This short manual describes some functions of the 2D dashes, which are not yet described in the general dashboard manual or may differ from the explanations there.

You can find the general dashboard manual on the 2D website 2d-datarecording.com: Downloads, manuals ⇒ general dashboard manual

Content of this short manual:

How to program shift lights 2
How to set an alarm 4
How to program the GAP function 6
How to use the Delta Sections function 9
How to program the Switch function 10
X2 settings 12
How to check the availability of the X2 transponder 12
How to display flags and other X2 related messages 13
Calculation functions of the calc-channels 15
Appendix 19
Portrayal of X2 Team notification and Box call 19
How to create a GAP table with the Analyzer 20

How to program shift lights

Shift lights indicate the RPM signal with LEDs. That means the higher the RPM signal is the more LEDs are turned on. You can program the LEDs individually – you decide when which LED should turn on and which color will be indicated.

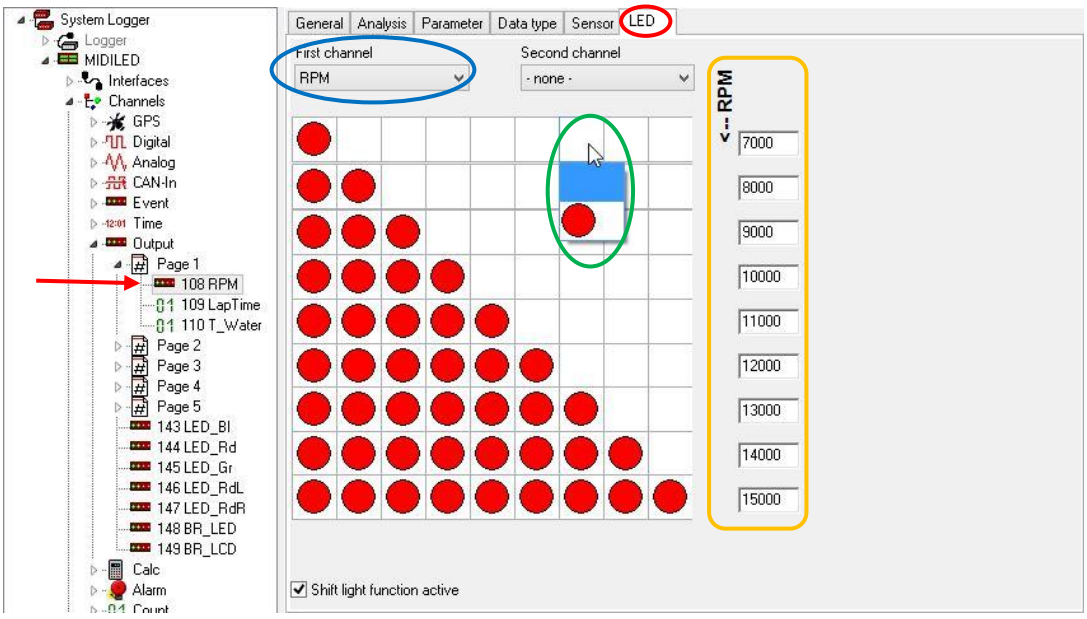
This example assumes that RPM and gear data are available in your dash.

To indicate the RPM data and to adjust the LEDs, go to “Output” and select a page. Click on the first channel. It is marked with the LED-bar.

Select the tab “LED” and set the first channel (marked in blue) to your RPM channel. By clicking on the corresponding fields you can choose if the LED should be turned on or not (marked in green). On the right side you can adapt the RPM (marked in orange).

If you program it like shown in the figure below you can identify the indicated RPM value:

- 7000 – 7999: first LED
- 8000 – 8999: first and second LED
- 9000 – 9999: first, second and third LED
- …



Confirm all your changes with <**Apply**>.

If you want you can add a second channel, for example “Gear”. Then you can adjust the RPM and LEDs to the used gear.

To indicate the RPM data as a function of the gear, go to “Output” and select a page. Click on the first channel. It is marked with the LED-bar.

Select the tab “LED” and set the first channel to your “RPM” channel and the second channel to your “Gear” channel (marked in blue). By clicking on the corresponding fields you can choose if the LED should be turned on or not. On the right side you can adapt the RPM (marked in orange) and beneath it you can add more “gears” or delete them (marked in green).

If you program it like shown in the figure below you can identify the indicated RPM value. In this example the first LED is left out to be used as an alarm output or status LED.

