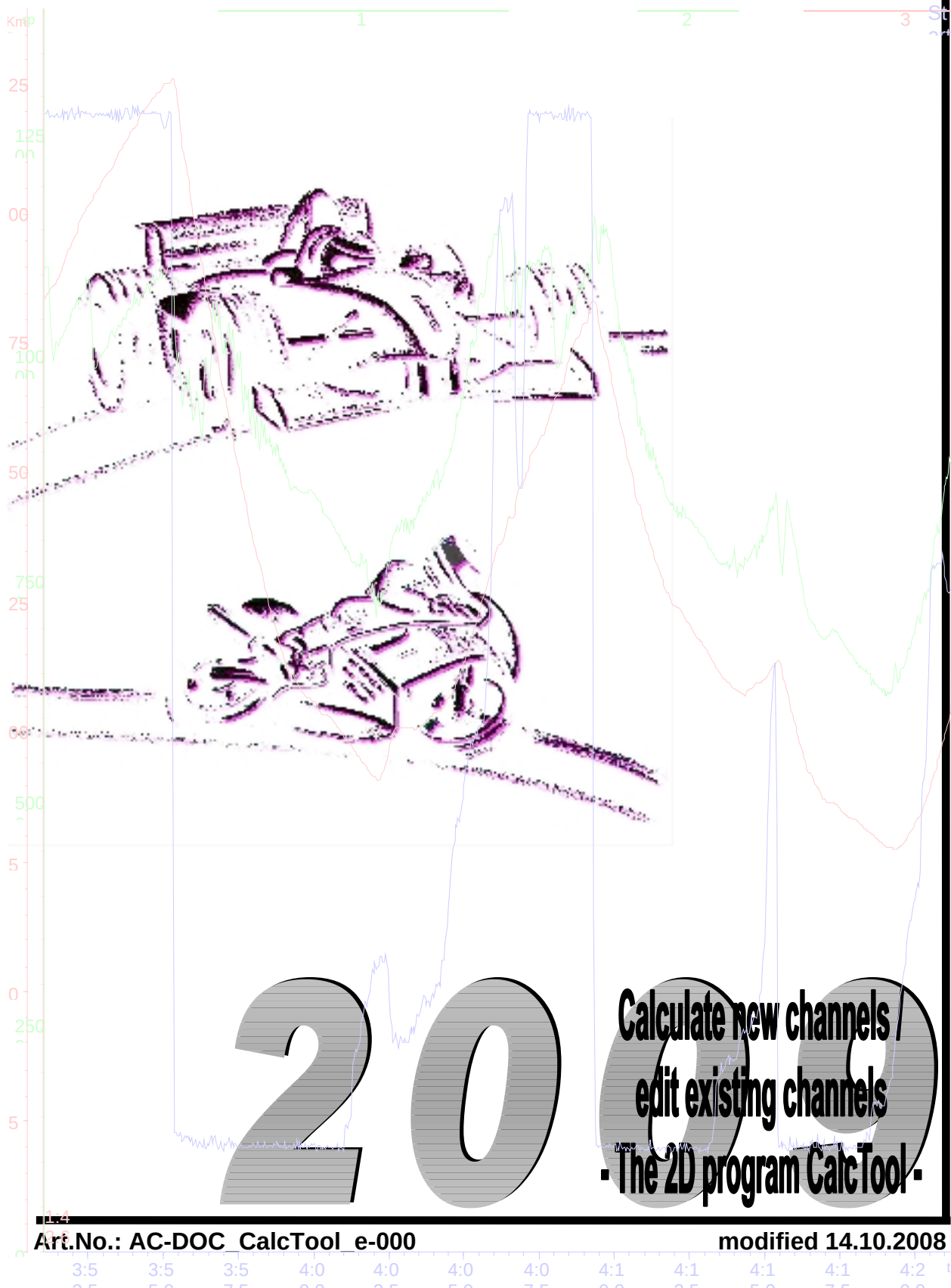


- English -



0. INTRODUCTION	4
1. PRE- AND POSTPROCESSOR	10
1.1 PRE-PROCESSOR-FUNCTIONS	10
1.1.1. Include-files	10
1.1.2. Conditional execution	10
1.1.2.1. Condition IfExists	10
1.1.2.2. Condition IfNotExists	10
1.1.2.3. Condition IfLaptimesExist	11
1.1.2.4. Condition IfNotLaptimesExist	11
1.1.2.5. Condition IfSpecValueExists	11
1.1.2.6. Condition IfNotSpecValueExists	11
1.1.2.7. Condition IfSpecValueContains	12
1.1.2.8. Condition IfNotSpecValueContains	12
1.1.3. Entering values at runtime	13
1.1.3.1. Entering a channel name: EnterChannel-function	13
1.1.3.2. Entering a numeric value: EnterValue-function	13
1.1.4. Substitution of place holders	13
1.1.4.1. Special channels	14
1.1.4.2. Predefined constants	14
1.1.4.3. Specs sheet-variables	15
1.1.4.4. Variables	17
1.1.4.5. Channel properties	17
1.1.5. Calculated mileage	18
1.2 POSTPROCESSOR-FUNCTIONS	18
1.2.1. Deleting channels	18
1.2.2. Hiding channels	18
1.2.3. Unhiding hidden channels	18
2. AVAILABLE FUNCTIONS	19
2.0 FORMAT OF NUMERIC PARAMETERS	19
2.1. STANDARD MATHEMATICAL FUNCTIONS	19
2.1.1. Addition	19
2.1.2. Subtraction	19
2.1.3. Multiplication	20
2.1.4. Division	20
2.1.5. Modulo	20
2.1.6. Integer division – DIV	21
2.1.7. Rounding to whole numbers	21
2.1.7.1. The Round function	21
2.1.7.2. The Floor function	21
2.1.7.3. The Trunc function	22
2.1.8. The Const function – CONST	22
2.1.9. The minimum value function – MinValue	22
2.1.10. The maximum value function – MaxValue	22
2.1.11. The first value function – FirstValue	22
2.1.12. The last value function – LastValue	23
2.1.13. The average value function – AvgValue	23
2.2. LOGICAL FUNCTIONS	23
2.2.1. AND function	23
2.2.2. OR function	23
2.2.3. XOR function	24
2.3. TRIGONOMETRIC FUNCTIONS	25
2.4. ADVANCED MATHEMATICAL FUNCTIONS	27
2.4.1. Square Root	27
2.4.2. Logarithms	27
2.4.3. Absolute value	27
2.4.4. Minimum function	27
2.4.5. Maximum function	27
2.4.6. Derivation	28
2.4.7. Integration	28

2.4.8. SUM-function.....	29
2.4.9. Power-function	29
2.4.10. Exp-function.....	29
2.5. IF-FUNCTION	30
2.6. ADDITIONAL FORMULAS	30
2.6.1. Formula Integer.....	30
2.6.2. Formula Real	31
2.6.3. Table function	31
2.6.4. Table 2 function	31
2.6.5. 2D-Table function.....	32
2.6.6. 2D-Table search function	33
2.6.7. 3D-Table function.....	34
2.6.8. Filter function	35
2.6.8.1. Simple filter interface.....	35
2.6.8.2. Full filter interface.....	36
2.6.8.3. Filter with variable depth	37
2.6.9. Frequency reduction/increase	38
2.6.10. Sampling rate adjustment for non 2D samplingrates: FreqBase	38
2.6.11. Demultiplexing a multiplexed channel: MUX-function.....	39
2.6.12. Masking out used bits: UseBits-function	39
2.6.13. GPS Shift.....	40
2.7. BIKE AND CAR PHYSICAL FORMULAS	41
2.7.1. Speed.....	41
2.7.2. Speed2.....	42
2.7.3. Slip	42
2.7.4. Banking	43
2.7.5.BANK2ACC function	43
2.8. CHANNEL MANIPULATION	44
2.8.1. Shifting a channel	44
2.8.2. Combining two 16-bit channels.....	45
2.8.3. Combining two 32 bit channels	46
2.8.4. Changing the storage type	47
2.8.4.1. WORD-function	47
2.8.4.2. BINARY-function	48
2.8.4.3. Signed function.....	48
2.8.5. SWAP-function – Changing the byte-order of a channel	49
2.8.6. Changing dimension.....	49
2.9. SPECIAL FUNCTIONS FOR PHASE-CALCULATION	49
2.9.1. MaxHold - function.....	50
2.9.2. MinHold – function.....	52
2.9.3. FillFromBool – function	52
2.10. STOPPING EXECUTION	52
2.10.1. Abort calculation of group when an error occurred: OnErrorExitGroup	52
2.10.2. Abort whole calculation when an error occurred: OnErrorExitTotal	53
2.10.3. Abort whole calculation: ExitTotal	53
2.11. CALLING EXTERNAL PROGRAMS	53
3. HOW TO USE THESE FUNCTIONS IN ANALYZER	54
3.1. NAMING CHANNELS.....	55
3.2. REPLACING CHANNELS.....	55
3.3. THE LINK BETWEEN CALCULATION AND SPECSHEETS	56
3.3.1. Partial reading of SpecSheet values.....	57
3.4. AUTOMATIC CHANNEL-CALCULATION	57
4. TABLE OF ALL CALCULATION FUNCTIONS	58
APPENDIX A: – EXAMPLE OF A CAL FILE.....	63
APPENDIX B: THE LINK BETWEEN CALCTOOL AND SPECSHEETS.....	65
APPENDIX C: WHAT'S NEW ?	66

0. Introduction



CalcTool is part of the 2D software WinARace. It is used to create calculated channels from channels measured and recorded by a 2D Data Acquisition System.

CalcTool is called automatically from the communication software WinIt to linearize non linear channel data.

The user can define the channels to be calculated by writing a so called CAL-file, which is a simple text file containing the instructions for the calculation.

In the following we describe the set of functions of the software and their syntax. We try to demonstrate the use of each function with an example.

In the [Appendix](#) we will give you one practical example for a calculation file (CAL – file).



Take notice of: Some functions of the program CalcTool depends of the software license you have bought. The following table shows you the differences between the several licence types.

Licence type	Licence differences
• Small	You can only linearize table channels (called from program WinIt)
• System	• Additional to the linearization you can calculate new channels. • You can save only 10 calculation channels. (Intermediate results and temporary channel were not disregarded)
• Full	• No Restrictions.

How can I start the program CalcTool manually ?

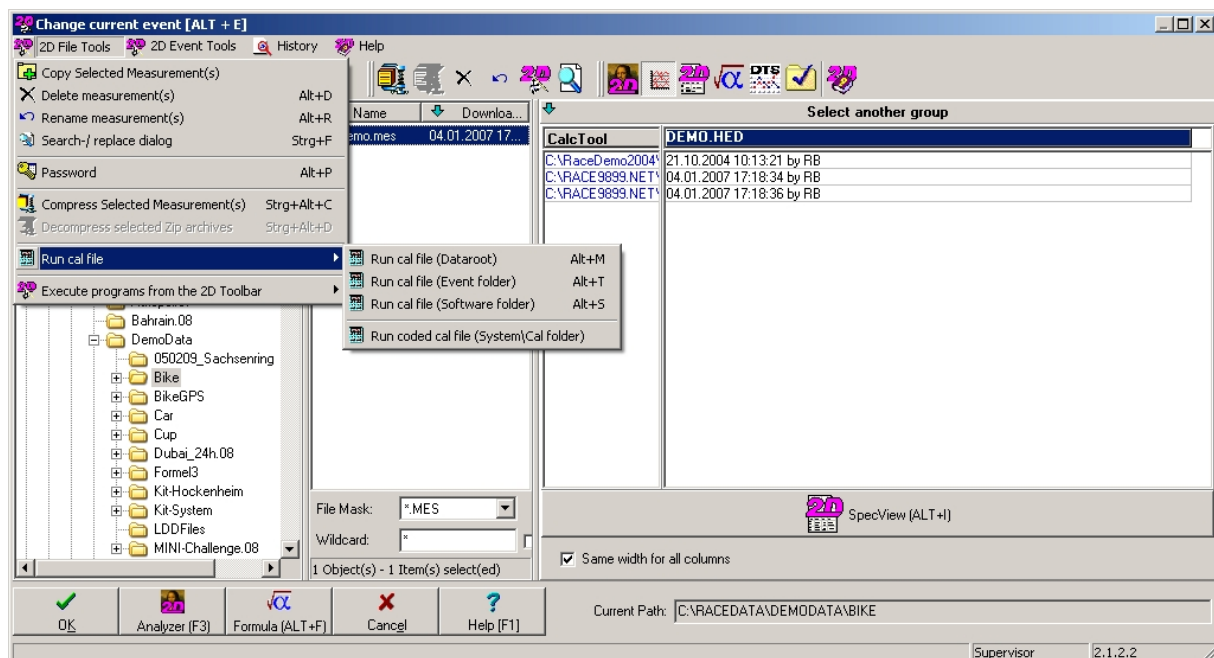
They are many different ways to start the program CalcTool with before selected CAL files.

In principle we distinguish into two different kind of CAL-files, they are:

- No coded CAL files (= files with extension *.cal)
- Coded CAL files (=files with extension *.ccf)

First possibility

- Via the 2D Event Explorer
- Use the button "Change Event" in WinARace
- Select menu item "2D File Tools / Run cal file"



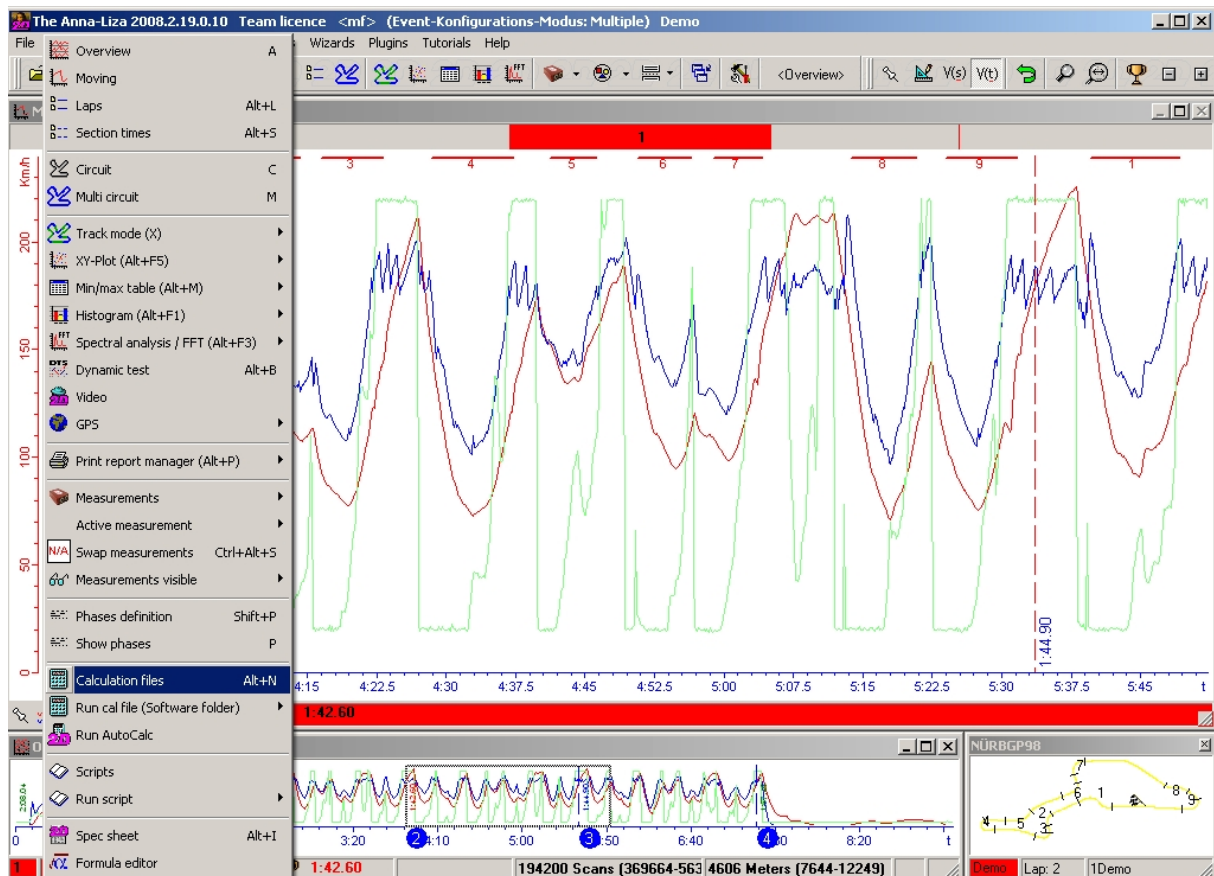
The user can choose from 4 different options, these are:

1. Start the program CalcTool with selected measurements and using any CAL-file (by selection) from the Dataroot à shortcut <Alt+M>
2. Start the program CalcTool with selected measurements and using any CAL-file (by selection) from the current Event folder. à shortcut: <ALT + T>
3. Start the program CalcTool with selected measurements and using any CAL-file (by selection) from the software folder. This folder corresponds with the sub-directory called as "CalFiles" located into your current Race version. shortcut: <ALT + S>
4. Start the program CalcTool with selected measurements and using any coded CAL-file (by selection) from "System\Cal folder". This folder corresponds with the sub-directory called as "System\Cal" located into your current Race version.

