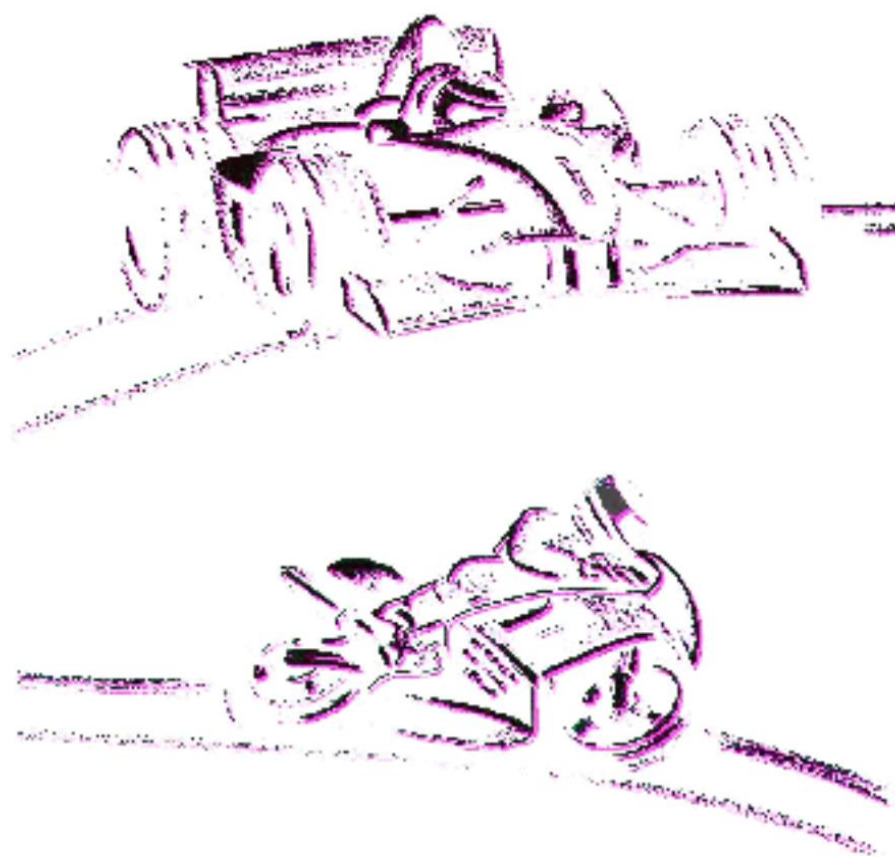


- English -



Online Calculation Channels (CALC channels)

1 Revision History

Revision	Description	Release Date	Author
0	Initial Release	2018-04-18	
1	Revision 1	2021-07-09	FS

Revision 1

Fundamental revision and addition of the latest commands and examples.

2 Content

1	REVISION HISTORY	2
2	CONTENT	3
3	INTRODUCTION	4
4	CHANNEL SETTINGS	6
4.1	GENERAL	6
4.2	ANALYSIS	6
4.2.1	Calibration	6
4.2.2	Calculation formular	7
4.3	PARAMETERS	8
4.4	DATATYPE	8
5	IMPORTANT INFORMATION	9
5.1	SYNTAX	9
5.2	ERRORS	9
5.3	PROCESSING ORDER	9
5.4	COMPATIBILITY	9
5.5	REGROUPING CAN-CHANNELS FOR SENDING VIA CAN/RF	10
5.6	CALCULATION CHANNEL DEFINITIONS	11
5.6.1	CDF-file opened in text editor	11
5.6.2	Export	12
5.6.3	Import	12
6	EXPRESSIONS	13
7	APPENDIX	21
7.1	EXAMPLES	21
7.1.1	Brake Balance Calculation	21
7.1.2	Laps per run and target laps	22
7.1.3	Triggering an alarm channel when Oil pressure is low	22
7.1.4	Calculation of Fuel Consumption 1	23
7.1.5	Calculation of Fuel Consumption 2	23
7.1.6	Calculate the driven mileage (Reset in Winit via button F3)	24
7.1.7	Calculate the driven mileage (Permanent accumulation)	24
7.1.8	Count the number of detonations in a run	24
7.1.9	Reset variable m1 via Button	24
7.1.10	Counting the number of samples	25
7.1.11	TimeCounter (PitTimer)	25
7.1.12	AnalogTest toolchain	28

3 Introduction

A calculation or "Calc" channel can perform mathematical operations to existing channels, according to the user-defined formula entered to the channel and can be found in nearly every 2D CAN module.



Important information

These calculations are carried out synchronously with the recorded channels measurement data and thus online in the module itself, so that the calculation results are available in real time.

CALC channels can have a lot of different purposes. They can be used to put different channels together in one CAN send identifier, execute commands to get information from the module or to create new channels.

Their information will be available **online** in your system and can be sent to different devices too.



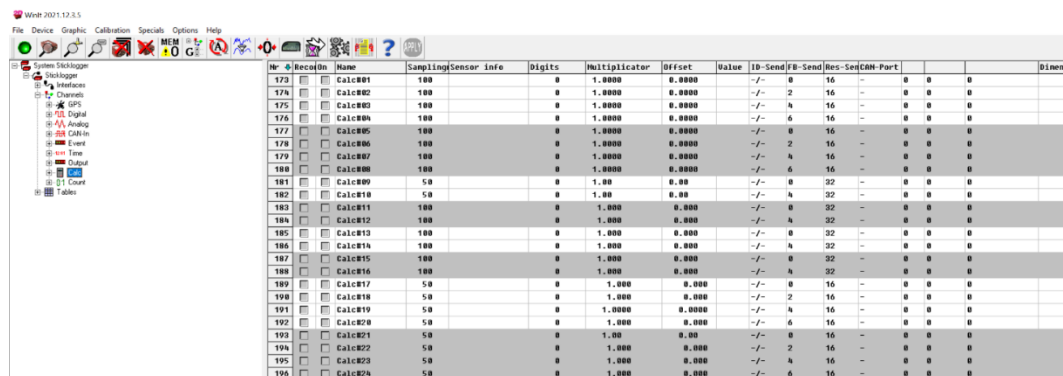
Important information

2D Loggers are also able to record its own CALC channels for debugging reasons.

This manual will help you find the commands to program the CALC channels and will also give a few examples on how CALC channels can be used.

Please be aware that there may be differences depending on the internal CALC library used in the different modules. Therefore, some commands may not work.

The resulting value of the CALC-channel can be copied to the display output, used as a trigger channel for alarm or event channels, sent to the datalogger for recording, or used in other CALC channels.



