain in the folder where they are created because they shall later be moved manually to a suitable location of choice. In the test bench environment, the archive files are moved to the `C:\Outbox` directory where they are fetched by the Collector service and integrated into the result database.



The Presentation App / Marvis

The presentation program is the tool with which you can look at, compare and evaluate stored measurement data. Within the presentation program you can represent the data on *layout pages*. Each page contains graphic modules (such as text fields and curve graphs) which display the data.

The data can come from the archive files as well as from the result database. Archive files are opened directly. For database queries, the *database assistant* will help you. After you loaded archives or carried out a database query, you get a list of all loaded measurements, identified by the measurement time, the serial number, type, test bench and further information.

After measurement have been loaded, the data can be displayed in the existing layouts and graphic modules with the help of *Rapports*, which can produce a complete test record or evaluation. Rapports offer many options and possibilities for producing test records, which can even spread on several pages.

This brief introduction only describes how to use Rapports, not how to create Rapports. The creation of Rapports, as well as the interactive display of data, are described more detailedly in the presentation manual.

Installing the Presentation Program

The presentation program is already installed on the measurement computer. You can also install the program on your personal computer to allow you to examine archive files there or access the result database.

Installation packages for this program can be found on our web server under www.discom.de/download/Presentation, where you also will find the installation instructions.

Together with the presentation *program* you also need – similar to the TasAlyser - a presentation *project*. A Presentation project consists of several files, which are contained in a project folder. Normally an appropriate project can be found in the folder C:\Discom\Analysis\Presentations or in the project directory of the appropriate TasAlyser project on the measurement computer. You can simply copy the folder with the presentation project onto your personal computer.

When you start the presentation program for the first time after installation, it does not yet know the project you want to work on. You will see a prompt informing you that the project base file is missing. Confirm the prompt, navigate in the **Open File** dialog to the project directory and open the base file found there (with the extension *bse*, e.g. *Presentation.bse* or *Base.bse*).

The presentation program subsequently remembers which project you opened last (even if you have deleted and updated presentation in the meantime).

Updating the Presentation

If you already installed the presentation program and only want to update it, you can download a package with the program files only from our server(www.discom.de/download/Presentation), the so called binaries. These packages are named Presentation-bin-(Date).zip.

After you unzipped the package, you get a folder Presentation. This folder contains in particular the file „Update Presentation.bat“. Run this batch and the binaries are being updated. (Of course, you have to end all running Presentation programs first.)



Help from Discom

The Discom team will do its best to help you not only with any problems you might have with the software and/or hardware, but also with strange noise phenomena, the choice of suitable parameters and any other issues to do with noise analysis.

Depending on the type of problem, you can make it easier for us to help you by making suitable information available - this information will usually be files from the measurement computer. This chapter describes how you can supply us with the necessary information and what else we need so that we can help you as efficiently as possible.

TasBackupTool

On the measurement computer's desktop you will find a folder called "Rotas for Exports" which contains a number of useful shortcuts. One of these is the *TasBackupTool* or *Software Maintenance Tool*:



Use it to create backups of the measurement and Presentation projects including all settings, the parameter database and other project related files.

When you start the *Backup Tool*, you are presented with an edit box where you can change the name of the backup folder (the default name is built from date and time) and two buttons. **Backup Project** will create a copy of all project settings, and **Backup Programs** will create a backup of the executable software itself.

You only need to create Program Backups when you intend to update the TasAlyser, Presentation or other application binaries. In normal cases, a Project backup is sufficient.

Make regular project backups after major changes to the project or the parameters. Also, if you want to send the project to Discom for further analysis (see sections below), you should do this by creating a backup with the Backup Tool and then zip this backup into a compressed archive.

The backups are created in the directory D:\Backup\Discom using the names you set in the Backup Tool (typically date and time).

Transmitting Files

Before you send us a file or a whole directory by email or upload, you should compress it. In the first place, this reduces the volume of data and, in the



second, you can then send a whole directory including subdirectories in one package without problem.

The program *7Zip* is pre-installed on all measurement computers. This is a free compression program, which you can also download from www.7zip.org.