Second possibility

From the Analysis software "Analyzer"

- "Functions / Calculation files" ⌘ shortcut <Alt + N>
- "Functions / Run cal file (Software folder)"



Via the first menu item (Functions / Calculation files) you can select any CAL-file from the current Event.

The second menu item search for CAL-files into a special folder. The folder is named "CalFiles" and must be exist directly into your current WinARace directory !

Over the first menu item (=shortcut <Alt+N>) the "calculation file manager window" will be opened. From there you can open all CAL-files which offered also in the 2D Event explorer window described in chapter before ([refer first possibility](#)).

Thus CAL-files from the current data root, the current event and from the software folder ("CalFiles").

As additional possibility the analysis software offers the selection of CAL-files from the current measurment directory. Use for this the tab "Measurement" within the "Calculation file manager window"

How can I create own CAL files ?



The software offers an own CAL-file editor named as " CalcEdit.exe". All CAL files are linked according to standard with this editor. That means a double click (with left mouse button) to a CAL file opens these in the editor.

To change/create new CAL files this editor should be always used, because this offers the following advantages:

- Examination of the function regulations on correctness (= syntax examination)
- Autocompletion

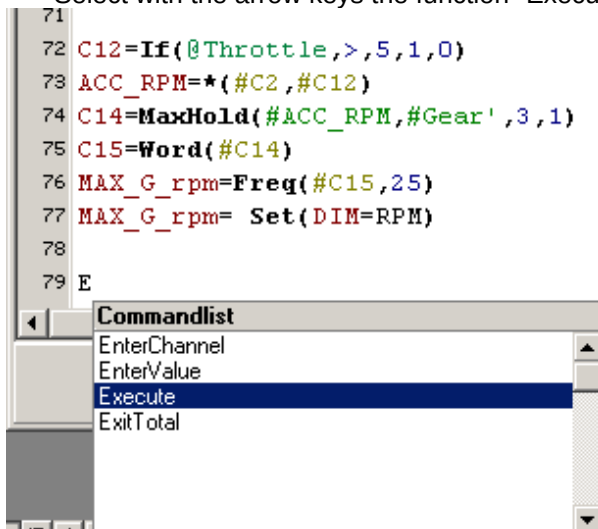
How do I use the "autocompleting function" correctly ?

The user avoids input errors as it uses the "autocompletion" of the editor. This indicates the correct syntax to a function.

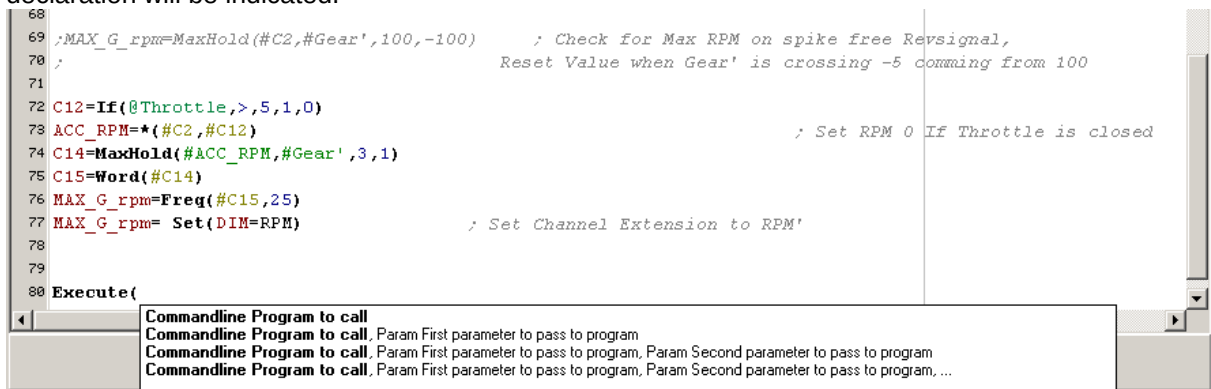
Example:

The user would like to use the function "Execute" ([refer chapter 2.11](#)). Proceed as follows:

- Open first the CAL file in the "calculation file manager" by using the button <Edit> or if you would like to create a new file use button <New>
- Enter the initial letters of the function à "E"
- Confirm your input with the keys <CTRL> + <Space Bar>
- The software will display a "command list" window
- Select with the arrow keys the function "Execute" and confirm with <Return>



A function call is always followed by a "(" character. Enter this ! After a short delay the correct function declaration will be indicated.



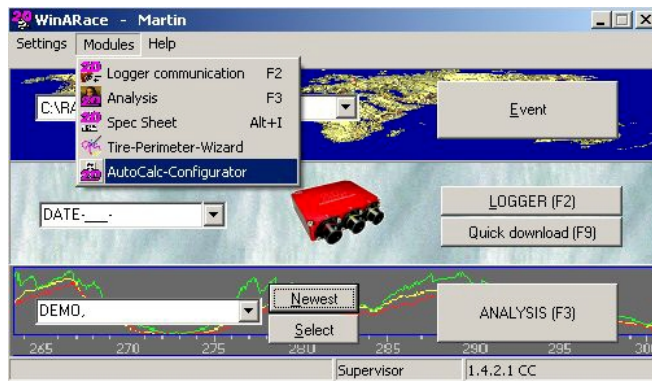
- On the basis of the function declaration you can recognize to call the function "Execute" with one or more parameters separated by a comma.

- The coloured distinction helps you to enter a correct syntax.

Automatic processing of the measurement data by configure the “AutoCalc Configurator”



You can start the AutoCalc-Configurator directly from WinARace



Define the sequence of channel calculations:

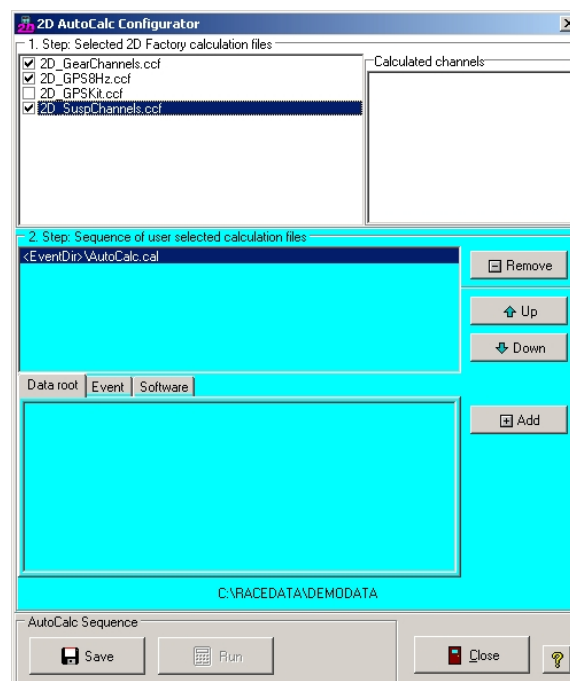
- automatically after download
- manually in Analyzer (a refer chapter 2.2.2)

1. Step: Selected 2D Factory calculation files

Choose from a predefined list of delivered 2D Factory calculation files.

Factory calculation file names always start with "2D_".

Calculation files having the following 3 characters equal after "2D_" are forming a group. Those



calculation files can only be selected exclusively inside their group, e.g. you can select "2D_GPS8Hz.ccf" or "2D_GPS4Hz.ccf" or "2D_GPSKit.ccf", but only one of them.



The Factory calculation files are coded and thus cannot be modified by the user.

2. Step: Sequence of user selected calculation files

User calculation files are located in 3 different locations, named inside data root, event or software folder.

The user calculation files are managed by the user, e.g. in Cal file manager inside Analyzer.

Add a calculation file to the AutoCalc sequence by clicking on the "Add" button or by right-click on the list

Change the order of the user calculation sequence by moving an entry up or down.

Each entry of the user calculation sequence can be removed by clicking on the appropriate button.



Save the sequence after set-up and optionally run it to check proper calculation.

1.Pre- and Postprocessor



Besides the instructions for calculation CalcTool provides a set of functions which are performed before or after the real calculation. These functions are done either by the so called pre-processor (function before calculation) or the postprocessor (after the calculation). This chapter describes the use of the pre- and postprocessor.

1.1 Pre-processor-functions

The Pre-processor functions are done before the calculation of the channels.

1.1.1. Include-files

It is possible to include files in a calculation file by using the Include-command.

Syntax:	Meaning
{\$! FileName}	Paste the content of the file FileName at the current position.

1.1.2. Conditional execution

In some cases it could be helpful to execute a calculation only if a certain channel or Spec Sheet value exists (or if it not exists). For this reason CalcTool provides the conditional execution of calculation groups.

It is possible to use more than one condition for a group, but the condition lines have to be one after the other at the start of the group.

1.1.2.1. Condition IfExists

With this condition the user has the possibility to calculate a single group only if a channel exists.

Syntax:	Meaning
[MYCALC]	Start of group
IfExists(#CheckChannel)	Condition
Result = ...	This calculation is only done if CheckChannel exists.



[Example:](#)
[Refer chapter 1.1.3.3 \(Specsheet variables\)](#)

1.1.2.2. Condition IfNotExists

This condition is the opposite of the one above. A single calculation group is only calculated if a certain channel does not exist.

Syntax:	Meaning
[MYCALC]	Start of group
IfNotExists(#CheckChannel)	Condition
Result = ...	This calculation is only done if CheckChannel does not exist.

1.1.2.3. Condition IfLaptimesExist



With this condition the user has possibility to calculate a single group only if the measurement contains laptimes.

Syntax:	Meaning
[MYCALC]	Start of group
IfLaptimesExist	Condition
Result = ...	This calculation is only done if the measurement contains laptimes.

1.1.2.4. Condition IfNotLaptimesExist



This condition is the opposite of the one above. A single calculation group is only calculated if the measurement does not contain laptimes.

Syntax:	Meaning
[MYCALC]	Start of group
IfNotLaptimesExist	Condition
Result = ...	This calculation is only done if the measurement <u>does not</u> contain laptimes.

1.1.2.5. Condition IfSpecValueExists



With this condition the user has possibility to calculate a single group only if the Spec Sheet of the measurement contains a certain entry.

Syntax:	Meaning
[MYCALC]	Start of group
IfSpecValueExists(Group.Entry)	Condition
Result = ...	This calculation is only done if the Spec sheet of the measurement contains group "Group" with the entry "Entry".

1.1.2.6. Condition IfNotSpecValueExists



This condition is the opposite of the one above. A single calculation group is only calculated if the Spec Sheet of the measurement does not contain the entry "Entry" in the group "Group".

Syntax:	Meaning
[MYCALC]	Start of group
IfNotSpecValueExists(Group.Entry)	Condition
Result = ...	This calculation is only done if the Spec Sheet of the measurement <u>does not</u> contain an entry "Entry" in the group "Group".

1.1.2.7. Condition IfSpecValueContains



With this condition the user has possibility to calculate a single group only if the Spec Sheet of the measurement contains a certain entry and if the entry contains a certain string.

Syntax:	Meaning
[MYCALC]	Start of group
IfSpecValueContains(Group.Entry, SearchStr) Result = ...	Condition This calculation is only done if the Spec sheet of the measurement contains group "Group" with the entry "Entry" and the entry contains the searchstring.

1.1.2.8. Condition IfNotSpecValueContains



This condition is the opposite of the one above. A single calculation group is only calculated if the Spec Sheet of the measurement does not contain the entry "Entry" in the group "Group" or if the entry exists but does not contain a certain searchstring.

Syntax:	Meaning
[MYCALC]	Start of group
IfNotSpecValueContains(Group.Entry, SearchStr) Result = ...	Condition This calculation is only done if the Spec Sheet of the measurement <u>does not</u> contain an entry "Entry" in the group "Group" or the entry exists but does not contain a certain search string.

1.1.3. Entering values at runtime

To make calculation files more flexible, CalcTool provides two functions to enter information by the user at runtime.

1.1.3.1. Entering a channel name: EnterChannel-function

Syntax:	Meaning
C1=EnterChannel('Text shown in dialog')	Opens a dialog to ask the user for a channel name. The text in the parameter is shown in the dialog.
C1=EnterChannel('Text', #Channel)	Like above, but with a default channel name, which will be selected in the shown dialog

This function opens a dialog to ask the user for the name of a channel to use in further calculation. The user may choose from a list of channels available in the specific measurement.

It is possible to add the name of a channel as second parameter. When the dialog opens and a channel with the given name exists, it is selected in the dialog.

When the user cancels the dialog the calculation will stop at this line. Channels which were calculated in lines before will be stored, lines and groups after this line will be ignored.

1.1.3.2. Entering a numeric value: EnterValue-function

Syntax:	Meaning
@Name=EnterValue('Text shown in dialog')	Opens a dialog to ask the user for a numeric value. The text in the parameter is shown in the dialog.
@Name=EnterValue('Text', Value)	Like above, but with a default value, which will be selected in the shown dialog

This function opens a dialog to ask the user to enter a numeric value to use in further calculation. The dialog shows the text parameter of the function. The entered value is stored in a variable with the name which is given on the left side of the "=". The name has to start with a "@"-character.

The variable can be used in the following calculation like any variable. (See variables)

It is possible to add a number as second parameter of the function. When the dialog opens this value is already entered in the dialog. To use this default value you simply have to push the <ENTER>-Button, otherwise overwrite the value with the one you want to use.

When the user cancels the dialog the calculation will stop at this line. Channels which were calculated in lines before will be stored, lines and groups after this line will be ignored.

1.1.4. Substitution of place holders



A place holder starts with the @-character.

There are four kinds of place holders possible.

1. The first are special channels read from the Event.Ini.
2. The second are often used constants.
3. The third is the possibility to replace identifier of the spec-sheet by variable names out of this sheet.
4. And the last is replacing variable names by their value.

1.1.4.1. Special channels

In the analysis - module, the user may define special channels (Such as the speed channel to integrate the distance). These are stored in the file Event.ini. This gives the user the possibility, to use one calculation file on different systems even if channels with the same meaning do not have the same name. Simply use the place holders, CalcTool will replace them with the corresponding channel name.

Example:

If the special channel Front_Susp is defined as FSusp in the Event.Ini (Menu of Analyzer: "Settings"- "Special Channels").