Nr	RecOn	Name	Sampling	Sensor Info	Digits	Multiplicator	Offset	Value	ID-Send	ID-Recv	Res-Send	CAN-Port	Dimens
173		Calc#01	100		0	1.0000	0.0000	-/-	0	16	-	0 0 0	
174		Calc#02	100		0	1.0000	0.0000	-/-	2	16	-	0 0 0	
175		Calc#03	100		0	1.0000	0.0000	-/-	4	16	-	0 0 0	
176		Calc#04	100		0	1.0000	0.0000	-/-	6	16	-	0 0 0	
177		Calc#05	100		0	1.0000	0.0000	-/-	8	16	-	0 0 0	
178		Calc#06	100		0	1.0000	0.0000	-/-	2	16	-	0 0 0	
179		Calc#07	100		0	1.0000	0.0000	-/-	4	16	-	0 0 0	
180		Calc#08	100		0	1.0000	0.0000	-/-	6	16	-	0 0 0	
181		Calc#09	50		0	1.00	0.00	-/-	0	32	-	0 0 0	
182		Calc#10	50		0	1.00	0.00	-/-	4	32	-	0 0 0	
183		Calc#11	100		0	1.000	0.000	-/-	0	32	-	0 0 0	
184		Calc#12	100		0	1.000	0.000	-/-	4	32	-	0 0 0	
185		Calc#13	100		0	1.000	0.000	-/-	0	32	-	0 0 0	
186		Calc#14	100		0	1.000	0.000	-/-	4	32	-	0 0 0	
187		Calc#15	100		0	1.000	0.000	-/-	0	32	-	0 0 0	
188		Calc#16	100		0	1.000	0.000	-/-	4	32	-	0 0 0	
189		Calc#17	50		0	1.000	0.000	-/-	0	16	-	0 0 0	
190		Calc#18	50		0	1.000	0.000	-/-	2	16	-	0 0 0	
191		Calc#19	50		0	1.00000	0.00000	-/-	4	16	-	0 0 0	
192		Calc#20	50		0	1.0000	0.0000	-/-	6	16	-	0 0 0	
193		Calc#21	50		0	1.00	0.00	-/-	0	16	-	0 0 0	
194		Calc#22	50		0	1.000	0.000	-/-	2	16	-	0 0 0	
195		Calc#23	50		0	1.000	0.000	-/-	4	16	-	0 0 0	
196		Calc#24	50		0	1.000	0.000	-/-	6	16	-	0 0 0	

The following points can vary between different 2D modules:

- Various 2D modules contain CALC-channels
Not every 2D module is containing CALC-channels.
- Amount
The number of available CALC channels may differ between the different 2D modules.

**Important information**

The number of available CALC channels ranges from 8 to 64 CALC channels!

- Type
Calculation channels are available as 16- and 32-bit channels.
However, not every 2D module also has 32-bit CALC channels.
- Number of calculation-characters per calculation function
The number of characters that can be used per calculation channel varies between different 2D modules.
*Example: 1*1*1 are 5 characters*
- Number of possible calculations per second
Depending on the 2D module, the number of possible calculations per second also differs.
*Example: 1*1*1 are two calculations*

**Important information**

Some 2D modules are able to execute up to 1 million calculations per second!

If any of the above points limit your application, please contact 2D via the [contact form](#) and we will help you find the most suitable module!

4 Channel settings

4.1 General

In this tab the CALC-channel name, dimension and short-cut can be chosen.

Also, the channel can be turned on and activated for recording.



Further Information

- It is also possible to turn on a CALC channel only for a calculation but not record the channel.

4.2 Analysis

4.2.1 Calibration

Setting the offset and multiplier value of the respective CALC channel.



Further Information

Multiplier and Offset can be set for each CALC channel individually!


The multiplier value determines the resolution of the respective channel.

Multiplier and Offset are especially important when using the respective CALC channels for displaying on Dashboards!

4.2.2 Calculation formular

In field *Calculation formular* the respective calculation of the CALC channel can be chosen.

The possible formulars can be found in chapter **Expressions**.




Further Information

Multiple calculation expressions can be freely combined.



Important information

In box calculation formula only a limited number of characters can be inserted. Space characters included and the maximum character numbers are varying between the different modules!

Other channels can be called either via using the channel name (#Bestlap) or the channel number (#33).




Further Information

When calling up via the channel name, upper and lower case does not matter.

Beside using other channels for calculations, also variables can be used for calculations.

Thereby, it is distinguished between three different types of variables:

- x-variables:
Will be set to 0 after the Dash is unpowered. Variable **x** represents the channel value of the previous sample. Often used at if-conditions.
- m-variables:
Keeping its last value after power off and can be erased by pressing Empty(F3) in *Winit*.



Important information

m- variables are only available in limited numbers (m: 1-6) and only in BigDash!

- p-variables:
Keeping its last value after power off, are only able to increase its value and cannot be erased! p1 variable should only be used for distance or lifetime counters.



Important information

p-variables are only available in limited numbers (p: 1-2) and in MidiDash, BigDash and TFTDash! With newer Firmware (Summer 2021) in MidiDash variable p2 can be used for storing values during power off!

4.3 Parameters

In this tab the Samplingrate [Hz] of the CALC channel can be set.

4.4 Datatype

In this tab it can be chosen if the channel is signed or unsigned.

5 Important information

5.1 Syntax



Further Information

All possible operators and functions are described in the table following.

expr = numerical constant or variable or channel
expr = expr operator expr expr := function "(" expr ")"
expr = expr operator expr expr := function "(" expr ")"
numerical constant = positive or negative decimal number
numerical constant = hex number // "0x" or "0h" followed by hexadecimal digits 0..F
numerical constant = binary number // "0b" followed by binary digits (0 and 1)
channel = "#" or "#" variable = m1 or m2 or ..., or m6 or p1 or p2 or x

5.2 Errors

If the calculation channel has the value "-1" or "65535" (depending on if the digits are signed or not) even though you expect other values, please open the respective CALC channel and see respective error message.



Further Information

- In box Calculation formula only a limited number of characters can be inserted.
Space characters included!

5.3 Processing order

CALC channels can also be used in other calculations for, e.g., creating new channels.

Because the CALC channels of 2D modules are processed cyclically, the order of the CALC channels is also important when using CALC channels in other calculations.

<input type="checkbox"/>	<input checked="" type="checkbox"/>	BestLap	200	bestlap()
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Laptime_est	200	#Bestlap + #GAP

Furthermore, it is recommended to use same sampling rate at CALC channels which are used together in a calculation.

5.4 Compatibility

CALC channels can be used for compatibility purposes e.g., when channels should be sent to ECU's with different samplingrate or channel format.

5.5 Regrouping CAN-channels for sending via CAN/Rf

The CALC channels can be used to group channels in order to use as few identifiers as possible when transmitting via CAN/Rf or a different sampling rate should be used for sending the channels.

The purpose of both applications is to save transmission capacity.

Thereby new sending identifiers with e.g., an analogue, digital, CAN and time channels can be easily created by only calling the respective channels with #... in CALC channel and also sending rate (samplingrate) can be freely chosen.

Example: Sending GPS, analogue, Time and Count channels via CAN-ID 600 with 10 Hz:

On	Samplingrate	Name	Sensor info	Digits	Multiplicator	Value	Offset	ID-Send	FB-Send	Res-Ser
<input checked="" type="checkbox"/>	10	Send1	#ValidSat	0	1.000	0.0	0.000	600/-	0	16
<input checked="" type="checkbox"/>	10	Send2	#U_KL_30	0	1.000	0.0	0.000	600/-	2	16
<input checked="" type="checkbox"/>	10	Send3	#SysTime	0	1.000	0.0	0.000	600/-	4	16
<input checked="" type="checkbox"/>	10	Send4	#Laps	0	1.000	0.0	0.000	600/-	6	16



Further Information

Beside only calling the respective channels with #... also all other calculations can be applied here!