Gear 0:

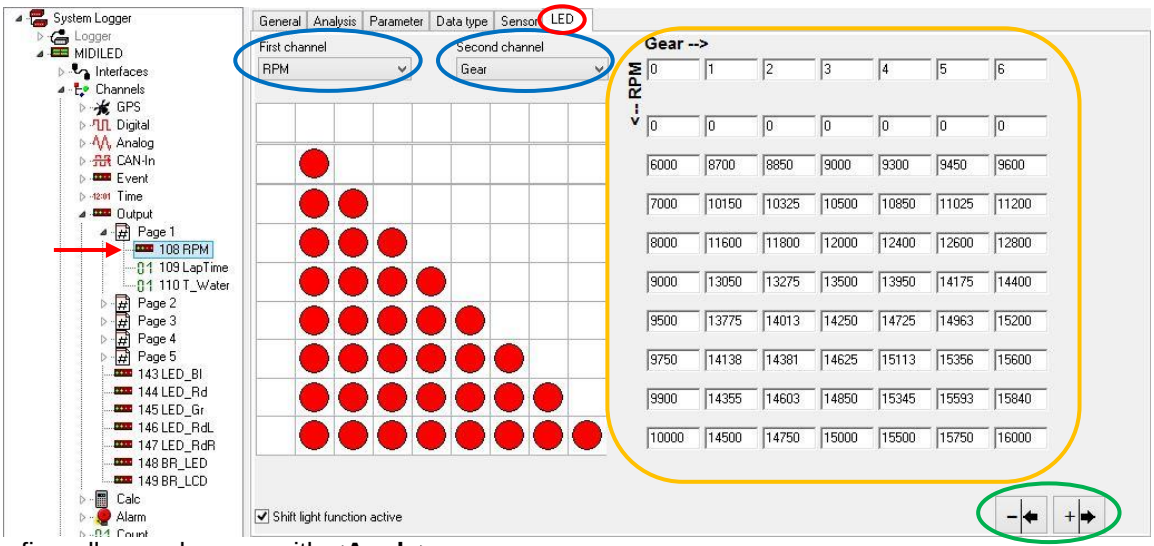
- 6000 – 6999: second LED
- 7000 – 7999: second and third LED
- 8000 – 8999: second, third and fourth LED
- …

Gear 1:

- 8700 – 10149: second LED
- 10150 – 11599: second and third LED
- 11600 – 13049: second, third and fourth LED
- …

Gear 2:

- 8850 – 10324: second LED
- 10325 – 11799: second and third LED
- 11800 – 13274: second, third and fourth LED
- …



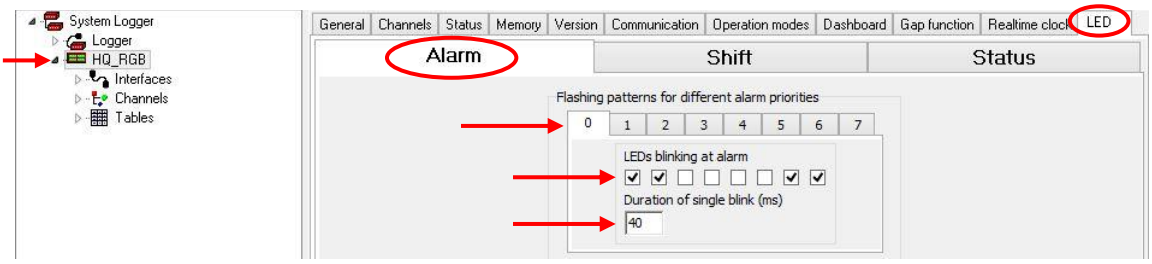
Confirm all your changes with <**Apply**>.

How to set an alarm

Alarms are events with highest priority. Therefore they will overwrite all other information currently displayed on the 2D dash. There are up to 8 different alarm priorities programmable. Within one alarm priority the alarm-channel with the smallest number has the highest priority, for example *ALARM#1* has a higher priority than *ALARM#5* if both are programmed with the same alarm priority.

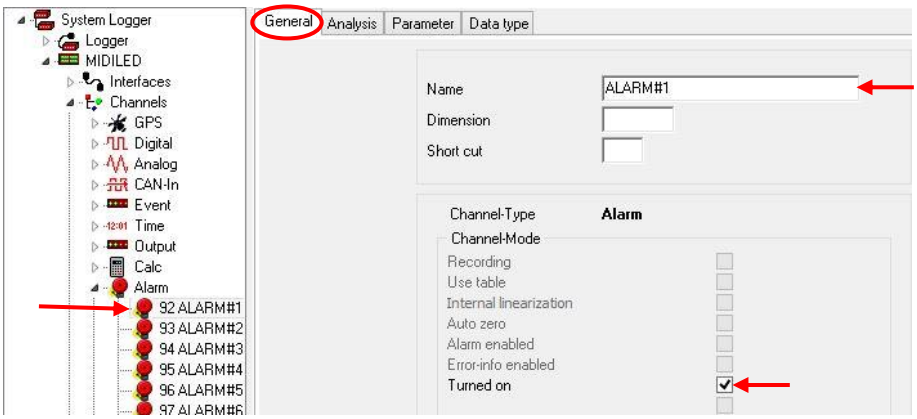
The first step in programming alarms is to specify which LEDs to use for alarms with flashing LEDs. *Note:* Once specified, all alarms of one alarm priority with flashing LEDs use the same LEDs.