FSus_lo = F(@Front_Susp, F(L_PHAMM3HZ77))

is replaced by

FSus_lo = F(#FSusp, F(L_PHAMM3HZ77))

Following special channels are possible:	Meaning
Int_Speed	Speed channel used for integration of the distance
Drive_RPM	Channel with the rpm
Drive_Speed	Speed channel used for gear calculation
Throttle	Channel with the throttle signal
Front_Susp	Signal of front suspension (cars front left)
Rear_Susp	Signal of rear suspension (cars rear left)
Front_SuspLeft	Signal of left front suspension
Acc_Lat	Signal of lateral acceleration

1.1.4.2. Predefined constants



The name of a constant starts with the @-character.

Following constants are available in CalcTool:	
@PI	3.1415...
@2PI	6.283....
@PI/2	1.5707..
@E	Eulerconstant 2.71828...
@G	Gravity-constant 9.80665
@LN2	Logarithms naturalis of 2
@RAD2DEG	180/PI constant to calculate radiant to degrees
@DEG2RAD	PI/180 constant to calculate degrees to radiant

1.1.4.3. Specs-sheet-variables



The user has the possibility to use values from the Specs-sheet as constants for calculation. In some cases it would be good, if not only a constant value is possible but a variable. This function uses the value read from the Specs-sheet as a variable name for further use. (See next chapter variables)

Example:

The following example shows, how to recalculate the speed channel of a BIKE if the circumference of the tyre was not set correct before the measurement.



We assume, that the user has written the name of the used tyre in the Specs-sheet of the measurement in the group [Tyre] with the identifier "Front".

In this example the name of the tyre is "Soft05".

There is an entry inside the Specs-sheet (File *.HED) of the run

...

[Tyre]

...

Front = Soft05

...

Additionally a file called Tires.HED exists inside the race directory.

In this file (TIRES.HED) we find the following entries:

[Variables]

Soft05 = 1885

Hard05 = 1885

another = 1885

M4124C = 1887

DefaultCircumference=1890

Then the expression @Tyre.Front is replaced by 1885.

First: @Tyre.Front is replaced by @Soft05

Second: @Soft05 is replaced by 1885



(If you want to use values from the Specs-sheet directly [refer chapter 3.3. The Link between Calculation and Specs-sheets.](#))

Example:

The following example extends the previous example with a condition.

The recalculation of the speed is done in one group. The next group replaces the values of the original measured speed channel "VFront" with the result of a recalculation via a different tire diameter from the variable list. If the value does not exist in the variable list, nothing is calculated.

Syntax	Meaning
[VFRONT]	; Group for calculating a new speed channel
C1=I(#VFront, @DefaultCircumference)	; CalcTool reads the value for ; "DefaultCircumference" from the variable file ; in this example the value is 1890 ; C1 receives the values from VFront divided by 1890
C2=*(#C1, @Tyre.Front)	; CalcTool replaces "@Tyre.Front" by "@Soft05" ; what is read from the Specs sheet group "Tyre" ; with the identifier "Front" ; In the next step "@Soft05" is replaced by the ; value 1885, read from the variable-file. ; The result-channel C2 receives the product ; of the values of C1 and the new circumference ; Store the resulting channel in word-format
RESULT=WORD(#C2, 0, 16XX)	
[VFront]	; Group for substitution of the speed channel by ; the recalculated one. The name of the group has ; to be the name of the original speed-channel
IfExists(#SpeedRecalc)	; Condition. Rest of the group will only be ; computed if a channel with the name ; "SpeedRecalc" exists (if the previous calculation ; was OK)
NewResult= Word(#SpeedRecalc, 0, 500)	; Convert the recalculated speed channel to word ; using the borders 0 and 500 km/h ; If the keyword "NewResult" is used, the ; calculated channel will replace an original ; measured channel.



If an original channel is replaced by a calculated one using the keyword "NewResult", the original channel is **not** deleted. It is only invisible for the analysing software. If you use a channel for calculation, which is originally recorded by the logger, always the original channel is used even if it is replaced by a calculation with "NewResult".

1.1.4.4. Variables



It is possible to use variables in calculation files.
Like all placeholders a variable starts with “@” followed by any characters.

When CalcTool finds an identifier with the leading “@”, which is neither the

1. name of a special channel
2. nor the name of a Constant ([refer chapter 1.1.3.2](#))
3. nor the name of a Specsheel-variable ([refer chapter 1.1.3.3](#)), it reads from the variable file the value corresponding to that identifier.

The user can define, which variable-file should be used. The syntax to define the use of a variable-file is:

Syntax:	Meaning
{SV FILENAME}	Read variables from file FileName.

The filename has to contain the full path information. It is possible to use four predefined directories:

<DataDir>	Datadirectory of race
<EventDir>	Directory of the current event
<RaceDir>	Directory of the race installation
<MesDir>	Directory of the current measurement

Example:

{SV <EventDir>\MyVars.Hed}. Use the file “MyVars.Hed” in the event-directory as variable-file.



If no variable-file is specified, CalcTool uses the default filename, which is “CalcVars.Hed” located in the current event, but only if the files exists.

An other predefined filename placeholder could be used to load a variable file depending on the name of the current logger:

<LoggerName>	Will be replaced by the name of the logger with which the measurement was recorded.
--------------	---

Example:

{SV <DataDir>\<LoggerName>.Hed}.

Assumed that the name of the logger is “Log1”, CalcTool will load the file “Log1.Hed” from the data directory.

1.1.4.5. Channel properties

In some cases it could be helpful to use channel properties like samplingrate for calculation. These properties are replaced by their corresponding values before the calculation is started. CalcTool allows to use the following channel properties:

Syntax:

#Channel.Rate	à Samplingrate of the channel (i.e. 100 for a channel sampled with 100Hz)
#Channel.DT	à Samplingtime of the channel (i.e. 0.01 for a 100 Hz channel)
#Channel.P1	à First of the three parameters of a channel. Meaning depends on channel type
#Channel.P2	à Second of the three parameters of a channel. Meaning depends on channel type
#Channel.P3	à Third of the three parameters of a channel. Meaning depends on channel type

1.1.5. Calculated mileage

The 2D software system provides a possibility to document, for which distance several parts of a vehicle were used.

Normally this is done by the communication tool WinIt. When the software downloads a measurement from a logging device, it tries to find a channel giving the distance of the run. This distance is added to the mileage values of the permanent SpecsSheet corresponding to the downloaded logger.

When a part is replaced by a new one, you have to set the mileage value of that part to zero in the permanent file. With this you can always see in the permanent SpecsSheet file of the logger, for how many kilometers parts of your vehicle were used.

If you use a logging device, which does not have a distance channel, WinIt can not determine the distance of the run. In this case you can use the Mileage-function of CalcTool. This function integrates a given speed channel to the distance of the run and adds it to the mileage values in the permanent file corresponding to the given logger name.

Syntax:	Meaning
Mileage(#SpeedChannel, LoggerName)	Calculate the distance of a run using the channel "SpeedChannel" and update the mileage in the permanent file (SpecsSheet) of the logger with the name "LoggerName"



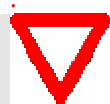
For more information about permanent files and SpecsSheets see the documentation of SpecView.

1.2 Postprocessor-functions



Postprocessor functions are done after the calculation of the channels.

1.2.1. Deleting channels



Function to delete channels, which were created by a previous calculation.
It is not possible to delete channels, which were measured by the data acquisition system

Syntax:	Meaning
DELETE(#CHANNELNAME)	It is possible to use the "*" as a wildcard character. Delete(#) deletes all channels created by CalcTool.

1.2.2. Hiding channels

Syntax:	Meaning
HIDE(#CHANNELNAME)	Function to hide a channel. A hidden channel is a channel, which is present but not visible in the analysing software.

1.2.3. Unhiding hidden channels

Syntax:	Meaning
UNHIDE(#CHANNELNAME)	Function to unhide a channel. If you want a hidden channel to become visible in the analysing software use this command.

2. Available functions



In the following chapters we will describe which functions are available and how to use them.

2.0 Format of numeric parameters

Some functions of CalcTool require one or more numbers as parameter.

There are three different possibilities to write a numeric parameter:

- **Decimal:** The number is written in normal decimal syntax; i.e. 350 or 17.5
- **Hexadecimal:** This notation starts with "0x" or "0h" and is followed by hexadecimal digits (0..F); i.e. 0xFF or 0hFF for the decimal value of 255. Only integer numbers can be written in this notation.
- **Binary:** This notation starts with "0b" and is followed by binary digits (0 and 1) i.e. 0b11111111 for the decimal value of 255. Only integer numbers can be written in this notation

2.1. Standard mathematical functions

2.1.1. Addition

With this function you can add both, a channel with a channel and a channel with a constant.

Syntax:	Meaning
+ (#CHANNEL_1, #CHANNEL2)	(#Channel_1) + (#Channel_2)
+ (#Channel, Constant)	(#Channel) + Constant

Practical examples:

- Add the preload to a suspension movement.
- Geometry calculations (Height of the vehicle ...).

2.1.2. Subtraction

With this function you can do three types of subtraction:

- Channel minus another channel
- Channel minus a constant
- Constant minus a channel

Syntax:	Meaning
- (#CHANNEL_1, #CHANNEL_2)	(#Channel_1) - (#Channel_2)
- (#Channel, Constant)	(#Channel) - Constant
- (Constant, #Channel)	Constant - (#Channel)

Practical examples:

- Subtract rear right suspension to the left one to know the rear roll of the car.
- Geometry calculations (Height of the vehicle ...).
- Calculation of the rear roll angle of the car by subtracting right angle and left angle calculated with arctan of (Suspension travel / rear track of the car).

2.1.3. Multiplication

As the addition, you can multiply both, two channels and a channel with a constant.

Syntax:	Meaning
* (#CHANNEL_1, #CHANNEL_2)	(#Channel_1) * (#Channel_2)
* (#Channel, Constant)	(#Channel) * Constant

Practical example:

- Multiply the suspension movement with the stiffness of the spring in order to give you an idea of the load on the suspension.
- Do the square of the speed (Speed²=Speed*Speed) to do special calculations (See the end of this chapter).

2.1.4. Division

Here you have 3 different sorts of formulas:

- Division of a channel by a channel
- Division of a channel by a constant
- Division of a constant by a channel

Syntax:	Meaning
/ (#CHANNEL_1, #CHANNEL_2)	(#Channel_1) / (#Channel_2)
/ (#Channel, Constant)	(#Channel) / Constant
/ (Constant, #Channel)	Constant / (#Channel)

Practical example:

- Convert speed from km/h into m/s (Speed_m = / (#Speed_k , 3.6)).
- Convert speed from km/h into mph (Speed_{mph} = / (#Speed_k , 1.62))

2.1.5. Modulo

The result of the modulo function is the rest of a division (i.e. 10 modulo 4 is 2)

CalcTool support three kinds of modulo functions:

- Channel modulo channel
- Channel modulo constant
- Constant modulo channel

Syntax:	Meaning
MOD (#CHANNEL_1, #CHANNEL_2)	(#Channel_1) Mod (#Channel_2)
Mod (#Channel, Constant)	(#Channel) Mod Constant
Mod (Constant, #Channel)	Constant Mod (#Channel)

Practical example:

- Mask out the lowest 5 bit of a channel (Lo5Bit=Mod(#Channel, 32)).

2.1.6. Integer division – DIV

The result of the DIV-function (integer division) is the quotient of a integer division without a rest (i.e. 10 DIV 4 is 2)

CalcTool support three kinds of DIV-functions:

- Channel div channel
- Channel div constant
- Constant div channel

Syntax:	Meaning
DIV (#CHANNEL_1, #CHANNEL_2)	(#Channel_1) Div (#Channel_2)
Div (#Channel, Constant)	(#Channel) Div Constant
Div (Constant, #Channel)	Constant Div (#Channel)

2.1.7. Rounding to whole numbers



CalcTool provides three different functions to round fractional numbers to whole numbers.

2.1.7.1. The Round function

The Round function goes to the next higher whole number if the fraction of the number is bigger or equal 0.5.

If the fraction is smaller than 0.5 it goes to the next smaller whole number.

Syntax:	Meaning
ROUND (#CHANNEL)	Calculates a channel with whole numbers

Example:

1.5 à 2
-1.5 à -1
1.49 à 1

2.1.7.2. The Floor function

The Floor function goes always to the next smaller whole number.

Syntax:	Meaning
FLOOR (#CHANNEL)	Calculates a channel with whole numbers

Example:

1.5 à 1
-1.5 à -2
1.99 à 1

2.1.7.3. The Trunc function

The Trunc function simply cuts the fractional part of the number.

Syntax:	Meaning
TRUNC (#CHANNEL)	Calculates a channel with whole numbers

Example:

1.5 à 1
-1.5 à -1
1.01 à 1

2.1.8. The Const function – CONST

The result of the Const-function is a channel with a constant value.

Syntax:	Meaning
CONST (CONST VALUE)	Creates a channel with constant value (=const value)

2.1.9. The minimum value function – MinValue

The MinValue-function searches all values of a given channel to find the minimum. It creates a channel with the constant value of this minimum.

Syntax:	Meaning
MinValue(#Channel, Rate)	Creates a channel with constant value. The value is the minimum value of channel "Channel". Rate defines the samplingrate with which the channel is stored.

2.1.10. The maximum value function – MaxValue

The MaxValue-function searches all values of a given channel to find the maximum. It creates a channel with the constant value of this maximum.

Syntax:	Meaning
MaxValue(#Channel, Rate)	Creates a channel with constant value. The value is the maximum value of channel "Channel". Rate defines the samplingrate with which the channel is stored.

2.1.11. The first value function – FirstValue

The FirstValue-function reads the first value of a given channel and creates a constant channel using this value.

Syntax:	Meaning
FirstValue(#Channel, Rate)	Creates a channel with constant value. The value is the first value of channel "Channel". Rate defines the samplingrate with which the channel is stored.

2.1.12. The last value function – LastValue

The LastValue-function reads the last value of a given channel and creates a constant channel using this value.

Syntax:	Meaning
LastValue(#Channel, Rate)	Creates a channel with constant value. The value is the last value of channel "Channel". Rate defines the samplingrate with which the channel is stored.

2.1.13. The average value function – AvgValue

The AvgValue-function calculates the average value of a given channel and creates a constant channel using this value.

Syntax:	Meaning
AvgValue(#Channel, Rate)	Creates a channel with constant value. The value is the average value of channel "Channel". Rate defines the samplingrate with which the channel is stored.

2.2. Logical functions

Logical functions are binary logical calculations.



These functions only work with original channels measured by the system (which are stored as word) or with channels, which were converted to a word channel in a previous calculation or line (see WORD-function).

2.2.1. AND function

There are two different formulas possible with the AND function:

- Channel AND channel
- Channel AND Constant

Syntax:	Meaning
AND(#Channel_1,#Channel_2)	(#Channel_1) And (#Channel_2)
AND (#Channel, Constant)	(#Channel) And Constant

Practical example:

- Mask out the lowest 8 bit of a channel (Lo8Bit=And(#Channel, 0xFF)).