5.6 Calculation channel definitions

Since 2018 it is possible to export the calculation definitions of the CALC channels and import exported ones as well. This might be a very helpful function, if working with different data systems.

The exported Calculation channel definition is saved as .CDF-file which can be opened and edited with text editor.

CDF-files can also be imported to CALC channels group again.



Important information

When importing Calculation channel definitions, the import always starts at **first** CALC channel for importing and overwrites all channels which are used in CDF-file!



Further Information

When only lines should be inserted to CALC channels via CDF file, the existing CALC lines should be exported first, and new lines inserted with text editor. Subsequently, the new CDF-file is imported.

5.6.1 CDF-file opened in text editor

Example PitTimer (chapter 7.1.11):

Samplingrate	Name	Sensor info	Digits	Multiplicator	Value	Offset
25	TrgCH	#x2_CF	0	1.000		0.000
25	TrgCH2	#x2_CF	0	1.000		0.000
25	TrgInVal	3	0	1.000		0.000
25	TrgOutVal	18	0	1.000		0.000
25	BoxWait	20	0	1.000		0.000
25	hh	div(#Hhmm,1)	0	0.001		0.000
25	SOD	#hh*3600+(#HHMM-#HH)*6000+#second	0	1.000		0.000
25	m1	m1=if(#TrgCh==#TrgInVal, #SOD, m1)	0	1.000		0.000
25	m1	m1=if(#TrgCh2==#TrgOutVal, 0, m1)	0	1.000		0.000
25	TimeSince	if(m1==0, 0, #SOD-m1)	0	1.000		0.000
25	Timer	if(#TimeSince<_BoxWait,#BoxWait-#TimeSince,0)	0	1.000		0.000

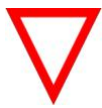
```

_TrghCH=_x2_CF;//1,0,25,5,1,
_TrghCH2=_x2_CF;//1,0,25,5,1,
_TrghInVal=3;//1,0,25,5,1,
_TrghOutVal=18;//1,0,25,5,1,
_BoxWait=20;//1,0,25,5,1,
_hh=div(_Hhmm,1);//0.001,0,25,5,1,
_SOD=_hh*3600+(#HHMM-#HH)*6000+_second;//1,0,25,8,1,
_m1=m1=if(_TrghCh==_TrghInVal, _SOD, m1);//1,0,25,5,1,
_m1=m1=if(_TrghCh2==_TrghOutVal, 0, m1);//1,0,25,5,1,
_TimeSince=if(m1==0, 0, _SOD-m1);//1,0,25,5,1,
_Timer=if(_TimeSince<_BoxWait,_BoxWait-_TimeSince,0);//1,0,25,5,1,
_Calc#12=;//1,0,25,5,1,
_Calc#13=;//1,0,25,5,1,
_Calc#14=;//1,0,25,5,1,
_Calc#15=;//1,0,25,5,1,
_Calc#16=;//1,0,25,5,1,

```

First line:

Name + Sensor info	Multiplicator	Offset	Samplingrate	Length	Digits after dot
_TrghCH=_x2_CF	1	0	25	5	1



Important information

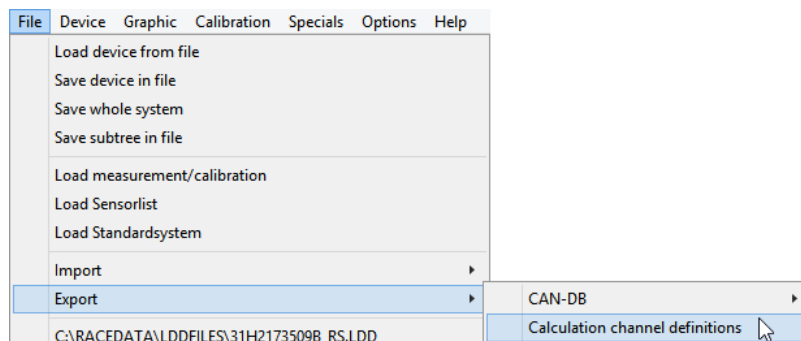
This CDF file does not contain the following information:

Turned: On/Off
Recording: On/Off
Resolution: 16Bit/32Bit
Datatype: Signed/Unsigned

5.6.2 Export

To export your calculation definitions of the calc channels, select the module you want to export the calculation definitions from.

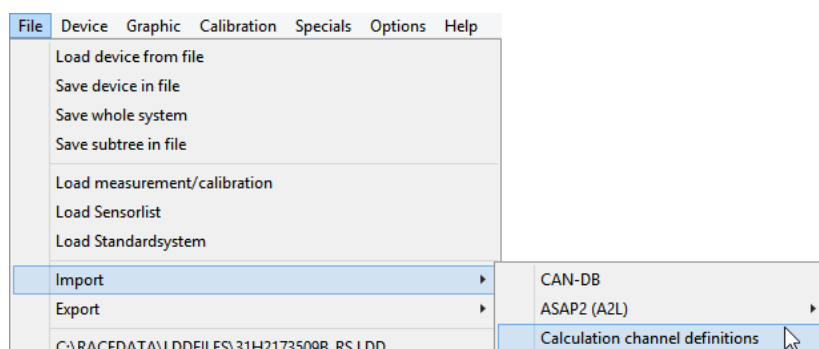
Then select the menu item “File” ⇒ “Export” ⇒ “Calculation channel definitions”:



This will export all calc channel definitions of this module. You only have to select where to store the file and maybe enter an individual file name.

5.6.3 Import

To import calc channel definitions, you select the module, you want to add the calc channels to and select the menu item “File” ⇒ “Import” ⇒ “Calculation channel definitions”:



In the following window you simply have to select a calc definition file. This will import all calc channel definitions of the selected file. Confirm the changes with <Apply>.