To compress you select the relevant file or directory in Windows Explorer and call up the context menu using a right-click. Here you will find a submenu **7Zip**. From this submenu select one of the instructions “Add to Archive *xxx.zip*” (a suitable file name is usually offered to you) or “*xxx.7z*”. 7Zip will then provide an archive file in the same location as the file or the directory. Copy the archive file onto a transportation medium, e.g. a USB stick, and transfer this to a PC with internet access.

If you receive a compressed file from us or download one from our web page then you can use 7Zip to unpack the files. Right-click the file, go into the 7Zip submenu and select either the “Unpack File” or “Unpack here” instruction.

When the TasAlyser Does Not Work

The first question we will ask is “What does not work?” Some examples:

- Does the program start?
- Are there error messages on startup? (If yes, what do they tell?)
- Or does the malfunction occur during normal operation? If yes, what are the circumstances? (e.g. “Always when a test cycle begins”)
- Is the program “stuck”, meaning it does not react to the mouse, does not close, or similar?
- Has the TasAlyser crashed? (Often you’ll then get an error message “Debug Assertion Failed”, which you can close with **OK**.)
- Or is it a problem of communication, meaning that the measurement program no longer reacts to (all) test bench commands?
- Although the program is working you cannot see the rotational speed any more, or you can see the rotational speed, but no signals in the scopes?

When there is an error message from the TasAlyser, the text is always helpful. If there is a communication problem with the test bench you should consult the output window, communication section, for advice (see “Test Bench Connection“ on page 38 as well as “Docking Windows” on page 36). If the rotational speed or the noise signals are suddenly missing, check the



appropriate sensors and the cables between the sensors and the measurement computer. Otherwise, follow the procedure given below:

1. If the TasAlyser does not react any more, then you have to “kill” it, (using the Windows task manager). Restart the program but do not begin a test. Note any error messages at program startup.
2. If the TasAlyser reacts, or if you have restarted it in 1., use the instruction **Info on the TasAlyser** in the **Help** menu. Here you will find a version number and specification “Build” with a date. Make a note of this information.
3. After this, use the instruction **Project Directory** from the instruction **File**, in order to call up Windows Explorer, which displays the project directory. Terminate the TasAlyser.
4. Use the pre-installed 7Zip, as described above, to compress the `Application` folder in the project directory. Send this folder to us, as well as the version information in 2. and any error messages in 1.
5. If you know of any circumstances or other details which have caused the TasAlyser’s malfunction, mention these also in your email.

Please contact us immediately. In many cases, we can state the cause of the problem immediately or after a short examination of the files and then offer a suitable remedy. Otherwise, we will discuss with you how to proceed further.

Strange Noises

We are always interested in learning about new noise phenomena and investigate these with you. Other than with problems with the TasAlyser program, where we need settings and log files (see previous section), with noise phenomena we are interested in the noises themselves (Wave files) and/or associated measurement data archives.

Wave files

Use the function for recording Wave files to record the test cycles of some interesting aggregates (see “Wave Audio Recording and Playback ” on page 39). Give these files an appropriate name and upload them on our ftp server. You can read the directory in which the measurement program has stored its recordings in the control window of the Wave Recorder.

Measurement Data Archives

Usually, we also need the the associated measurement data archives together with the wave files. It is best to deactivate the transport of the archives to the database collector temporarily, so that the individual archives remain in the



project directory's **TempArchives** folder. Compress this folder and send it to us or upload it onto our server together with the wave files.

Unwanted Test Results

Sometimes it occurs that the noise analysis system rejects too many tests with NOK result. Generally speaking, a n.O.K is caused by some measure value trespassing its limit and can be fixed by shifting the corresponding limit. Before you proceed that way, you should nevertheless try to evaluate *why* this happens. Maybe, the noise analysis system is quite right in rejecting the transmissions.

First of all, you should listen to the actual noises using the TasWavEditor or audio monitor of TasAlyser. You can do this either at the test bench (using head phones), or copy the Wave files to your desktop PC and use TasWavEditor. Use the presentation program to compare n.O.K. measurements with (probably older) O.K. measurements.

For us to be able to advise you, we need measurement data archives as well as the parameter database. The latter is found in the project directory's subdirectory **ParamDb**. Go into this folder and compress the mdb file(s) which are contained there. Simply send us the result by email. (Do not compress the whole folder because if you do, you will also include the **Backup** subfolder into the package, which can be very large, but does not help us in the analysis.)