2.2.2. OR function

Like the AND function, the OR function may be used with two channels or a channel and a constant:

- Channel OR channel
- Channel OR Constant

Syntax:	Meaning
Or (#Channel_1, #Channel_2)	(#Channel_1) Or (#Channel_2)
Or (#Channel, Constant)	(#Channel) Or Constant

2.2.3. XOR function

The Xor function supports two kinds of formulas:

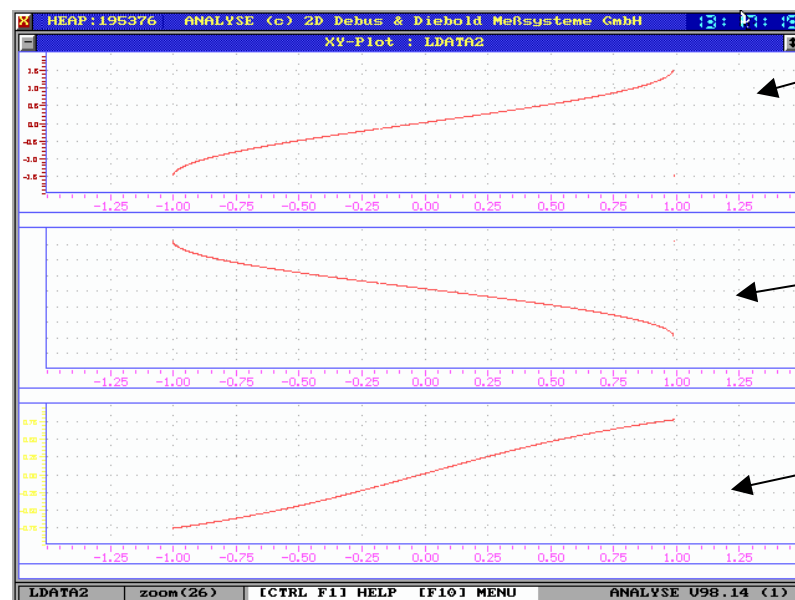
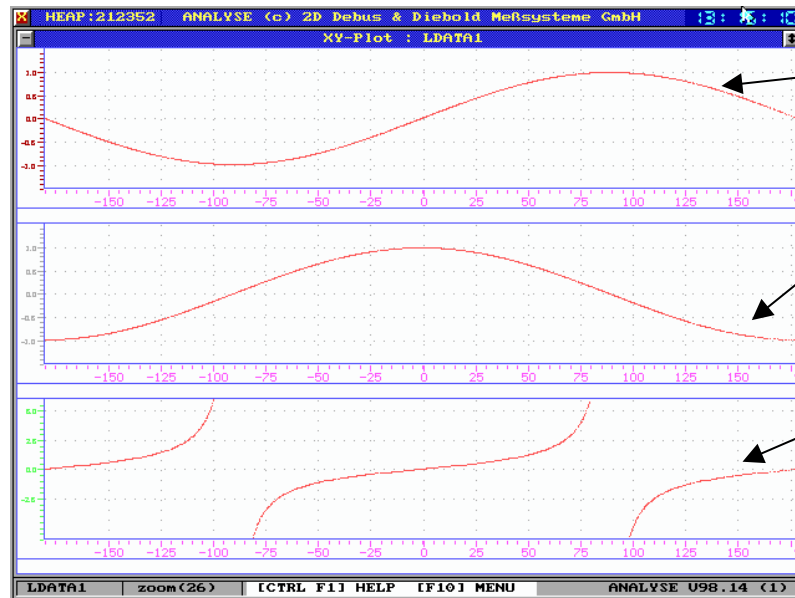
- Channel XOR channel
- Channel XOR Constant

<i>Syntax:</i>	<i>Meaning</i>
Xor (#Channel_1, #Channel_2)	(#Channel_1) Xor (#Channel_2)
Xor (#Channel, Constant)	(#Channel) Xor Constant

2.3. Trigonometric functions



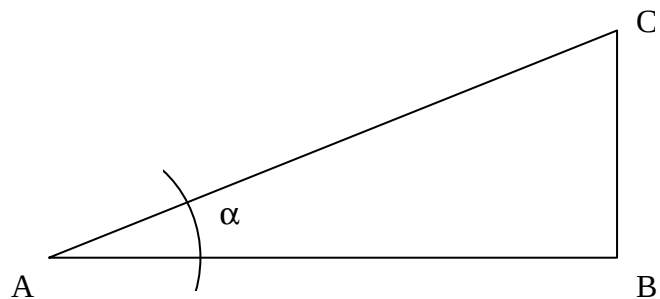
These functions allow you to calculate the sinus, cosinus, tangens, arcsinus, arccosinus and arctangens of a channel. The input for sinus, cosinus and tangens will be in radians or degrees, but the result for arcsinus, arccosinus and arctangens will be only in radians.



Syntax	Meaning	Comment
dsin(#Channel)	Sinus(#Channel)	(#Channel is in degrees)
dcos(#Channel)	Cosinus(#Channel)	(#Channel is in degrees)
dtan(#Channel)	Tangens(#Channel)	(#Channel is in degrees)
sin(#Channel)	Sinus(#Channel)	(#Channel is in radians)
cos(#Channel)	Cosinus(#Channel)	(#Channel is in radians)
tan(#Channel)	Tangens(#Channel)	(#Channel is in radians)
arcsin(#Channel)	ArcSinus(#Channel)	(Result is in radians)
arccos(#Channel)	ArcCosinus(#Channel)	(Result is in radians)
arctan(#Channel)	ArcTangens(#Channel)	(Result is in radians)

Practical examples:

- Calculation of a distance knowing a length and an angle.
- Calculation of an angle knowing 2 lengths.
- Calculation of the Rear roll angle (e.g. subtraction practical examples).



$$\sin(a) = BC / AC$$

$$\cos(a) = AB / AC$$

$$\tan(a) = BC / AB$$

$$a \text{ (in radians)} = \arcsin(BC / AC) = \arccos(AB / AC) = \arctan(BC / AB)$$

$$(a \text{ in degrees}) = 57,29 * (a \text{ in radians})$$

2.4. Advanced mathematical functions

2.4.1. Square Root

This function calculates the square root of any channel. The calculation is made with the absolute value of the channel (This means it also works with negatives values !!!).

Formula: $\text{sqrt}(\#Channel) \quad \Leftrightarrow \quad \sqrt{|\#Channel|}$

Practical example:

Calculation of combined G's for a car to define the total tractive effort available for braking while turning:

$$\text{CombinedG's} = \sqrt{(\text{LateralG's})^2 + (\text{LongitudinalG's})^2}$$

2.4.2. Logarithms

This function allows you to calculate the logarithms from any channel to the base of 10.

Formula: $\text{Log}(\#Channel) \quad \Leftrightarrow \quad \text{Log}_{10}(\#Channel)$

2.4.3. Absolute value

This function gives the absolute value of the calculated channel.

Syntax:	Meaning
abs(#Channel)	$ \#Channel $

Practical example:

Compare 2 channels without considering the sign of these channels (e.g. KUS example at the end of the chapter).

2.4.4. Minimum function

Syntax:	Meaning
min(#Channel_1,#Channel_2)	The result of this function is the minimum of 2 compared channels.

Practical example:

- Comparing car left and right rear suspension and take the minimum of these 2 channels.

2.4.5. Maximum function

Syntax:	Meaning
max(#Channel_1,#Channel_2)	The result of this function is the maximum of 2 compared channels.

Practical example:

- Comparing car left and right rear suspension and take the maximum of these 2 channels.

2.4.6. Derivation

It allows you to derive a channel to measure its speed or its acceleration.

Syntax:	Meaning
Derivate(#Channel)	d(#Channel) / dt

Practical example:

- Measure the longitudinal acceleration by deriving the speed of the vehicle.
- Measure the speed of a fork movement by deriving the travel of the fork.



Warning

Each time you derive a channel which unit is km/h, the result is automatically in m/s².

2.4.7. Integration

It allows you to integrate a channel to measure its speed, its motion or its energy.

Formula: Integrate (#Channel, P1) $\hat{=}$ $\int (\#Channel) dt$

Parameter:	Meaning
P1:	Start value, if this value is present the resulting function is set to this value every lap trigger. If this value is <u>not</u> present, the resulting is the integration over the whole measurement

Practical example:

- Make a counter on a channel (Combination of Integration and If functions) such as detonation or over rev. (See example at the end of the chapter)
- Calculate energy of a channel (e.g. on brake signal to know energy spent by lap).



When you integrate a function, the result is the integration of this function plus a constant. That's why there could be a little difference within the result. The best example of this is to derive the speed then integrate this acceleration. Normally you should get back to the speed, but if you look at the result, there is a small travel of your recalculated speed due to this integration's constant. (if you wish to do this exercise, do not forget to multiply the result of your integration by 3,6 to get back to a speed in km/h.)

2.4.8. SUM-function

This function calculates the sum of all values of a channel.

Formula: $\text{Sum}(\#Channel, P1) \quad \Leftrightarrow \quad \sum Channel[i]$

Parameter:	Meaning
P1:	Start value, if this value is present the resulting function is set to this value every lap trigger. If this value is <u>not</u> present, the resulting is the sum over the whole measurement

Practical example:

- Counting events: Create a calculation channel with the if-function, that receives the value 1 when a channel has a value greater than a border otherwise the value 0.
- Use the SUM function to count these events.

2.4.9. Power-function

Power raises a base to any power.

Here you have 3 different sorts of formulas:

- Raise a channel as base to power from an other channel
- Raise a channel as base to a constant power
- Raise a constant base to a power from a channel

Syntax:	Meaning
Power (#Base, #Exponent)	#Base ^ #Exponent
Power (#Base, Exponent)	#Base ^ Exponent
Power (Base, #Exponent)	Base ^ #Exponent

2.4.10. Exp-function

Exp calculates the value of e (euler-constant) raised to the power of a channel.

Syntax:	Meaning
Exp (#Exponent)	2.71828 ^ #Exponent



This function can be replaced by the previous function Power and the use of the preprocessorconstant @E: Power(@E, #Exponent)

2.5. If-function

This function is used to compare one channel with a constant or an other channel (or a constant with a constant). The result of this comparison is a constant (channel with constant value) or a channel.

Formula:

IF(#CHANNEL, OPERATOR, COMPARATOR, TRUERESULT, FALSERESULT)

#CHANNEL	Name of the channel to compare
Operator	Compare function: >, >=, <, <=, =
Comparator	Name of channel to compare #Channel with constant to compare with
Or	constant
TrueResult	Name of channel. It's value is taken, when the comparison is true
FalseResult	Name of channel. It's value is taken, when the comparison is false
Or	constant

Theoretical examples:

If (#Channel_1, <=, #Channel_2, A, B)
 If (#Channel_1, <=, #Channel_2, #Channel_3, #Channel_4)
 If (#Channel, =, C, A, B)
 If (A, =, B, C, D)

Practical example:

Put conditions on a channel and get a logical one (1 or 0) for further calculations (e.g. Kus, radius, speed2 examples)

2.6. Additional formulas

2.6.1. Formula Integer

With this function, you can use any channel with another formula. Here the integer formula is used:

$$F(x) = M * (x - O) / D$$

Syntax:	Meaning
F(#Channel, I(M,D,O))	set integer formula of channel #Channel
M	multiplicand
D	divisor
O	Offset

Practical examples:

- Use channel #Channel with another offset if zero position was not correct.
- Use channel #Channel with another formula if channel calibration was not correct.

2.6.2. Formula Real

As the above formula, you can use any channel with another formula. Here the float formula is used:

$$F(x) = F * x + O$$

Syntax:	Meaning
F(#Channel, R(F,O))	set real formula of channel #Channel
F	Factor (Multiplicand / Divisor)
O	Offset

Practical examples:

- Use channel #Channel with another offset if zero position was not correct.
- Use channel #Channel with another formula if channel calibration was not correct.

2.6.3. Table function

This function allows you to connect a table to a channel knowing the formula between the channel you want to use and digits. The result is the calculation of a channel before the formula is used on this channel. Indeed, the calculation is made from the digits or the voltage of the table.

Syntax:	Meaning
F(#Channel, T(Name))	#Channel will be connected to the table "Name"
Name	name of the table file

Practical examples:

- Obtain the power of the engine with RPM due to a table linking the power with RPM (e.g. Rforce example at the end of the chapter).
- Generate a linear channel from a non-linear one with the calibration sheet you get from the manufacturer.(e.g. Lambda sensor)

2.6.4. Table 2 function

With this function, you can connect a table to a channel like the table function above. The difference with the other table function is that the calculation is made with the result of the channel's formula. In fact, this table is linked directly to the data that are on your screen in the analysis mode whereas, with the table function above the table, is linked to the data recorded on the vehicle before they are translated in physical values by the channel's formula.

Syntax:	Meaning
F(#Channel, 2(Name))	#Channel will be connected to the table "Name"
Name	name of the table file

2.6.5. 2D-Table function

Sometimes it could be necessary to use a table not only depending from one channel but two.
For this use the 2D table function.

Syntax:	Meaning
F(#Channel1, #Channel2, T(TableName.Ext, IntType))	#Channel1 and #Channel2 will be connected to the table "TableName.Ext" name of the table file with extension
TableName.Ext	Type of interpolation (i.e. LIN)
IntType	



Attention:

If you do not write the extension of the table file in the line, CalcTool tries to read a .CSV – file.

This function needs a table file in the format CSV (comma separated values). This is a simple textfile where the single values are separated by a certain character. The separation character has to be a “;”. The decimal separator for the numbers in the file has to be a “.” (1/2 is 0.5 not 0,5 like in Germany). This format can be created with Microsoft Excel™.

Practical example:

The radius of a wheel of a motorcycle is dependant from the banking angle of the bike and it's speed. The higher the banking angle is the lower is the radius of the wheel (in fact the size of the wheel is the same, but radius to the point where the wheel touches the ground becomes smaller). The faster a wheel turns the bigger gets the wheel by the rotation power.

The formula for this calculation is:

Result = F(#Speed, #Banking, T(WheelRadius.CSV, Lin))

	0	50	100	150	200	250	300	350
0	315	317	320	322.5	325	327	329	330
10	309	311	314	316.4	318.8	320.7	322.7	323.7
20	301	302.9	305.8	308.2	310.5	312.5	314.4	315.3
30	295	296.9	299.7	302	304.4	306.2	308.1	309
40	288	289.8	292.6	294.9	297.1	299	300.8	301.7
50	285	286.8	289.5	291.8	294	295.9	297.7	298.6
60	283	285.5	288	290	392.5	394.3	296.1	297

This is an example for a table giving the radius of a wheel from the speed and the banking angle of a bike.

This is the view like inside Microsoft Excel™.

In a texteditor the file would look like this:

```
;0;50;100;150;200;250;300;350;
0;315;317;320;322.5;325;327;329;330;
10;309;311;314;316.4;318.8;320.7;322.7;323.7;
20;301;302.9;305.8;308.2;310.5;312.5;314.4;315.3;
30;295;296.9;299.7;302;304.4;306.2;308.1;309;
40;288;289.8;292.6;294.9;297.1;299;300.8;301.7;
50;285;286.8;289.5;291.8;294;295.9;297.7;298.6;
60;283;285.5;288;290;392.5;394.3;296.1;297;
```

In the first row are the column headers with the different speed values in km/h.
The first column contains the row headers for the different banking angles of the bike in °.
In this example the distance between two row or column headers is the same. This is not needed.

Finding the radius at a speed of 100 km/h and a angel of 20° is easy. We have to look in the field where the column is 100 and the row is 20. The result is 305.8.
When CalcTool tries to find a value depending on speed and banking angle it searches for the columns where the speed is between and for the rows where the banking angle is between and makes a linear interpolation.

Example:

Radius for banking angle 33° at a speed of 110 km/h.
The columns for the interpolation are 100 and 150.
The rows are 30 and 40.
The resulting value would be 298.03.

When the input values are outside the table, for example a speed of 360 km/h, CalcTool uses the values from the edge. In the example it would use the column of 350 km/h.

2.6.6. 2D-Table search function

CalcTool provides a kind of inverse function for the 2D-Table function. It uses the same type of table files like the function above.

The function T2DSearch needs two channels as input parameters. The first is the Y-channel and is giving values in Y-direction of the table. The second is the value-channel giving values from the inner area of the table.

The values of the resulting channel are the values in X-direction, which would lead to the Value channel when used together in the 2D-Table function.