6 Expressions

Arithmetic expressions			
Sum	$expr1 + expr2$	Result is the sum of $expr1$ and $expr2$	$3 + 4$ $\#4 + 10.7$ $\#speed + \#4$
Difference	$expr1 - expr2$	Result is the difference between $expr1$ and $expr2$	$3 - 4$ $\#4 - 10.7$ $\#speed - \#4$
Product	$expr1 * expr2$	Result is the product of $expr1$ and $expr2$	$3 * 4$ $\#4 * 10.7$ $\#speed * \#4$
Division	$expr1 / expr2$	Result is $expr1$ divided by $expr2$	$3 / 4$ $\#4 / 10.7$ $\#speed / \#4$
Integer division	$div(expr1, expr2)$	Result is $expr1/expr2$ with remainder discarded	$div(3, 4)$ $div(\#4, 10.7)$ $div(\#speed, \#4)$
Modulo	$expr1 \% expr2$	Result is the remainder of $expr1/expr2$	$3 \% 4$ $\#4 \% 10.7$ $\#speed \% \#4$
Power	$expr1 ^ expr2$	Result is $expr1$ to the power of $expr2$	$2 ^ 8$ $\#4 ^ 2$ $2 ^ \#4$
Square root	$Sqrt(expr)$	Result is the square root of $expr$ ($expr$ must be positive).	$sqrt(2)$ $sqrt(\#4)$
Logarithm	$Log10(expr)$	Result is the base-10 logarithm of $expr$ ($expr$ must be positive).	$log10(2)$ $log10(\#4)$ $log10(\#speed)$
Natural logarithm	$ln(expr)$	Result is the natural logarithm (to base e) of $expr$ ($expr$ must be positive).	$ln(2)$ $ln(\#4)$ $ln(\#speed)$
Sign	$sig(expr)$	Result is the sign of $expr$: $1 \rightarrow$ if $expr$ is positive; $-1 \rightarrow$ if $expr$ is negative; $0 \rightarrow$ if $expr$ is zero.	$sig(-2)$ $sig(\#4)$ $sig(\#speed)$

Absolute value	<code>abs(expr)</code>	Result is the absolute value of <i>expr</i> .	<code>abs(-2)</code> <code>abs(#4)</code> <code>abs(#speed)</code>
Derivation	<code>der(expr)</code> <code>deriv(expr)</code>	Result is the rate of change between the values of <i>expr</i> computed in the previous cycle and the current value. (time derivative)	<code>der(#4)</code> <code>der(#dist)</code> <code>der(#4-#speed)</code>
Sum over time	<code>sum(expr)</code>	Result is the sum of all the values of <i>expr</i> .	<code>sum(1)</code> <code>sum(#4)</code> <code>sum(#dist)</code>
Integration	<code>i(expr)</code> <code>integ(expr)</code>	Result is the integrated values of <i>expr</i> over time.	<code>i(#speed)</code> <code>i(#acc)</code>

Relational expressions			
Smaller	<code>expr1 < expr2</code>	Result is 1 if <i>expr1</i> is smaller than <i>expr2</i> . 0, otherwise	<code>#4 < #10</code> <code>#4 < 10.7</code> <code>10 < #speed</code>
Smaller or equal	<code>expr1 <= expr2</code>	Result is 1 if <i>expr1</i> is smaller than or equal to <i>expr2</i> . 0, otherwise.	<code>#4 <= #10</code> <code>#4 <= 10.7</code> <code>10 <= #speed</code>
Greater	<code>expr1 > expr2</code>	Result is 1 if <i>expr1</i> is greater than the value of <i>expr2</i> . 0, otherwise.	<code>#4 > #10</code> <code>#4 > 10.7</code> <code>10 > #speed</code>
Greater or equal	<code>expr1 >= expr2</code>	Result is 1 if <i>expr1</i> is greater than or equal to <i>expr2</i> . 0, otherwise.	<code>#4 >= #10</code> <code>#4 >= 10.7</code> <code>10 >= #speed</code>
Equality	<code>expr1 == expr2</code>	Result is 1 if <i>expr1</i> is equal to <i>expr2</i> . 0, otherwise.	<code>#4 == #10</code> <code>#4 == 10.7</code> <code>10 == #speed</code>
Inequality	<code>expr1 != expr2</code>	Result is 1 if <i>expr1</i> is not equal to <i>expr2</i> . 0, otherwise.	<code>#4 != #10</code> <code>#4 != 10.7</code> <code>10 != #speed</code>

Logical expressions			
Logical AND	<i>expr1</i> && <i>expr2</i>	Result is 1 if both <i>expr1</i> and <i>expr2</i> are not equal to 0. 0, otherwise.	(0<#4)&&(#4
Logical OR	<i>expr1</i> <i>expr2</i>	Result is 1 if at least one of <i>expr1</i> and <i>expr2</i> is not equal to 0. 0, otherwise.	(#4 < 0) (#4>10)
Logical NOT	! <i>expr</i>	Result is 1 if <i>expr</i> is 0. 0, otherwise	!(#4 > 0)

Bit expressions			
Bitwise AND	<i>expr1</i> & <i>expr2</i>	Result is the bitwise AND of <i>expr1</i> and <i>expr2</i> .	#4 & 0xFF
Bitwise OR	<i>expr1</i> <i>expr2</i>	Result is the bitwise OR of <i>expr1</i> and <i>expr2</i> .	#4 0x80
Bitwise NOT	~ <i>expr</i>	Result is the bitwise complement of <i>expr</i>	#4 & ~0x00

Filter expressions			
Filter	flt(<i>#ch</i> , <i>expr1</i>)	Computes a τ -filter over the values of channel <i>#ch</i> with a window size of <i>expr1</i> seconds.	flt(#4, 5)
Average	avg(<i>#ch</i> , <i>expr1</i>)	Computes an avg-filter over the values of channel <i>#ch</i> with a window size of <i>expr1</i> seconds.	avg(#2, 0.5)
RCLP	rclp(<i>#ch</i> , <i>expr1</i>)	Computes a RCLP-filter over the values of channel <i>#ch</i> with a RC of <i>expr1</i> .	rclp(#2,0.03033)
	rclphz(<i>#ch</i> , <i>expr1</i>)	Computes a RCLP-filter over the values of channel <i>#ch</i> with a filter frequency of <i>expr1</i> in Hz .	rclphz(#2, 5)

Variables			
m1, ..., m6		Variables for storing computation results. Stored values survive power off/on cycles. Will be reset to 0 after data download. (Pressing Empty/F3 in WinIt)	m1 = m1+1 m1 = m1+#4
p1, p2		Variables which can only be incremented. Stored values survive power off/on cycles. Cannot be reset to 0.	p1 = p1+1 p1 = p1+#4
x		Holds the value from the previous computation. Each channel has its own variable x. Reset to 0 at power on.	x = x+1 x = x+#4

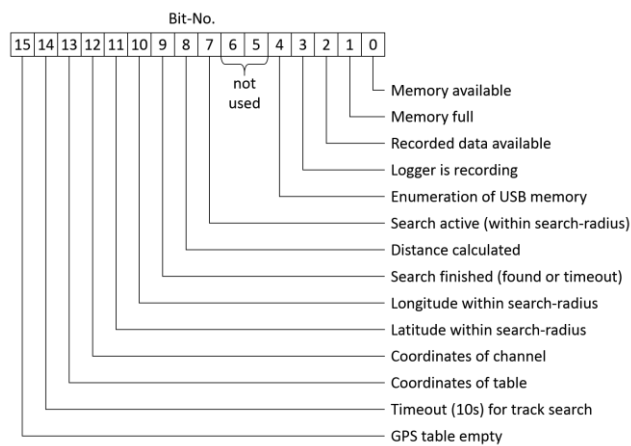
Conditional expressions			
Conditional execution	if(<i>expr1</i> , <i>expr2</i> , <i>expr3</i>)	The result is <i>expr2</i> if <i>expr1</i> is not zero. Otherwise, the result is <i>expr3</i> .	if(#3==0,#4,#5)