The T2DSearch function interpolates linearly between given values.

Syntax:	Meaning
T2DSearch(#YChannel, #ValueCh, T(TableName.Ext, IntType) [, SearchDirection])	#YChannel and #ValueCh will be used to find the X-values in the table "TableName.Ext"
TableName.Ext	name of the table file with extension
IntType	Type of interpolation (i.e. LIN)
SearchDirection	Optional parameter to define the search Direction: LeftToRight or RightToLeft. If no search direction is given LeftToRight is used

Example:

For the example we use the same table as in the chapter before (Even if it does not make much sense).
For the banking angle of 33° and a wheel radius of 298.03 mm the corresponding vehicle speed is 110 km/h.

2.6.7. 3D-Table function

Similar to the 2D-Table function CalcTool provides a 3D-Table function.
This function can be used, when a resulting channel is dependant from three channels.

Syntax:	Meaning
F(#Ch1, #Ch2, #Ch3, T(TableName.Ext, IntType))	#Ch1, #Ch2 and #Ch3 will be connected to the table "TableName.Ext" name of the table file containing an "*" with extension
TableName.Ext	Type of interpolation (i.e. LIN)
IntType	

Since a normal table like inside Excel can only be used for two dimensions (channels) more than one table file is needed.

The third dimension is done by using several .CSV-files. Each file is for one value of the third channel. The value for which the file should be used has to be part of the filename of the channel.

Practical example:

The torque an engine delivers to a wheel is dependant on the selected gear, the engine revolutions and the throttle position.

Result = F(#Gear, #RPM, #Throttle, T(Torque_at_*.CSV, Lin))

You can see in the example that the filename of the table contains an asterisk ("*"). At this position of the filenames has to be the throttle value for which this file should be used.

Torque_At_0.CSV, Torque_At_20.CSV, Torque_At_40.CSV, Torque_At_50.CSV, Torque_At_60.CSV, Torque_At_70.CSV, Torque_At_80.CSV, Torque_At_90.CSV and Torque_At_100.CSV would be the tablefiles for the throttle position of 0, 20, 40, 50, 60, 70, 80, 90 and 100 %.

CalcTool makes a linear interpolation for each value of throttle, gear and rpm.

2.6.8. Filter function

Filters are used to smoothen a noisy signal or to adopt information about the dynamic movement of a signal (suspension, etc.). With this function you can calculate filtered channels.

CalcTool provides two modes of defining a filter. The first is a short form where the user only has to define the filter-frequency, for the other values of the filter definition, default values are taken. In most cases these values lead to the wanted result.

If you do not receive the result you wanted and you are familiar with FIR (finit impulse response) filters you may use the full filter interface of CalcTool.

2.6.8.1. Simple filter interface

With this interface you define a FIR-filter (finit impulse response) with the following default settings.

The type is a lowpass filter.

The windowtype is a hamming window.

The number of iterations is either 20 times the quotient of the samplingrate and the crossfrequency (given by the user) or a default value of 401, the smaller value is taken. (401 or $20 * \text{Samplingrate} / \text{CrossFrequency}$)

Syntax:

F(#Channel, F(CrossFrequency))

Example:

F(#Channel, F(2.0))

Assuming a samplingrate of 25 Hz, the speed channel #Channel is filtered with a Hamming window, a crossfrequency of 2.0 Hertz and the number of iterations is 251 (calculated: 250 but the iterations have to be odd).

2.6.8.2. Full filter interface



The full filter interface of CalcTool gives you the possibility to define all parameters of a filterfunction.

If you want to use a default value as described above, you can leave away that part of the definition. Here is the formula with all possible parameters.

With the full filter interface the number of iterations (dept of the filter) can be given directly as a number or as a time. When CalcTool finds a number followed by "Sec" or "s" at the position of the iterations, it calculates the iterations from the given time value using the current sampling rate of the channel.

Example:

F(#Channel, F(AVG 0.5 Sec))

Will calculate an center average filter with 50 iterations for a 100 Hz channel.

Syntax:

F(#Channel, F(FilterType \ WindowType \ CrossFrequency \ Number Of Iterations))

Filter Type:

L_P	Lowpass filter
H_P	Highpass filter
B_P	Bandpass filter
AVG	Center average filter (Average built from NumberOfIterations of values, half left of the current value and the other half right of the value)
AVG+	Look ahead average filter (Average built from NumberOfIterations of values from the current value to the right)
AVG-	Look back average filter (Average built from NumberOfIterations of values from the current value to the left)
MED	Median filter
TOP	Maximum filter (like median, but using the maximum value from the given interval)
MAX	Same as TOP
BOT	Minimum filter (like median, but using the minimum value from the given interval)
MIN	Same as BOT
TAU	Tau filter according to formula $X_{Filtered}(t) = X_{Filtered}(t-1) + 1/Iterations * (X(t) - X_{Filtered}(t-1))$

Window Type:

None	Rectangle
Hann	Hanning
Hamm	Hamming
Black	Blackman
Kaib	Kaiser Bessel
Parz	Parzen
Welch	Welch

Cross frequency: 2 Hertz will be written in the formula: 2.0Hz

Number of iterations: Select the numbers of elements for the FIR (Finite Impulse Response) algorithm.

Practical example:

- If you want to make calculation with filtered data, just insert a calculation channel in your string group. By this way, you don't have to go to filter module in the analysis mode before your calculations. So, you can gain time.
- Handling both raw and filtered data at same time in analysis mode.

Example:

F(#Channel, F(L_P Hann 2.0Hz 100))

The speed (channel #Channel) is filtered with a lowpass filter with a Hanning window, a cross frequency of 2.0 Hertz and the number of iterations is 100.

2.6.8.3. Filter with variable depth

In some cases it might be useful not to use a fix depth for filtering but a variable.

For this cases it is possible to define a filter with a variable filter depth. The depth of the filter is given as a channel. This is not defined for finit impulse response filters.

Syntax:

F(#Channel, F(FilterType #DepthChannel))

Filter Type:

AVG	Center average filter (Average built from NumberOfIterations of values, half left of the current value and the other half right of the value)
AVG+	Look ahead average filter (Average built from NumberOfIterations of values from the current value to the right)
AVG-	Look back average filter (Average built from NumberOfIterations of values from the current value to the left)
MED	Median filter
TOP	Maximum filter (like median, but using the maximum value from the given interval)
MAX	Same as TOP
BOT	Minimum filter (like median, but using the minimum value from the given interval)
MIN	Same as BOT

Example:

C1 = I(#RPM, 250)

Result = F(#Susp_F, F(AVG #C1))

The example shows an center average filter, where the depth of the filter is dependant on the temporary channel "C1". This channel is calculated to 1/250 of the RPM-signal of the vehicle in the first line. The higher the RPM signal of the engine the deeper is the filter.

2.6.9. Frequency reduction/increase



This function allows you to change the sampled frequency of a channel. You can reduce the sampled frequency (You will lose information) or increase the frequency (You will not get more information).

Syntax:	Meaning
Freq(#Channel, NewRate)	Change samplingrate of Channel #Channel to NewRate
#Channel	channel name
NewRate	new sample rate

Practical example:

To make complicated calculations you need normally a lot of memory. If this memory is not available you can reduce the channel-size. For NewRate not every value is possible. Valid values are basic samplingrate (bs), bs divided by 2, divided by 2

2.6.10. Sampling rate adjustment for non 2D samplingrates: FreqBase

If you use a 2D device to record data from a can bus, it might be that the refresh rate of the device sending the data is not one of the possible recording rates of the logger.

For example a GPS sends data with 4 Hz, but the logger recording this channel has only the following samplingrates to select from: 400 Hz, 200Hz, 100 Hz, 50 Hz, 25 Hz, 12.5 Hz.

If the channel is recorded with 50 Hz and the GPS sends data with 4 Hz, the resulting channel should have always 12.5 times the same value ($50 / 4 = 12.5$). Since it is not possible to have a value a half time, the channel has 12 or 13 identical values. If such a channel is used for further calculation, the results will show some aliasing.

To solve this use the function FreqBase.

This function changes the samplingrate of a channel and provides a channel where the number of identical values is always the same.

Syntax:	Meaning
FreqBase(#Channel, NewRate, GPSRate)	Changes samplingrate of channel #Channel to NewRate
#Channel	Name of the channel
NewRate	New samplingrate
GPSRate	Rate at which the GPS sends new data



Attention:

The value for NewRate has to be less or equal to the main samplingrate of the logger.
The second restriction is, that NewRate has to be divisible by the GPSRate without rest.

Practical example:

The GPS sends speed data (V_SAT) with 4 Hz. The data is recorded with a samplingrate of 50 Hz.

The new rate has to be a samplingrate of the logger and has to be divisible by 4, the lowest possible rate here is 100:

FreqBase(#V_SAT, 100, 4)

2.6.11. Demultiplexing a multiplexed channel: MUX-function

To reduce the traffic on the can bus sometimes more than one channel is sent on a single identifier on the same byte position. On an other position of the identifier there are some bits defining how to interpret the current value. Record the bits containing the data as one channel (DataChannel), and the bits defining how to interpret the value as the second channel (MuxChannel).

Syntax:	Meaning
MUX(#DataChannel, #MuxChannel, MuxVal)	Extracts the values of DataChannel when MuxChannel has the value MuxVal
#DataChannel	Name of the channel containing the multiplexed data
#MuxChannel	Name of the channel containing the multiplex values
MuxVal	Value at which the data should be taken

Practical example:

The channel "DataChannel" contains multiplexed data for water- and oil-temperature.
The "MuxChannel" contains either 0 for water- or 1 for oiltemperature.

WaterTemp=MUX(#DataChannel, #MuxChannel, 0)

OilTemp=MUX(#DataChannel, #MuxChannel, 1)

2.6.12. Masking out used bits: UseBits-function

To use as few identifiers as possible it is usual to aggregate more than one channel in a 8, 16 or 32 bit value on the CAN bus. With 2D devices it is only possible to record one or two bytes from the CAN bus.

Sometimes it is necessary to mask out only several bits from the recorded channel, this can be done by using the UseBits-function.

Syntax:	Meaning
UseBits(#DataChannel, MaskValue)	Extracts the values of DataChannel using MaskValue to mask out the used bits.
MaskValue	Value to mask out used bits

Practical example:

The channel "DataChannel" is a 16 bit channel recorded from the CAN bus. The four least significant bits represent the selected gear of the vehicle. The 12 most significant bits represent the speed.

Gear=UseBits(#DataChannel, 0b1111)

Speed=UseBits(#DataChannel, 0b1111111111110000)

written in hexadecimal notation:

Gear=UseBits(#DataChannel, 0xF)

Speed=UseBits(#DataChannel, 0xFFF0)

2.6.13 GPS Shift



Usable for all GPS modules which records additionally the channel named as "T_Shift" (besides the usual used channels).
This channel indicates, how much milliseconds ago the values of the other channels were valid.



To use this function you must have the following software releases:
Software à CalcTool.exe (version 2008.2.8.2.11 or higher)
Editor à Calcredit.exe (version 2008.2.4.5.6 or higher)

Syntax:	Meaning
GPSShift(#GPSChannel, #ShiftChannel, NewFreq)	#GPSChannel: The first parameter of this function represents a GPS channel like V_Sat, Lat_DeZ... (for example) #ShiftChannel: The second parameter is the channel "T_Shift" (mentioned above). NewFreq: The last parameter represents the frequency (sampling rate) with which the result channel is to be created.



Note:

- The higher the frequency is, the Shift function works more exactly (it must be always rounded on whole scans).
- A value larger then the main sampling rate resulted in a warning in the protocol file; in this case the main sampling rate will be used.
- Result channels are always channels of type float (double if the GPS channel was double, otherwise channels with a single precision)
- Between the "shifted" GPS values an linear interpolation will be done.

2.7. Bike and Car physical formulas

2.7.1. Speed

The logger calculates the speed by using the digital signal from the speed sensor, the tire-dimension and the number of pulses for this wheel. If the wheel turns very slow, the logger can not calculate the correct speed because of the big time-distance between two pulses.

This becomes a problem when you wish to measure speed up to 0. For example to make breaking tests. The signal looks very raw and the values close to 0 are wrong.

The solution for this is a speed post calculation.

In this case the formula is

Syntax:	Meaning
Speed_L4 (#Channel)	#Channel: input channel (Normally a speed channel)

Practical example:

Speed_L4(#VRear)



Restrictions:

- Do not use this function on filtered values.
- Do not use this function in combination with tables.
- To measure speed till 0 means also that the car or bike must stand still some seconds at 0 km/h, so wait some seconds before you continue to accelerate. In some cases (without RPM channel) it's better not to start logger with 'AUTO MODE' because in this case the logger does not record when the speed is 0.

2.7.2. Speed2



This function is used to calculate a major speed-channel from two separately measured speeds. The result is the average of the two speeds. If the difference between the two channels is greater than 10%, the resulting value is the greater signal.

<i>Syntax:</i>	<i>Meaning</i>
Speed2 (#Channel_1,#Channel_2)	#Channel_1, #Channel_2: input channels (Normally two speed channels)

Practical example: Speed2(#VFront, #VRear)



Restrictions:

- Do not use this function on filtered values.
- Do not use this function in combination with tables.
- Use this function only with speed-channels.

2.7.3. Slip



This function allows you to calculate the percentage of the vehicle's slip. For this it is necessary to measure both rear and front speed.

<i>Syntax:</i>	<i>Meaning</i>
Slip (#V_Front, #V_Rear)	
#V_Front	Name of front speed channel
#V_Rear	Name of rear speed channel

2.7.4. Banking



This function has to be used to calculate the banking angle of the bike with the Gyro channel.

As you know the gyro measures the banking angle speed of the bike. So, if you integrate this signal, you should get the inclination of the bike. But with this kind of sensor, a small error can lead to a totally wrong result.

For example, considering we've got a gyro measuring a range of 300° (from -150° to 150°) with 4095 bits. So 0° will be at 2047 or 2048 bits. This means there is an error of ½ bit. Your lap lasts 2 minutes and your sampling rate is 50 Hz. Therefore, you have 6000 samples per lap. Each sample, you have, as we said, an error of ½ bit. So your total error is ½ * 6000 = 3000 bits. But for 1 bit you have approximately 0,1° (300/4095) of error. Finally your error will reach 300° by lap.

To correct this, the function integrates the gyro's signal with a linear regression to avoid that the result is modified by the error explained above.

Syntax:	Meaning
Banking (#AGyro, P1)	
#Agyro	Channelname of the gyroscope channel.
P1	Angle of the bike on the start/finish line. (0.0 if the bike is straight). If you think the bike is not straight on this part of the circuit, you will have to estimate the banking. For example you can do some measurements with 0.0. Then you have to make an average on the same type of corners but with opposite directions : You can have one at 76° and the other one at -40°. So you will get a difference in absolute of 36°. Therefore P1 can be 18 (36/2). This parameter is important because, as you have seen with the integration, the calculation is reset to P1 each lap.

Practical example:

- Calculation of the inclination of the bike.

2.7.5.BANK2ACC function

As you know it is not worth measuring lateral acceleration on a bike. However to use the racetrack calculation function, you will need a "virtual" lateral acceleration. This function calculates the lateral acceleration of the bike from the inclination of the bike calculated with banking function.

Syntax:	Meaning
BANK2ACC (#Abanking)	#Abanking is the channel where the inclination of the motorcycle is calculated with the banking channel

Practical example:

- Calculation of the racetrack.

2.8. Channel manipulation

2.8.1. Shifting a channel

This function allows to shift a channel on the time-axis. The unit for the shift may either be seconds or scans. One scan is the time between two measured values of the channel. That means if the channel is recorded with 50Hz one scan is $1/50\text{Hz} = 0.02\text{ sec}$. Use a positive value to shift the channel to the right and a negative for the other direction. If you want to shift by time, simply add the unit "sec" to the number of the second parameter.

Syntax:	Meaning
Shift(#Channel, ShiftCount)	
#Channel	input channel to shift
ShiftCount	Number of scans or seconds to shift the channel

Example:

Shift(#VFront, -2 sec) Channel VFront is shifted two seconds to the left

Shift(#VFront, 100) Channel VFront is shifted 100 values to the right (Assumed that the samplingrate is 50 Hz that means 2 sec)

2.8.2. Combining two 16-bit channels

Some 2D Data Acquisition Systems provide the possibility to record channels, which are read from CAN-bus. This bus is often used in cars and motorcycles to connect sensors to the motor electronic. With 2D systems it is possible to record the values from this bus. CAN-channels in a 2D system have a maximum resolution of 16 bit but the data of the motor electronic may have a resolution of 32 bit. The solution for this problem is to store a 32 bit channel divided in two 16 bit channels. After the data is read from the recording device these two channels are combined to one 32 bit channel using the function Make32, MakeGPS or MakeSingle. If the data read from the CAN-bus originally was a 32 bit integer use Make32 or MakeGPS to create a integer channel, if it was a 32 bit float (IEEE) then use MakeSingle to create a single precision float channel.

The functions Make32 and MakeGPS are very similar. The difference is that MakeGPS changes the range of the values from 0 to 360 to -180 to +180 by subtraction, if the values of the original channel were in the range 0 to 360.

First we have a look at the integer function Make32:

The resulting channel receives the formula of the channel with the lower 16 bit.

Syntax:	Meaning
MAKE32(#ChannelHi, #ChannelLo)	
#ChannelHi	input channel containing the high 16 bit
#ChannelLo	input channel containing the low 16 bit

Practical example:

Combining two 16 bit channels from a GPS to the global latitude:

MAKE32(#LatituHi, #LatituLo)

If you want to create a 32 bit channel with automatic change of the value range to fit in -180° to +180° use the function MakeGPS.

Again the resulting channel receives the formula of the channel with the lower 16 bit.

Syntax:	Meaning
MakeGPS(#ChannelHi, #ChannelLo)	
#ChannelHi	input channel containing the high 16 bit
#ChannelLo	input channel containing the low 16 bit

Practical example:

Combining two 16 bit channels from a 2D GPS to the global longitude:

MakeGPS(#Long_deHi, #Lon_deLo)

The formula to create a single precision float channel from two 16 bit channels is MakeFloat:

Syntax:	Meaning
MakeSingle(#ChannelHi, #ChannelLo)	
#ChannelHi	input channel containing the high 16 bit
#ChannelLo	input channel containing the low 16 bit

2.8.3. Combining two 32 bit channels

If channels are recorded from the CAN-bus (see chapter before), it might be possible, that the data sent by a CAN-device has a resolution of 64 bit. Since 2D systems store information from the CAN-bus in 16-bit channels, you have to combine four of these to one 64 bit channel. This has to be done in two steps, the first is to combine four 16 bit to two 32 bit channels, using the Make32-function, the second is, to combine these two channels to one with 64 bit. The 64 bit number on the bus can be either a integer number or a double precision float. For integer use the function Make64 for the float use MakeDouble.

<i>Syntax:</i>	<i>Meaning</i>
MAKE64(#ChannelHi, #ChannelLo)	
#ChannelHi	input channel containing the high 32 bit
#ChannelLo	input channel containing the low 32 bit

<i>Syntax:</i>	<i>Meaning</i>
MakeDouble(#ChannelHi, #ChannelLo)	
#ChannelHi	input channel containing the high 32 bit
#ChannelLo	input channel containing the low 32 bit

Practical example:

Combining four 16 bit to a double precision channel:

[Double]

C1=Make32(#LoHi, #LoLo)

C2=Make32(#HiHi, #HiLo)

Result=MakeDouble(#C1,#C2)

2.8.4. Changing the storage type

Most channels in a 2D system are recorded as 16 bit integer values (what is called a word) combined with a multiplier and an offset. Channels created by the calculation tool are stored as float values (single precision with 32 bit) which need the double space on the harddisk of the computer. With the Word- and the Binary-function it is possible to change the store type of a channel to word, to reduce the storing space of a measurement. But do not forget, that this lowers the precision of the channel.

A second reason to use these functions is that some other functions expect a word channel for input. If the channel you want to use in such a calculation is the result of a previous calculation, you have to change the store type before using the Word- or the Binary-function.

2.8.4.1. WORD-function

If the Word-function is used with only one parameter (a channel), CalcTool determines the minimum and the maximum value of the original channel. The formula is calculated that the minimum is transformed to 0 and the maximum to 65535 (maximum word).

In some cases it would be helpful if other values would be used. If the signal has spikes you might use your own border values.

Values lower than the lower border are set to this border.

Values higher than the upper border are set to the upper border value.

The third possibility to use the Word-function is to give the step (the accuracy) which should be used for one digit. The minimum value for the resulting channel is 0 and the maximum 65535 multiplied with this step.

Example: Word(#Speed, 0.01)

The resulting channel has a accuracy for 0.01, the minimum value is 0 and the maximum is 655.35

Syntax:	Meaning
Word(#Channel)	
#Channel	channel to change the store type
Word(#Channel, LowerBorder, UpperBorder)	
#Channel	channel to change the store type
LowerBorder	All values of the original channel which are lower than this border are set to this border,
UpperBorder	All values of the original channel which are higher than this border are set to this value
Word(#Channel, Step)	
#Channel	Channel to change the storage type
Step	Step representing the value of one digit

2.8.4.2. BINARY-function

The Binary-function is very similar to the Word-function. The difference is the parameter Range. The Range gives the maximum number of different values in the resulting channel.

A binary-function with the range 65536 has the same result as a word function would have.

Syntax:	Meaning
Binary(#Channel, Range)	
#Channel	channel to change the store type
Range	Maximum number of different values in resulting channel
Binary(#Channel, Range, LowerBorder, UpperBorder)	
#Channel	channel to change the store type
Range	Maximum number of different values in resulting channel
LowerBorder	All values of the original channel which are lower than this border are set to this border
UpperBorder	All values of the original channel which are higher than this border are set to this value

Practical example:

Binary(#Acc_x, 4096, -50, 50)

The resulting channel is a word channel with a formula that calculates 0 digits to -50 (so the offset will be -50) and 4095 digits to +50 (so the multiplier will be $100/4095 \rightarrow 0.02442$)

2.8.4.3. Signed function



The source channel has to be a unsigned Integer channel, that means Byte, Word or Double Word (=DWord) format

Syntax:	Meaning
Signed(#Channel)	Channel to change the data format
#Channel	The function changes the data format from unsigned to signed.

Example:

Byte:	0..255	à	-127..128
Word:	0..65535	à	-32767..+32768
DWord	0..4294967295	à	-214748367..+214748368

2.8.5. SWAP-function – Changing the byte-order of a channel

Some 2D Data Acquisition Systems provide the possibility to record channels, which are read from CAN-bus. This bus is often used in cars and motorcycles to connect sensors to the motor electronic. The user has to define in the communication software which bytes should be taken from a single CAN-identifier to build the channel data. If the byte order is incorrect the resulting channel will be unusable. The SWAP-function was designed to change the byte order of such channels. It expects a word or longint channel as input. (16 or 32 bit integer)

Syntax:	Meaning
Swap(#Channel) #Channel	channel to change the byte order

Practical example:

- Swap the bytes of a CAN-channel, that was recorded with the wrong byte order

2.8.6. Changing dimension

Normally CalcTool does not calculate the dimensions of channels, except integration and derivation of speed and distance. A resolution channel receives the dimension of the first input channel. In some cases this dimension is not correct. To change the dimension of a channel use the Set-function.

Syntax:	Meaning
SET(DIM=Value) DIM	Name of the property to set (in the moment only DIM is possible)
Value	Value to set the property to

Practical example:

Set the dimension of a calculated gear channel to 'Gear':

C1=...

C1=Set(DIM='Gear')

2.9. Special functions for phase-calculation

In the analysing software it is necessary to reduce the volume of the data to make special analysis (e.g. MinMax-tables, XY-plots, etc.). Often not all data is of interest, but the fastest lap, a special corner or when the values of a channel are in a special range. The reduction of the data is defined by so called phase-conditions. It is possible to combine several phase-conditions. A phase is one part of the measurement where the phase-conditions meet.

The following functions are used to calculate channels, which are used for phase conditions.

2.9.1. MaxHold - function



This function stores the maximum of a channel. The maximum is kept until the value of an other channel (the trigger channel) goes below one border value after it was above an other border value.

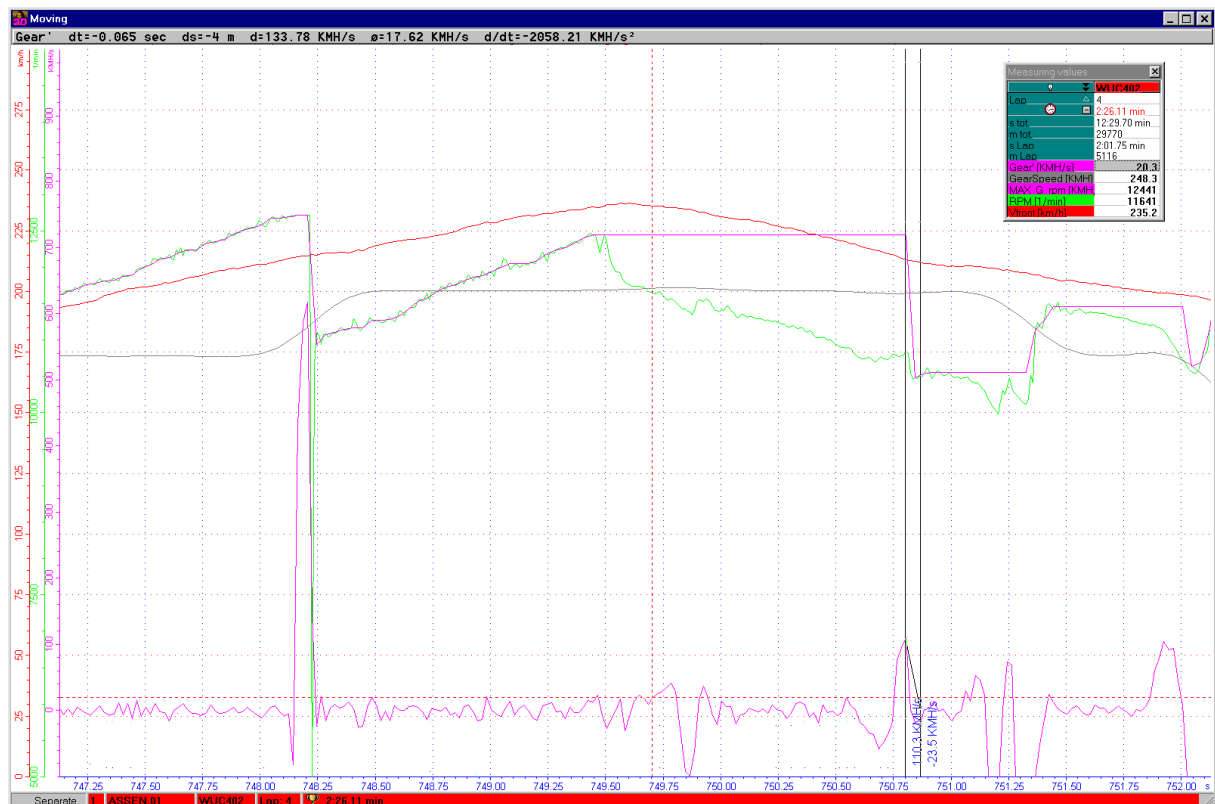
Syntax:	Meaning
MaxHold(#CheckChannel, #TriggerChannel, Border1, Border2)	
#CheckChannel	Channel to check for the current maximum
#TriggerChannel	Channel that triggers the reset of the maximum
Border1, Border2	Constants for the trigger conditions When the value of the trigger channel goes below Border2 after being above Boder1, the maximum is reset to the current value of the channel CheckChannel

Practical example:

Syntax:	Meaning
[MAX_RPM]	; The Result Channel Gets The Name MAX_RPM
C1 = F(#Vrear, F(med 5))	; get rid of spikes (median filter with 5 samples)
C2 = F(#RPM, F(med 5))	; get rid of spikes (median filter with 5 samples)
C4 = *(12000,#C1)	; Rear Speed * 12000 / RPM
	; = Speed at 12000 revs (Gearbox Speed)
C5 = I(#C4,#C2)	
C5 = SET(DIM=KMH)	; Set Channel Extension to 'KMH'
C6 = WORD(#C5,0,350)	; generate Word Channel with limits 0 and 350
GearSpeed = F(#C6,F(L_PHAMM3Hz33))	
C7 = F(#C6,F(AVG 3))	
Gear' = Derivate(#C7)	C8=MAXHOLD(#C2,#Gear',100,-5) ; Take filtered RPM Channel (without spikes)
RESULT=FREQ(#C8,25)	

- This example calculates a speed- and RPM-channel without spikes (C1,C2) .
- Then the quotient speed/RPM * 12000 is calculated (C4,C5) (which is the gearboxspeed).
- The dimension of this channel is set to "KMH" and a WORD channel is created (C6), which uses less memory than the default single precision result channels.
- This channel is filtered with a 3Hz Hamming low pass filter and stored as channel "GearSpeed".
- Now the unfiltered GearSpeed channel (C6) is derived and stored as "Gear".
- This channel is used now in the MaxHold Function, to trigger the found Maximum after every Gear Change. (C8). with the Parameters 100 and -5 the reset condition is defined. To reset the Maximum, the Gear' channel needs to become 100 first. Than, after dropping back to below -5, the Value of the MaxHold Function is set to the value of C2.
- To save Memory, the MaxHold Channel is stored as 25 Hz channel.

MaxHold function



2.9.2. MinHold – function



This function stores the minimum of a channel. It works like the MaxHold function but with the minimum. The minimum is kept until the value of an other channel (the trigger channel) goes above one border value after it was below an other border value.

Syntax:	Meaning
MaxHold(#CheckChannel, #TriggerChannel, Border1, Border2)	
#CheckChannel	Channel to check for the current minimum
#TriggerChannel	Channel that triggers the reset of the minimum
Border1, Border2	Constants for the trigger conditions
	When the value of the trigger channel goes above Border2 after being below Border1, the minimum is reset to the current value of the channel CheckChannel

2.9.3. FillFromBool – function

This function stores values from a source channel, when an other channel has a value unequal to zero. If the second channel has the value zero, the function keeps the old value of the source channel.

Syntax:	Meaning
FillFromBool(#SourceChannel, #BoolChannel)	Take value from SourceChannel when BoolChannel <> 0 otherwise keep old value
#SourceChannel	Channel to take values from
#BoolChannel	Boolean-channel. When zero old values are kept otherwise value from SourceChannel is taken

2.10. Stopping execution

When an error occurs in the calculation it sometimes makes no sense to continue calculating after the error, because all following calculations depend on the error free result of previous lines. For this CalcTool provides three different abort functions.

2.10.1. Abort calculation of group when an error occurred: OnErrorExitGroup

Syntax:	Meaning
OnErrorExitGroup	Stop executing the calculation of the current group if an error occurred in lines before.

Calculating the current group is aborted when an error occurred in the lines before. Channels calculated before this line will be stored. CalcTool continues with the next group.

2.10.2. Abort whole calculation when an error occurred: OnErrorExitTotal

<i>Syntax:</i>	<i>Meaning</i>
OnErrorExitTotal	Stop executing the calculation of the whole CAL-file if an error occurred in lines before.