Assignment			
Assign	variable = <i>expr</i>	Stores the value of <i>expr</i> in a variable. The result of the assignment is the value of <i>expr</i> .	m1 = max(#3, #4)

Triconometric expressions			
Rad	rad(<i>expr</i>)	Converts the value of <i>expr</i> from degrees to rad.	rad (#4) rad(1)
Degree	deg(<i>expr</i>)	Converts the value of <i>expr</i> from rad to degrees.	deg(#4) deg(1)
Sine	sin(<i>expr</i>)	Result is the sine value of <i>expr</i> . (<i>expr</i> has to be in radians).	sin(#4) sin(1)
Cosine	cos(<i>expr</i>)	Result is the cosine value of <i>expr</i> . (<i>expr</i> has to be in radians).	cos(#4) cos(1)
Tangent	tan(<i>expr</i>)	Result is the tangent value of <i>expr</i> . (<i>expr</i> has to be in radians).	tan(#4) tan(1)
Arc sine	asin(<i>expr</i>)	Result is the arc sine value of <i>expr</i> . (<i>expr</i> has to be in radians).	asin(#4 / #5)
Arc cosine	acos(<i>expr</i>)	Result is the arc cosine value of <i>expr</i> . (<i>expr</i> has to be in radians).	acos(#4 / #5)
Arc tangent	atan(<i>expr</i>)	Result is the arc tangent value of <i>expr</i> . (<i>expr</i> has to be in radians).	atan(#4 / #5)
Atan2	atan2(Y,X)	Extension of atan which takes two real numbers as arguments to output the atan function value in a range of values of 360 (i.e. all four quadrants), and does not have to limit itself (like the normal arc tangent) to two quadrants.	atan2(#4 / #5)
Sine (degree)	dsin(<i>expr</i>) sind(<i>expr</i>)	Result is the sine value of <i>expr</i> . (<i>expr</i> has to be in degree).	dsin(70)
Cosine (degree)	dcos(<i>expr</i>) cosd(<i>expr</i>)	Result is the cosine value of <i>expr</i> . (<i>expr</i> has to be in degree).	dcos(70)
Tangent (degree)	dtan(<i>expr</i>) tand(<i>expr</i>)	Result is the tangent value of <i>expr</i> . (<i>expr</i> has to be in degree).	dtan(70)
Arc sine (degree)	dasin(<i>expr</i>) asind(<i>expr</i>)	Result is the arc sine value of <i>expr</i> . (<i>expr</i> has to be in degree).	dasin(70)
Arc cosine (degree)	dacos(<i>expr</i>) acosd(<i>expr</i>)	Result is the arc cos value of <i>expr</i> . (<i>expr</i> has to be in degree).	dacos(70)
Arc tangent (degree)	datan(<i>expr</i>) atand(<i>expr</i>)	Result is the arc tangent value of <i>expr</i> . (<i>expr</i> has to be in degree).	datan(70)

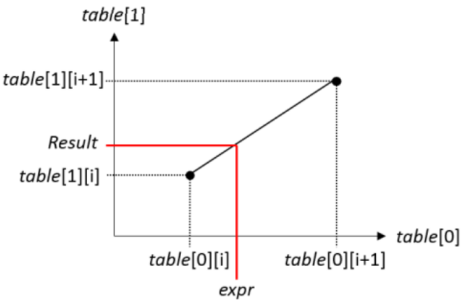
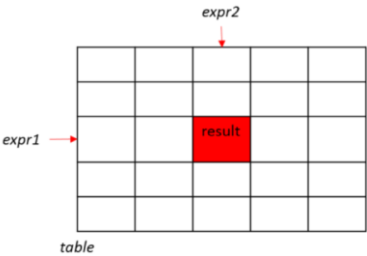
Other expressions			
Best laptime	bestLap()	Currently stored best laptime. Reseted if AutoZero-function at Laptime-Event is activated or Memory is cleared via EmPy (F3) button.	bestLap()
Delay	Delay(#ch, <i>expr1</i>)	Delaying #ch by value of <i>expr1</i> [sec].	delay(#4, 5.05) delay(#4, #5)
Random number	random(Range)	Output a random, integer number between 0 and <i>Range</i> . Output value is changing with sampling rate of CALC channel used.	random(100)

GPS related expressions			
GPS speed 3D	gpsspeed3d()	Output of a speed determined via GPS, which also considers the change in Altitude	gpsspeed3d()
GPS speed North/South	gpsspeedn()	Output of the GPS speed valid for the North/South direction only	gpsspeedn()
GPS speed East/West	gpsspeede()	Output of the GPS speed valid for the East/West direction only	gpsspeede()
GPS speed up/down	gpsspeedd()	Output of the GPS speed valid for the up/down direction only	gpsspeedd()
Integer Time of Week	itow()	GPS Time of Week in Milliseconds	itow()
Fraction Time of Week	ftow()	Fractional part of Integer Time of Week in Nanoseconds	ftow()
GPS week	gpsweek()	GPS week counter since beginning of year 1980 with rollovers every 1023 weeks	gpsweek()
GPS LeapSecond	gpsleaps()	Time difference between GPS- and UTC-Time (GPSTime=UTCTime + LeapSecond)	gpsleaps()

Debug expressions			
Logger status	stat (mask)	<p>Delivers the status of the logger. The result is a bit-coded value. <i>mask</i> can be used for masking out certain bits. If <i>mask</i> = 0 then all bits are delivered by stat.</p> 	stat (0)
Sector pointer	sptr()	Delivers the pointer index	sptr()
Sector number	secnr()	Delivers the current sector number	secnr()
Sector count	seccnt()	Delivers the number of sectors to save	seccnt()
CPU load	cpu_load_max()	Delivers the maximum CPU load	cpu_load_max()
	Cpu_load_avg()	Delivers the average CPU load	Cpu_load_avg()
	CPUload()	Delivers the current CPU load	CPUload()
Receive-counter	Sioirq()	16-bit counter for incoming characters	Sioirq()
Channel rate	chrate(#ch)	Delivers the sampling rate of the entered channel	chrate(#ch)
Channel value	chval(#ch)	Delivers the value of the entered channel	chval(#ch)

CAN errors	cantxerr(CAN#)	Error counter of the CAN line while sending, value $\geq 128 \Rightarrow$ bus off	cantxerr(CAN1)	
	canrxerr(CAN#)	Error counter of the CAN line while receiving, value $\geq 128 \Rightarrow$ bus off	canrxerr(CAN1)	
	canerr(CAN#)	<p>Bit-No.</p> <p>value 1 \Rightarrow bit error; 2 \Rightarrow form error; 3 \Rightarrow stuff error; 4 \Rightarrow other error not used Rx-error Tx-error CAN bus error CAN bus off</p>	canerr(CAN1)	