Calculating the current group is aborted when an error occurred in the lines before.
Channels calculated before this line will be stored.
CalcTool does not continue with the next group.

2.10.3. Abort whole calculation: ExitTotal

<i>Syntax:</i>	<i>Meaning</i>
ExitTotal	Stop executing the calculation of the whole CAL-file at this line.

Calculating the current group is aborted even if there was no error in the lines before.
Channels calculated before this line will be stored.
CalcTool does not continue with the next group.

2.11. Calling external programs

Sometimes it is useful, to start an external program after the download of a measurement is completed. This can be done by using the Execute-function.

<i>Syntax:</i>	<i>Meaning</i>
Execute(Commandline [, Param1][, ...])	Start external program defined by Commandline. Parameters to pass to the program are given seperated by commas.

Example:

[RunExternal]
Execute(c:\Race\CopyEcuFile.bat, <MesDir>)

After each download the current definition file of the ECU should be copied into the measurement directory. The copying is done by a batch file "CopyEcuFile.Bat". The parameter is a placeholder for the current measurement directory. CalcTool replaces it with the path of the measurement and passes it as a parameter to the batch file. Inside the batch file the first parameter can be accessed by using "%1", the second by "%2" and so on.

The command in the batch file could look like this:

```
copy C:\ECU\CurrentSetting.dat %1
```

Further placeholders can be:

<RaceDir> = The current software directory

<EventDir> = The current event directory

<DataDir> = The current data root

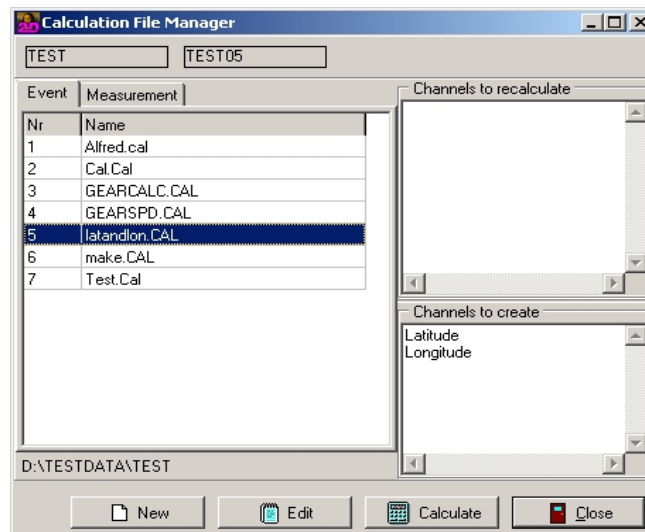
3. How to use these functions in Analyzer



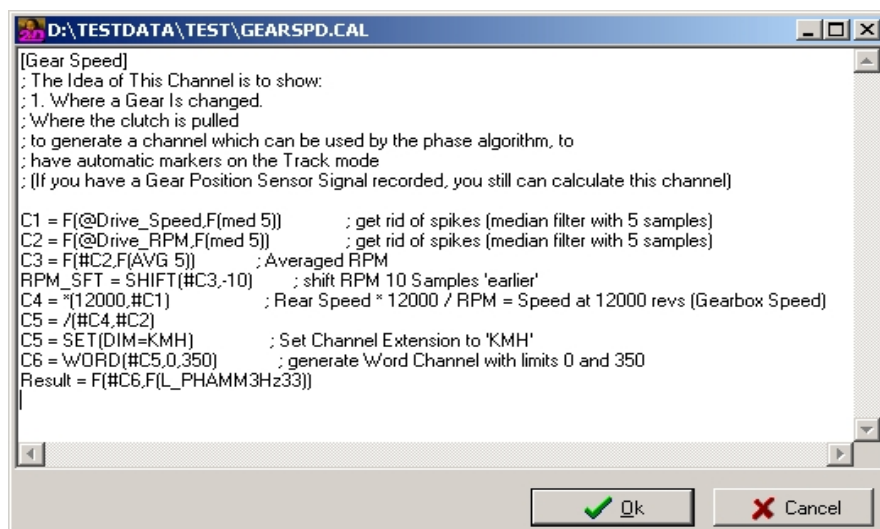
In the analysing software Analyzer are two menuitems for calculation channels available:

- "Functions" – "Calculation files/event"
Editing and calculating of CAL-files, which are located in the event, that means which are available from all measurements in that event.
- "Functions" – "Calculation files/run"
Editing and calculating of CAL-files, which are located in the measurement-directory (and only available for this run)

When one of the menuitems is selected, the Calculation File Manager appears (here with the CAL-files of the event):



In this dialog the user can create, edit and calculate one or more CAL-files.
For editing a simple texteditor is used (here with an example of a gear calculation):



3.1. Naming channels



A CAL-file must contain minimum one group. A group has to start with a name, which is written in brackets: [GroupName].

If the instructions of a group contain a line beginning with "Result=", the result of the calculation of this line is stored as a channel with the name of the group.

Each instruction starts with the name of a variable. If you do not want a channel to be stored on your computer, because it is a temporary channel for further calculation, use a so called temporary channel name. A temporary channel name starts with a "C" followed by a number.

Variables with other names are stored and can be used for analysing.

Example:

Syntax:	Meaning
[TEST]	; Name of the group
C1 = I(#Channel_1,#Channel_2)	; temporary channel C1
C2 = ...	; temporary channel C2
Temp = +(#C1,#C2)	; calculates a calculation channel Temp from temporary channels C1 and C2
A8 = Sin(#Temp)	; calculates a calculation channel A8 from the calculation channel Temp
RESULT = +(#A8,#C2)	; Result is addition of the temporary channel C2 and the calculation ; channel A8. It is stored under the name "TEST" because of the ; groupname.

In this example, three results of the calculations can be used as normal channels. This means that the software considers them as regular channels and so you have the same possibilities as with ones directly recorded. These results are Temp, A8 and Test (Result). On the other hand, the temporary channels can't be used as normal ones. The main use for them is to be steps of the calculation that we don't need to analyse.

3.2. Replacing channels

In some cases it might be useful, to replace a measured channel by a calculated one.

Perhaps the signal of the speed sensor is noisy and you want to work with a filtered speed signal in the analysis tool. To replace an existing channel with a calculated channel with the same name use the keyword "NewResult" instead of "Result"

Example:

```
[Speed]
NewResult = F(#Speed, F(Med 7))
```

This would create a channel filtered by a median filter with 7 iterations and store it under the same name as the input channel.

Note:

- The name of the group has to have the name of the channel, that should be replaced.
- It is not a true replacement, the original channel is still present but not visible for the user. That is done to provide the possibility to recalculate the channels. If the original channel is deleted, a recalculation is impossible.

3.3. The link between calculation and Specsheets



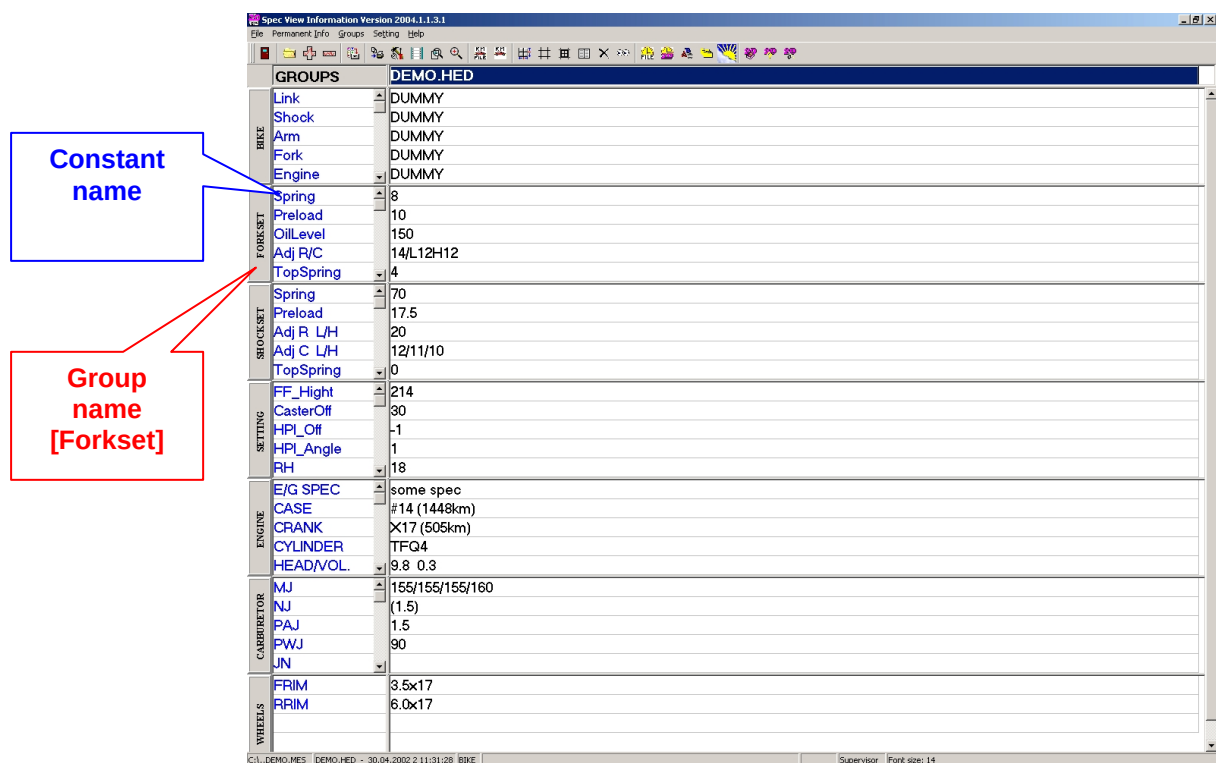
Many times you want to do some calculations with constant parameters depending of your vehicle's set-up (e.g. Springs of suspensions). Now it is possible to link the constants used in the calculation channels to the ones written in the Specsheets of the same file. The following formula can be used for any variable within the info file (Specsheet).

Formula:

constant (for calculation) = 'name of the group where constant is' . 'constant name in Specsheel'

Practical example::

Considering we've got this kind of info file:



Spec View Information Version 2004.1.1.3.1	
DEMO.HED	
GROUPS	DUMMY
Link	DUMMY
Shock	DUMMY
Arm	DUMMY
Fork	DUMMY
Engine	DUMMY
Spring	8
Preload	10
OilLevel	150
Adj R/C	14/L12H12
TopSpring	4
Spring	70
Preload	17.5
Adj R L/H	20
Adj C L/H	12/11/10
TopSpring	0
FF_Hight	214
CasterOff	30
HPL_Off	-1
HPL_Angle	1
RH	18
E/G SPEC	some spec
CASE	#14 (1448km)
CRANK	X17 (505km)
CYLINDER	TFQ4
HEAD/VOL.	9.8 0.3
MJ	155/155/155/160
NJ	(1.5)
PAJ	1.5
PWJ	90
JN	
FRIM	3.5x17
RRIM	6.0x17

We want to use in our calculation the value of the front suspension's spring stiffness for this file. The calculation is the multiplication of spring stiffness with channel #F_Susp. So due to the formula, you just have to write :

C1 = * (#F_Susp, Forkset.Spring)

Therefore the computer will use the value contained in Spring of the group FrontSusp (In this example it will be 8). Now each change in your settings will be directly linked to your calculations.



Warning

Be careful with the unit of the value in the Specsheel and the one you have to use in the calculation file. In fact sometimes they are not the same. For example your spring stiffness is in lb/mm and you want your final result in kg. So don't forget to convert in your calculation file the stiffness in kg/mm.

3.3.1. Partial reading of SpecSheet values



In some cases it might be useful not to read a complete SpecSheet but a part of it.

Syntax:	Meaning
Group.Name	Retrieves the complete value of the entry "Name" in the group "Group"
Group.Name.%N	Retrieves the Nth part of the value of the entry "Name" in the group "Group". The parts have to be separated by a space character.

Practical example:

The value of the entry "FWheel" in the group "Bike" is "X-Z 439".

"Bike.FWheel" is resolved to "X-Z 439"

"Bike.FWheel.%1" is resolved to "X-Z"

"Bike.FWheel.%2" is resolved to "439"

3.4. Automatic channel-calculation



If you want to make always the same channel-calculation after each download of data you may take advantage from the automatic channel calculation feature.

To use it simply put your calculations in a file "**AUTOCALC.CAL**" located in the current event-directory. After downloading and decompressing the data the communication tool WinIt looks in the event-directory for the file AUTOCALC.CAL and calls the calculation tool with this file.

4. Table of all calculation functions

The following table will give you an overview of available functions and the syntax to use them.

NAME	COMMAND	EXAMPLE	DESCRIPTION
Standard mathematics			
Addition	+	+(#Channel, Constant) +(#Channel1, #Channel2)	addition of channel #Channel and constant addition of channel #Channel1 and #Channel2
Const function	Const	CONST (#CONST VALUE)	Creates a channel with a constant value (=const value)
Division	/	/(#Channel, Constant) /(#Channel1, #Channel2) /(Constant, #Channel)	channel #Channel divided through constant channel #Channel1 divided through #Channel2 constant divided by a channel
Floor		FLOOR (#CHANNEL)	Calculates a channel with whole numbers
Integer division	DIV	DIV (#CHANNEL_1, #CHANNEL_2) Div (#Channel, Constant) Div (Constant, #Channel)	Channel div channel Channel div constant Constant div channel
Multiplication	*	*(#Channel1, Constant) *(#Channel1, #Channel2)	multiplication of #Channel1 with constant multiplication of #Channel1 with #Channel2
Modulo	Mod	MOD (#CHANNEL_1, #CHANNEL_2) Mod (#Channel, Constant) Mod (Constant, #Channel)	Channel modulo channel Channel modulo constant Constant modulo channel
Round		Round (#Channel)	Calculates a channel with whole numbers
Subtraction	-	-(#Channel, Constant) -(#V_Front, #VRear) -(C, #Channel)	subtracts a constant of channel #Channel1 difference of rear and front speed subtracts a channel from a constant
Trunc		TRUNC (#CHANNEL)	Calculates a channel with whole numbers

Advanced mathematics			
NAME	COMMAND	EXAMPLE	DESCRIPTION
Square root	Sqrt	sqrt(#A2)	square root of channel #A2
Logarithms	Log	log(#Channel)	logarithm's from #Channel to the base of 10
Sinus	Sin	sin(#A3)	sinus of channel #A3 (in rad)
Trigonometric			values
Sinus	DSin	dsin(#A3)	sinus of channel #A3 (in deg)
Trigonometric			values
Cosinus	Cos	cos(#A3)	cosinus of channel #A3 (#A3 in rad)
Trigonometric			
Cosinus	DCos	dcos(#A3)	cosinus of channel #A3 (#A3 in deg)
Trigonometric			
Tan	Tan	tan(#A3)	tangens of channel #A3 (#A3 in rad)
Trigonometric			
Tan	DTan	dtan(#A3)	tangens of channel #A3 (#A3 in deg)
Trigonometric			
Arctan, arcsin, arccos	Arctan, arcsin ...	arctan(#A4)	Arcus tangens of channel #A4; result of all functions in rad
Trigonometric			
Derivation	Derivate	Derivate(#D1)	Derivation of front speed = acceleration
Integration	Integrate	I(#A3) I(#A3,M)	Integrated channel #A3 Integrated Middle value M
SUM-function	Sum	Sum(#Event)	Calculates the sum of all values of channel #Event
Minimum	Min	min(#D1,#D3)	Result is the min from the 2 channels
Maximum	Max	max(#D1,#D3)	Result is the max from the channels
Absolute value	Abs	abs(#A5)	Absolute values #A5
Signum function	Sig	sig(#gyro)	If (gyro < 0) then sig(#gyro) = -1 If (gyro = 0) then sig(#gyro) = 0 If (gyro > 0) then sig(#gyro) = 1
If-function	If	If (#Ax,P1,P2,P3, P4)	If channel #Ax is P1 than P2 then result is P3 else result is P4 P1 is >, <, >=, <= or = P2 is #Ay or constant P3 is #Aw or constant P4 is #Az or constant

Bike / car physical formulas			
NAME	COMMAND	EXAMPLE	DESCRIPTION
Speed_L4	Speed	Speed_L4(#V_Front)	Improves speed-signal of channel #V_Front
Speed2	Speed2	Speed2(#V_Front, #V_Rear)	Calculates the average of two speed-signals.
Slip	Slip	slip(#V_Front, #V_Rear)	Slip function front/rearspeed in %
Banking	Banking	Banking(#Gyro,0.0)	Calculates the inclination of the bike with the gyro's signal (channel #Gyro) considering that the inclination of the bike in the start/finish line is 0.0 .
BANK2ACC	BANK2ACC	BANK2ACC(#Channel17)	Calculates the lateral acceleration of a bike with the calculated banking angle (channel #Channel17).