MinMax expressions				
Minimum	$\min(expr1, expr2)$	Result is the smaller of $expr1$ and $expr2$.	$\min(\#4, 5)$	
Maximum	$\max(expr1, expr2)$	Result is the greater of $expr1$ and $expr2$. (same as "<")	$\max(\#3, \#speed)$ $\max(\#speed, 10)$	
Hold minimum	$\text{hmin}(expr1, expr2)$	<p>$expr2$ is called "hold-period". Result is the minimum value of $expr1$. This value is held for the hold period. If $expr1$ evaluates to a smaller value during the hold period that value is the new result.</p>	$\text{hmin}(\#4, 5)$	
Hold maximum	$\text{hmax}(expr1, expr2)$	<p>$expr2$ is called "hold-period". Result is the maximum value of $expr1$. This value is held for the hold period. If $expr1$ evaluates to a bigger value during the hold period that value is the new result.</p>	$\text{hmax}(\#4, 5)$	

Table expressions			
Table	<code>tab(<i>table</i>, <i>expr1</i>)</code>	<p><i>table</i> is a table with two rows, <i>table</i>[0] and <i>table</i>[1]. The result of the tab-function is computed by linear interpolation as shown below:</p> 	
Linearize	<code>lin(<i>expr1</i>, <i>expr2</i>, <i>table</i>)</code>	<p><i>table</i> is a 2-dimensional table. Result is the value <i>table</i>[<i>expr1</i>][<i>expr2</i>].</p> 	

7 Appendix

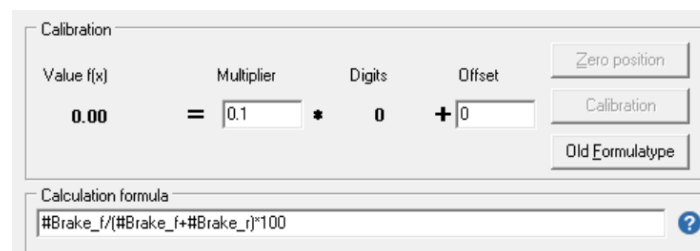
7.1 Examples

All examples are aligned to use CALC channels for displaying at 2D Dashboard!

7.1.1 Brake Balance Calculation

The front and rear brake pressure values (e.g., #Brake_f & #Brake_r) are measured by a connected data logging system, then sent to the 2D Dash via the CAN bus.

The 2D Dash reads the values from the CAN bus on channel 4 (#Brake_f) and channel 5 (#Brake_r).




Further Information

By entering the channel numbers, e.g. #4, the physical values of this channel are used in the calculation. If the position of the channel changes, you must modify the Calc Channel!

If the actual brake bias is **55.26%** (bias to the front), and a result with a resolution of 1% accuracy is acceptable, a <Multiplier> of 1 can be used, giving a calc channel result of "55". If a higher resolution is needed, e.g. 0.1%, a multiplier of 0.1 would be used to give a channel result of "55.3". The <Offset> value can remain at the default value of "0".

Entering a smaller number inside the <Multiplier> field will enable a higher level of resolution, with smaller steps between each measurement value change.



Further Information

If the entered value of <Multiplier> becomes too low, e.g. 0.00001, the device can no longer display the full range of physical values. As a rule, only use a <Multiplier> value that is practical for the application!

7.1.2 Laps per run and target laps

Because the Event channel *Laps* is not reseted at power-on but only by pressing empty button or downloading data, the displayed laps from different runs are always incremented.

To get only the laps of a run a CALC channel is used.

Calculation formula:

if(der(#Laps) >0, x+1, x)

Thereby, the derivation of Event channel *#Laps* is checked and because of the derivation change of *#Laps* it is incremented at each laptrigger.

By powering the system, the CALC channel is set to 0 again and also the lap calculation starts at 0 again.

The created CALC channel can be used to create a counter which down counts the laps from a given value.

Calculation formula:

15-#LapsPerRun

7.1.3 Triggering an alarm channel when Oil pressure is low

Under hard braking or cornering engine Oil pressure can suddenly drop and seriously damage an engine. If this occurs the driver should be warned by a dashboard alarm. With the engine on idle the Oil pressure is usually low but is of no significance as the engine is under no load. A CALC channel can be used to only trigger a low oil pressure warning when a genuinely problematic situation is occurring.

Channel 2 (*#RPM*) and Channel 3 (*#Oil pressure*) are read from the CAN bus. A channel is calculated that has the value of "1" when the engine is above 3000rpm **AND** the Oil pressure is below 1 bar, otherwise the CALC channel has a value of "0".

Calculation formula:

(#OilPressure<1)&&(#RPM>3000)

A <Multiplier> of 1 is entered as the channel can only have the values of 0 or 1 and therefore a resolution of 1 is appropriate. The <Offset> value can remain at the default value of "0".



Further Information

When a calculated channel is created to give a Boolean output (0 or 1), the <Multiplier> should always be 1!

An alarm channel will use this CALC channel as its trigger, turning on warning LED's when a dangerous low oil pressure event happens.



Documentation reference

For more information about Alarms please see the respective chapter in manual XXX from our website www.2D-Datarecording.com/manuals

7.1.4 Calculation of Fuel Consumption 1



Further Information

Fuel consumption is reseted when dashboard is powered off

Using a fuel injector counter signal #Fuelcons, the total fuel injected to the engine can be calculated by the 2D Dash. It is already known that a single injector pulse deposits 0.032768 litres of fuel into the engine.

Calculation formula:

$$\text{sum}(\text{if}(\text{der}(\#\text{Fuelcons}) < 0, 0.032768, 0))$$

In this formula the derivation of the injector counter signal (#Fuelcons) is calculated. If that calculation turns negative, 0.032768 litres of fuel (depending on the injector signal) are added to the result so far. This is an accumulative calculation that will continue to increase in value until the engine is turned off.

For this calculation, removing the power supply from the 2D Dash will cause the calculation channel will start again from zero.

7.1.5 Calculation of Fuel Consumption 2



Further Information

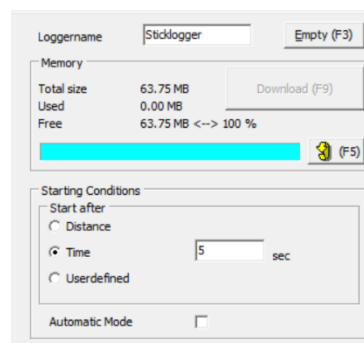
Reset value via Button F3 in the program Winit

Following a similar principle to the previous example, this formula provides a different method of resetting the fuel consumption value. This example formula will maintain the previous session value of Fuel consumption, even after power has been switched off!

Calculation formula:

$$m1 = m1 + \text{if}(\text{der}(\#\text{Fuelcons}) < 0, 0.032768, 0)$$

The calculation occurs the same as before, only in this case the result is stored in the dashboard as long as it is not reset by the program *Winit*.



Reset is performed using the button <F3> or <Empty>, as shown below.

**Important information**

Performing this action will also reset any stored data (e.g. Lapttime) from inside the 2D Dash!

7.1.6 Calculate the driven mileage (Reset in Winit via button F3)

Another example of using an accumulative calculation channel is explained below. Here the vehicle speed reference channel is used to determine the distance travelled during a session.

Calculation formula:

$$m1=m1+(\#Speed*3.6/Samplingrate)$$

The speed signal is multiplied by 3.6 and divided by the channel sampling rate to have a signal per second in meters. The accumulated value of distance travelled is **measured in metres**, and stored in the variable **m1**.