Additional formulas of existing channels as new channel		
NAME	COMMAND	DESCRIPTION
Formula Integer	F(#Channel, I(M, D, O)) M multiplicand D divisor O offset	integer formula of channel #Channel e.g. to use same channel data as #Channel1, but use another formula. For example to see digits use F(#Channel1, I(1, 1, 0))
Formula Real	F(#Channel, R(F, O)) F factor (multiplication/divisor) O offset	real formula of channel #Channel e.g. to use same channel with different Offset
Table	F(#Channel, T(name)) name = table filename	another table of channel #Channel e.g. to use same channel with another table
Table 2	F(#Channel, 2(Name)) name = table filename	#Channel will be connected to the table "Name"
3D-Table	F(#Channel1, #Channel2, T(TableName.Ext, IntType)) TableName.Ext IntType	#Channel1 and #Channel2 will be connected to the table "TableName.Ext" name of the table file with extension Type of interpolation (i.e. LIN)
Filter	F(#Channel, F(L_P Hann 1.0Hz 99)) L_P Hann 1.0Hz 99	Filter #Channel with low pass hanning window 1 Hz 99 Iterations.
Frequency	AVG 21 MED 33 Freq(#D1, 25)	Average filter 21 values Median 33 Iterations Channel D1 but with 25 Hz sampled
FreqBase	FreqBase(#Channel, NewRate, GPSRate) #Channel = Name of the channel NewRate = New samplingrate GPSRate = Rate at which the GPS sends new data	Changes samplingrate of channel #Channel to NewRate
Mux function	MUX(#DataChannel, #MuxChannel, MuxVal) #DataChannel = Name of the channel containing the multiplexed data #MuxChannel = Name of the channel containing the multiplex values MuxVal = Value at which the data should be taken	Extracts the values of DataChannel when MuxChannel has the value MuxVal

Channel manipulation		
NAME	COMMAND	DESCRIPTION
Shifting	Shift(#Channel, Scans)	Shifting a channel to the right for Scans. Use a negative value to shift to the left
Combining two 16 bit channels	MAKE32(#ChannelHi, #ChannelLo)	Combines two 16 bit channels to one 32 bit channel
Combining two 32 bit channels	MAKE64(#ChannelHi, #ChannelLo)	Combines two 32 bit channels to one 64 bit channel
Changing the storage type	#ChannelHi: Channel with the upper 16 bit #ChannelLo: Channel with the lower 16 bit Word(#Channel) Word(#Channel, LowerBorder, UpperBorder) #Channel = input channel containing the high 32 bit = input channel containing the low 32 bit = channel to change the store type LowerBorder = All values of the original channel which are lower than this border are set to this border. UpperBorder = All values of the original channel which are higher than this border are set to this value	Changes the store type of a channel to word
Swap function	Binary(#Channel, Range) Binary(#Channel, Range, LowerBorder, UpperBorder) Swap(#Channel) #Channel = channel to change the byte order	The Binary-function is very similar to the Word-function. The difference is the parameter Range Changing the byte-order of a channel
Changing dimension	Set(DIM='Km/h')	Changes the dimension of a channel

Special functions for phase-calculation		
NAME	COMMAND	DESCRIPTION
MaxHold	MaxHold(#CheckChannel, #TriggerChannel, Border1, Border2)	Stores the maximum of a channel. The maximum is kept until the value of an other channel (the trigger channel) goes below one border value after it was above an other border value.
MinHold	MinHold(#CheckChannel, #TriggerChannel, Border1, Border2)	Stores the minimum of a channel. The minimum is kept until the value of an other channel (the trigger channel) goes above one border value after it was below an other border value.



If you need additional functions which are not included in RACE yet, please contact 2D. It is possible to extend the function library on request.

Appendix A: – Example of a CAL file



You will get this CAL–file if you install our full software package WinARace. In this case you will get three different demo measurements. You can find those demo measurements at the fixed directory “C:\RaceData\DemoData”.

Presuppose C: is the system drive of you operating system (windows).

The CAL – file at the table has the name “CAL.CAL” and exists at the “Bike” directory. You can load and edit a Cal-file with a simple text file editor – for example NotePad.exe.



Take notice of: For a better understanding of a calculation statement you should write comments in the Cal – files. To do so you must write a semicolon as a fixed prefix character for a comment.

Example:

Calculate hi and low speed suspension signal

Syntax	Explanation / Link
[Suspension]	
FSus_lo=F(@Front_susp,F(L_PHAMM3HZ77))	; FIR Low Pass Hamming Filter with 3 Hz and 77 iterations – refer Filter function
RSus_lo=F(@Rear_susp,F(L_PHAMM3HZ77))	
FSus_hi=F(@Front_susp,F(L_PHAMM25HZ77))	; 25 Hz Finit Impulse Response Low Pass Filter with Hamming Window - refer Filter function
RSus_hi=F(@Rear_susp,F(L_PHAMM25HZ77))	
; generate suspension movement speed	
FSus' = Derivate(#FSus_hi)	; Derivation of the FSus_hi Channel
RSus' = Derivate(#RSus_hi)	refer Derivation
; generate 25 Hz channel to get finer resolution for FFT!!!	
FSus_25 = FREQ(#FSus_lo,25)	; Reduce sampling rate of a channel, here result frequency will be 25 Hz
RSus_25 = FREQ(#RSus_lo,25)	refer Frequency
[Bike_acc]	
C1=F(@Int_Speed,F(med 5))	; result Channel will be Bike_acc
	; get rid of spikes (median filter with 5 samples)
	refer Filter function
C2=F(#C1,F(L_PHAMM3Hz55))	; get down to mass acceleration
	refer Filter function
Speed_lo=WORD(#C2,0,500)	; save memory, convert float channel to Word Channel, ; with limits 0 and 500
	refer changing the store type
C3=Derivate(#C2)	; Derivate Speed Signal
	refer Derivation
C4=WORD(#C3,-30,30)	
C5=F(#C4,F(AVG 5))	; make output looking more smooth (average with 5 samples)
	refer Filter function
;C5=F(#C4,F(L_PHAMM2Hz55))	; make output looking more smooth (average with 5 samples)
	refer Filter function

[GearSpeed]

; The Idea of This Channel is to show:
; 1. Where a Gear Is changed.
; Where the clutch is pulled
; to generate a channel which can be used by
the phase algorithm, to have automatic
markers on the Track mode
; (If you have a Gear Position Sensor Signal
recorded, you still can calculate this channel)

C1 = F(@Drive_Speed,F(med 5))

; get rid of spikes (median filter with 5 samples)
[refer Filter function](#)

C2 = F(@Drive_rpm,F(med 5))

; get rid of spikes (median filter with 5 samples)
[refer Filter function](#)

C3 = F(#C2,F(AVG 5))

; Averaged RPM
[refer Filter function](#)

RPM_SFT = SHIFT(#C3,-10)

; shift RPM 10 Samples 'earlier'
[refer Shifting a channel](#)

C4 = *(12000,#C1)

; shift RPM 10 Samples 'earlier'
[refer Shifting a channel](#)

C4 = *(12000,#C1)

; Rear Speed * 12000 / RPM = Speed at 12000
revs (Gearbox Speed)
[refer Multiplication](#)

C5 = /(#C4,#C2)

C5 = SET(DIM=KMH)

; Set Channel Extension to 'KMH'
[refer change dimension](#)

C6 = WORD(#C5,0,350)

; generate Word Channel with limits 0 and 350
[refer changing the store type](#)

GearSpeed = F(#C6,F(L_PHAMM5Hz33))

[refer Filter function](#)

;C7 = F(#C6,F(AVG 3))

[refer Filter function](#)

;Gear' = Derivate(#C7)

C22 = if (#GearSpeed,>,GearBox.1,1,0)

[refer If-function](#)

C23 = if (#GearSpeed,>,GearBox.2,1,0)

C24 = if (#GearSpeed,>,GearBox.3,1,0)

C25 = if (#GearSpeed,>,GearBox.4,1,0)

C26 = if (#GearSpeed,>,GearBox.5,1,0)

C27 = if (#GearSpeed,>,GearBox.6,1,0)

C28 = +(#C22,#C23)

[refer Addition](#)

C29 = +(#C28,#C24)

C20 = +(#C29,#C25)

C21 = +(#C20,#C26)

Gear_Nr = +(#C21,#C27)

[refer Addition](#)

Gear_Nr= SET(DIM=.)

; Set Channel Extension to .
[refer change dimension](#)

C30 = F(#Gear_Nr,F(AVG 5))

Gear' = Derivate(#C30)

;MAX_G_rpm=MAXHOLD(#C2,#Gear',100,-100)

; Check for max RPM on spike free Revsignal,
Reset Value when Gear' is crossing -5 coming
from 100

[refer MaxHold](#)

C12=if(@Throttle,>,5,1,0)

[refer If-function](#)

ACC_RPM=*(#C2,#C12)

; set RPM 0 if Throttle is closed - refer
[Multiplication](#)

C14=MAXHOLD(#ACC_RPM,#Gear',3,1)

[refer MaxHold](#)

C15=WORD(#C14)

[refer changing the store type](#)

MAX_G_rpm=FREQ(#C15,25)

[refer Frequency](#)

MAX_G_rpm= SET(DIM=RPM)

; Set Channel Extension to RPM' [refer Frequency](#)

Appendix B: The link between CalcTool and Specsheets



In the following example you will see the SpecsHEET group [CalcTool]. Specsheets are special documentation files that belong to every measurement done with the 2D equipment. These files are generated for every download (=reading the data from the datalogger).

A SpecsHEET file has the same name as the measurement but with the file extension “*.HED”. The corresponding SpecsHEET for each measurement must exist directly in the “MES- directory”.

In the files you can gather all the information that is relevant for the circumstances and special set-up for your measurements and events (e.g. the settings of a bike during a race weekend or testing). The user of the program – the engineer, the data recording man, or the mechanic, is in charge of the maintenance and integrity of the data. And you are able to eliminate a good part of additional paperwork for documentation.

[Groups] organise the details of the information.

If you start CalcTool with any CAL-file (refer chapter [“How can I start the program CalcTool manually ?”](#)) the program will write the name of this CAL-file into the group [CalcTool] of the related SpecsHEET. Additionally you will get date and time information of the CalcTool starting.

Take a look at this Example:

A SpecsHEET may have this look.

[CalcTool]

D:\TESTDATA\ZuLoeschchen\Test.CAL=07.06.2004 16:05:21
D:\TESTDATA\ZuLoeschchen\XDel.CAL=07.06.2004 16:05:00
D:\TESTDATA\ZuLoeschchen\MedTest.CAL=08.06.2004 10:07:13
D:\TESTDATA\ZuLoeschchen\TEST-01.MES\LoggerNameInclude.CAL=08.06.2004 10:10:25
D:\TESTDATA\ZuLoeschchen\TEST-01.MES\Include.CAL=08.06.2004 10:21:19

[Tyres]

FrontTyre_K1=GPZ25806_K1
FrontTyre_K2=GPZ25806_K2
FrontTyre_K3=GPZ25806_K3

[DOWNLOAD]

FrontDis (m)=0
RevCount (Refs)=0
Run total (s)=8075
Run recording (s)=8070
Start of download=02.06.2004 09:27:48
End of download=02.06.2004 09:32:05
Number of errors=0
Time to decompress=0.0 sec
System id=11H021BB825
Programversion=2.2.2.1
Userlevel=Supervisor

[Weather]

Condition=dry
DT/WT=25/20/61.2%
AirP/K=1000/1.027
track=anything
time=now

NAME	COMMAND / FUNCTION	Function since when available?
Combining two 16 bit channels chapter 2.8.2 page 30	MAKEGPS(#ChannelHi, #ChannelLo) #ChannelHi: input channel containing the high 16 bit #ChannelLo: input channel containing the low 16 bit	06.06.2005
chapter 2.6.6 page 24	F(#Ch1, #Ch2, #Ch3, T(TableName.Ext, IntType)) TableName.Ext ; name of the table file containing an with extension IntType ; Type of interpolation (i.e. LIN)	28.10.2005
Condition IfLaptimesExist chapter 1.1.2.3	If LaptimesExist Result = ...	28.11.2005
Condition IfNotLaptimesExist chapter 1.1.2.4	If NotLaptimesExist Result = ...	28.11.2005
Condition IfSpecValueExists chapter 1.1.2.5	IfSpecValueExists	28.11.2006
Condition IfNotSpecValueExists chapter 1.1.2.6	IfNotSpecValueExists	28.11.2006
Entering values at runtime chapter 1.1.3	Entering values at runtime	28.11.2006
Filter with variable depth chapter 2.6.7.3	Filter with variable depth	28.11.2006
UseBits-function chapter 2.6.11	Use Bits-function	11/28/2006
Stopping execution chapter 2.10	Stopping execution	28.11.2006
Word-function chapter 2.8.4.1	Word(#Channel, Step)	22.01.2007
Calculated mileage chapter 1.1.5	Mileage(#SpeedChannel, LoggerName)	11.04.2007
2D Table search function chapter 2.6.7	T2DSearch(#YChannel, #ValueCh, T(TableName.E[, SearchDirection])	11.04.2007
Partial reading of SpecSheet values chapter 3.3.1	Group.Name Group.Name.%N	30.05.2007

Appendix C: What's new ?

NAME	COMMAND / FUNCTION	Function since when available?
MinValue	MinValue(#Channel)	30.05.2007
MaxValue	MaxValue(#Channel)	30.05.2007
FirstValue	FirstValue(#Channel)	30.05.2007
LastValue	LastValue(#Channel)	30.05.2007
Signed	Signed(#Channel)	13.06.2007
Round	ROUND (#CHANNEL)	13.12.2007
Floor	FLOOR (#CHANNEL)	13.12.2007
Trunc	TRUNC (#CHANNEL)	13.12.2007
FillfromBool	FillFromBool(#SourceChannel, #BoolChannel)	19.03.2008
TAU filter	Refer chapter Full filter interface	19.03.2008
MinValue MaxValue FirstValue LastValue	Changed (additional) syntax declaration of the four functions: • MinValue(#Channel,Rate) • MaxValue(#Channel,Rate) • FirstValue(#Channel,Rate) • LastValue(#Channel,Rate)	11.06.2008
AvgValue	The Average value function AvgValue(#Channel, Rate)	11.06.2008
EnterValue	EnterValue('Text shown in dialog') EnterValue('Text', Value)	30.07.2008
EnterChannel	EnterChannel('Text shown in dialog') EnterChannel('Text', #Channel)	30.07.2008
Calling external programs	Execute(Commandline [, Param1][, ...])	06.08.2008
GPSShift	GPSShift(#GPSCHANNEL,#SHIFTCHANNEL, NEWFREQ)	14.10.2008