As in the previous example, Reset is performed using the button **<F3>** or **<Empty>** within the 2D program Winit.

7.1.7 Calculate the driven mileage (Permanent accumulation)

Alternatively, by using the variables **p1** and **p2**, the driven mileage calculation can be permanently accumulated through the life of the vehicle (not resettable).

Calculation formula:

$$p1=p1+(\#Speed*3.6/Samplingrate)$$

7.1.8 Count the number of detonations in a run

The channel **#Deto** has a value of 1 when detonation occurs and 0 when the engine runs smoothly. The total value of detonation events are counted with the help of the variable **m1**.

Calculation formula:

$$m1=(m1+\#Deto)$$

7.1.9 Reset variable m1 via Button

Instead of connection to a PC, then using 2D program Winit to reset the value of **m1**, an external button (Analog channel) can be used to reset the calculation value while on track. For this to work, an additional Calculation channel is created as follows.

Calculation formula:

$$m1=(m1+\#Deto)$$

The analogue signal of channel **#ANA1** is continuously checked, if its value drops below 2500 digits (short circuit to GND), the variable **m1** is reset to 0. In this example an analog signal was used to reset the value of **m1**, however CAN or Digital channels can also be used for this!

7.1.10 Counting the number of samples

If you require to count the number of samples, the Calculation formula entered is: **x+1**

At every sample point, 1 is added to the previous value of **x**. this process continues until power is removed from the 2D Dash.

7.1.11 TimeCounter (PitTimer)

This example is very useful where a minimum pitlane time is enforced by regulation of race classes or for any other timer applications.

By using the PitTimer-example a timer can be created which shows the time left before the vehicle can leave the pits after a pitstop.



Important information

By using a 2D dashboard that is able to store the values of variables during shutdown, the PitTimer will also work if the car is shut down during the pit stop! 2D already offers a similar function for the GT3 Endurance class with the SectionTime event, but it is not intended to switch off the car during the pitstop.

MidiDash, **BigDash** and **TFTDash** are able to handle power-off phases during pit stop and PitTimer will show the correct, remaining time when system is powered again.

Time input channels:

As time input channels the RealTimeClock as well as the GPS-Time channels can be used.

2D Dashboard	RTC-Time	GPS-Time
	Digital group	GPS channels
MiniDash	No RTC available	#HHMM, #SSHH
MidiDash	#RTC_HHMM, #RTC_Second	
TFTDash		
BigDash	#HHMM, #Second	

First, a SecondOfDay channel (#SOD) is created in Dashboard CALC channels, which combines the time channels of the Dashboard into one channel that only counts **up** the seconds of day.

hh	div(#Hhmn,1)	15000	0.001	15.0
SOD	#hh*3600+(#HHMM-#HH)*6000+#second	54446	1.000	54446.0



Further Information

Since the number of characters per line is limited, the first part (creation of hh) was moved to a separate CALC line.

Because at Dashboards only 16-Bit-Channels are available, the highest possible SOD value is $2^{16} = 65535$.

SOD value 65535 is a 6:12.15 pm because than the SOD value is starting at 0 again.

Hence, the PitTimer is not working correctly when pit entry is before 6:12.15 pm and switching the bike on is after 6:12.15 pm. For every other case it works correctly.

Trigger channels

The PitTimer function has been designed so that any channel of the dashboard can be selected as a triggers for the timer start.

In this example the Dorna-TransponderX2 message (*#x2_CF*) is used for triggering both PitEntry and PitExit.

The PitTimer is started when *#x2_CF* has the value 3 (*TrgInVal* 3) and stopped with value 18 (*TrgOutVal* 18).

TrgCH	#x2_CF	0	1.000
TrgCH2	#x2_CF	0	1.000
TrgInVal	3	0	1.000
TrgOutVal	18	0	1.000



Documentation reference

For more information about TransponderX2 messages please see the manual *Laptiming via TransponderX2* from our website

www.2D-Datarecording.com/manuals

Any other, as well as different trigger channels for starting and stopping the timers can be used by selecting different channels at *TrgCh* and *TrgCh2*.

Power cycle handling:

By using the CALC-variable *m1*, the value current value of *m1* is stored in Dashboard's memory even when the power-supply is switched off.

m1	m1=if(#TrgCh==#TrgInVal, #SOD, m1)	0	1.000
m1	m1=if(#TrgCh2==#TrgOutVal, 0, m1)	0	1.000
TimeSince	if(m1==0, 0, #SOD-m1)	0	1.000

The resulting channel *#TimeSince* shows the expired time since the start-trigger and is reseted to 0 at stop trigger.



Important information

Using PitTimer with power-cycle is only possible in **MidiDash**, **BigDash** and **TFTDash**! PitTimer can also be used at **MiniDash** but not together with power-cycles.



Important information

In **MidiDash** and **TFTDash** instead of *m1* variable, *p2* variable must be used for using PitTimer together with power-cycle.

Further use:

Further CALC-lines can be used to create different other timers like a countdown (*#Timer*) from a respective value (*#BoxWait*) at start-trigger to 0.

BoxWait	20	20	1.000	20.0
Timer	if(#TimeSince<#BoxWait,#BoxWait-#TimeSince,0)	17	1.000	17.0

The created channel *#TimeSince* can be used for displaying and/or for alarms

Parameter

Turned on	<input checked="" type="checkbox"/>
Compare function	>
Alarm threshold	0.0
Channel to check	Ch 58: TimeSince
Channel to show	Ch 59: Timer
Minimum alarm active time	0 ms
Minimal duration to show alarm	0 ms
Maximal duration to show alarm	0 ms
Text to show	PitTimer
Output channel to activate	<None>
Flashing LEDs	<input type="checkbox"/>
Priority	0

Alarm channel setting:

PitTimer-Setting:

Name	Sensor info	Digits	Multiplicator	Value
TrgCH	#x2_CF	0	1.000	0.0
TrgCH2	#x2_CF	0	1.000	0.0
TrgInVal	3	3	1.000	3.0
TrgOutVal	18	18	1.000	18.0
BoxWait	20	20	1.000	20.0
hh	div(#Hhmm,1)	10000	0.001	10.0
SOD	#hh*3600+(#HHMM-#HH)*6000+#second	38956	1.000	38956.0
m1	m1=if(#TrgCh==#TrgInVal, #SOD, m1)	38943	1.000	38943.0
m1	m1=if(#TrgCh2==#TrgOutVal, 0, m1)	38943	1.000	38943.0
TimeSince	if(m1==0, 0, #SOD-m1)	13	1.000	13.0
Timer	if(#TimeSince<#BoxWait,#BoxWait-#TimeSince,0)	7	1.000	7.0

Channel	Analysis
TrgCh	#x2_CF
TrgCh2	#x2_CF
TrgInVal	3
TrgOutVal	18
BoxWait	20
hh	div(#Hhmm,1)
SOD	#hh*3600+(#HHMM-#HH)*6000+#second
m1	m1=if(#TrgCh==#TrgInVal, #SOD, m1)
m1	m1=if(#TrgCh2==#TrgOutVal, 0, m1)
TimeSince	if(m1==0, 0, #SOD-m1)
Timer	if(#TimeSince<#BoxWait,#BoxWait-#TimeSince,0)



Further Information

To safe CALC channels the PitTimer function can be stripped down to 5 lines only by using channel numbers instead of names and waive on extra channels for e.g. *TrgInVal!*

The five lines are showing the PitTimer function at **BigDash** (#80 → #HHMM, #81 → #Second)

Name	Sensor info
SOD	((div(#80,1))*3600)+(#80-(div(#80,1)))*6000+#81
m1	m1=if(#x2_cf==3,#SOD,m1)
m1	m1=if(#x2_cf==18,0,m1)
TimeSince	if(m1==0,0,#SOD-m1)
Timer	if(#TimeSince<20,20-#TimeSince,0)

setting:

7.1.12 AnalogTest toolchain

By using different 2D modules with their special functions, a toolchain can be built up with the help of the CALC channels, which makes it possible to check the input analogue channels of an AD module.

The processed data is relevant for both the development and production departments. The former can make statements about the hardware and software of the modules, while the latter is limited here to assessing the quality of manufactured modules.

The 2D software package enables automatic processing, visualisation and documentation of the measurements carried out by means of provided module and software settings, scripts and toolbar buttons.

Please contact 2D via the [contact form](#) for further information.

7.1.12.1 Test description

Three modules are needed to set up this toolchain:

DA-Modul	- Generating the signals needed for the measurement with <i>Online Calculation Channels</i> (CALC channels)
AD-Modul	- Converts the analogue signals generated by the DA module to digital digit values and sends them via CAN bus to the logger.
Logger	- Recording of the CAN-bus channels sent by the AD module

A more detailed description of how the DA module generates the analogue channels via the CALC channels can be found in the following chapter 7.1.12.2.

The analogue signals output by the DA module (\triangleq simulator) are received and recorded by the data logger via the analogue inputs. The recorded analogue values are then compared in the 2D analysis software with the target values of the DA module, which are also recorded in the logger.

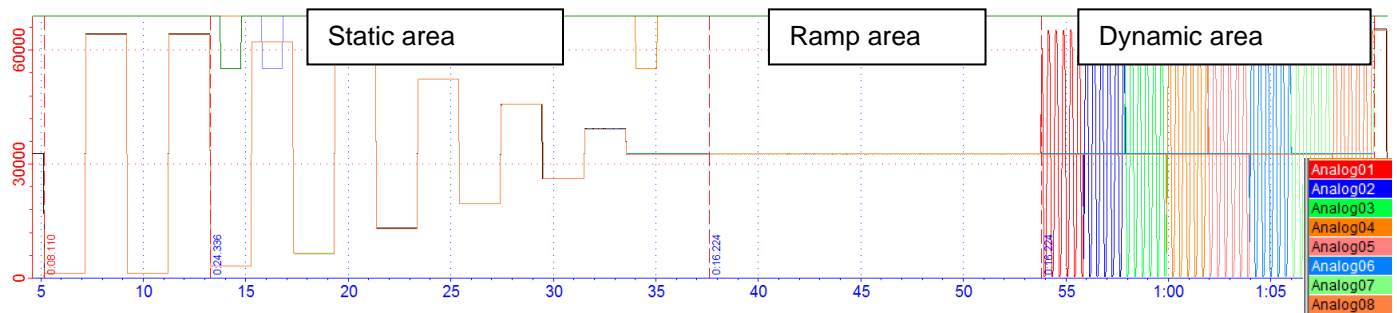
Further calculations can then be used to determine the values needed to check the modules.

The signal properties are evaluated:

- Signal value (enables statements about amplification)
- Noise (enables statements about filter equipment)
- Crosstalk (enables statements about mutual influence of the CH)

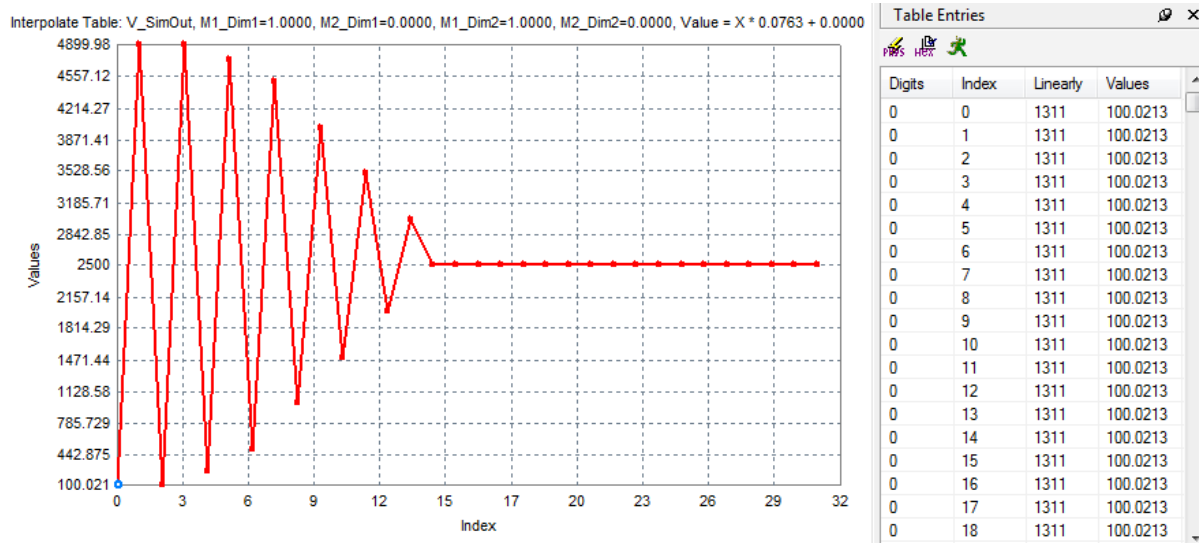
7.1.12.2 Test sequence

In order to be able to implement the toolchain described above, a test sequence is needed that outputs the required signals on 8 channels, independently of each other. For this purpose, a test cycle was developed that enables the execution of the tests in sequence. The cycle is divided into the following test areas, all of which are realised purely via the internal CALC channels of the DA module.



1. Static area (½ Cycletime)

The differences between the nominal and actual voltages as well as the noise of the channels are to be calculated later in the static range. For this purpose, the eight analogue channels synchronously output a voltage defined in the table *Tab_V_SimOut*. The table is stored in the setting of the DA



module.

2. Ramp area (¼ Cycletime)

In the ramp area, it must be checked whether the modules used are 12- or 16-bit versions. For this purpose, all 8 analogue channels synchronously output the same voltage, which is increased digit-selectively and cyclically by 63 digits, starting at 2.5V (\triangleq 32768 digits).

3. Dynamic area (¼ Cycletime)

In the dynamic range, the crosstalk of the channels and the settling time are to be calculated.

For this purpose, 1 analogue channel alternately outputs a square-wave voltage, while the other analogue channels output a constant voltage (2.5 V).