Select the dash in the system tree and select the tab “LED”. On the tab “Alarm” you can set any combination of the LEDs[1] for blinking alarms. There you can also adapt the “blink frequency” to your needs: change the “Duration of a single blink”.



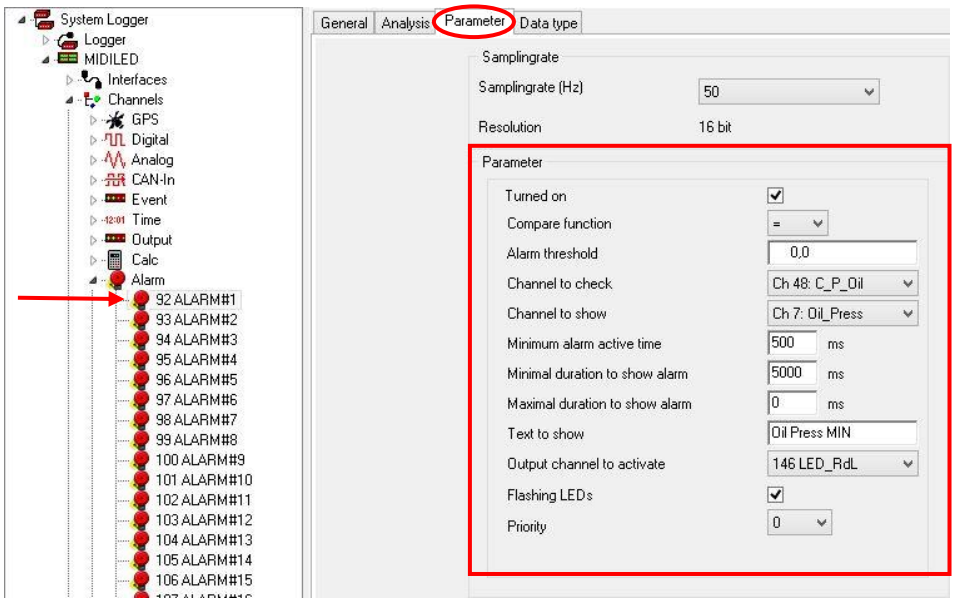
To set an alarm select your dash in the system tree. Go to “Channels” ⇒ “Alarm” and select the alarm channel you want to program.

In tab “General” you turn it on and rename this channel if you want. In some modules the alarm-channel is named after the channel to check.



On the tab “Parameter” you can parameterize the alarm. In the field “Parameter” you can select the trigger condition which will start the alarm, the channel which activates the alarm and which will be indicated, the duration of the alarm (if you set the maximal duration to zero, the alarm will be indicated as long as it lasts or the minimal duration) and you can enter a text which will be displayed. In “Output channel to activate” you can select a single LED, which will light up continuously, or if you want to have flashing LEDs select “Flashing LEDs” to activate the blinking LEDs (as described at the beginning of this section). At the end you find a drop down list to assign the alarm priority to the alarm channel.

[1] The number of LEDs depends on which module you are programming.



Confirm all your changes with <**Apply**>.

For example you want to program an alarm for the oil pressure being too low.

Turned on	Turn it on to activate this alarm channel
Compare function	Select the compare function; in this example "**=**" (equals)
Alarm threshold	Enter the value, to activate the alarm; in this example the temperature when the value of the calc-channel is "1"
Channel to check	Select the channel, which will be checked; in this example the calc-channel "C_P_Oil"
Channel to show	Select the channel, which value will be displayed when the alarm is triggered; in this example it´s the channel "Oil_Press"
Minimum alarm active time	The time delay between points of time where the channel value triggers the alarm condition (threshold) until the alarm is displayed
Minimal duration to show alarm	The alarm is active for minimum this time duration – even if the oil pressure is not too low anymore
Maximal duration to show alarm	The alarm is active for maximum this time duration – even if the oil pressure is still too low. If you set this value to zero, the alarm will last as long as the alarm condition is true – or the minimum duration lasts
Text to show	Enter a short text which will be shown when the alarm is active.
Output channel to activate	Select a single LED which lights up continuously
Flashing LEDs	If you activate this, the LEDs of the blinking alarm will be activated
Priority	Select one of the alarm priorities.

If you want to use the left (LED_RdL) or right (LED_RdR) LED of the shift lights for indicating an alarm, you have to leave out that LED on all pages when you program your shift lights (see example for programming shift lights).

If more than one alarm with entered text is active at the same time, up to two text are shown parallel. After 5s the texts of active alarms will rotate/change.

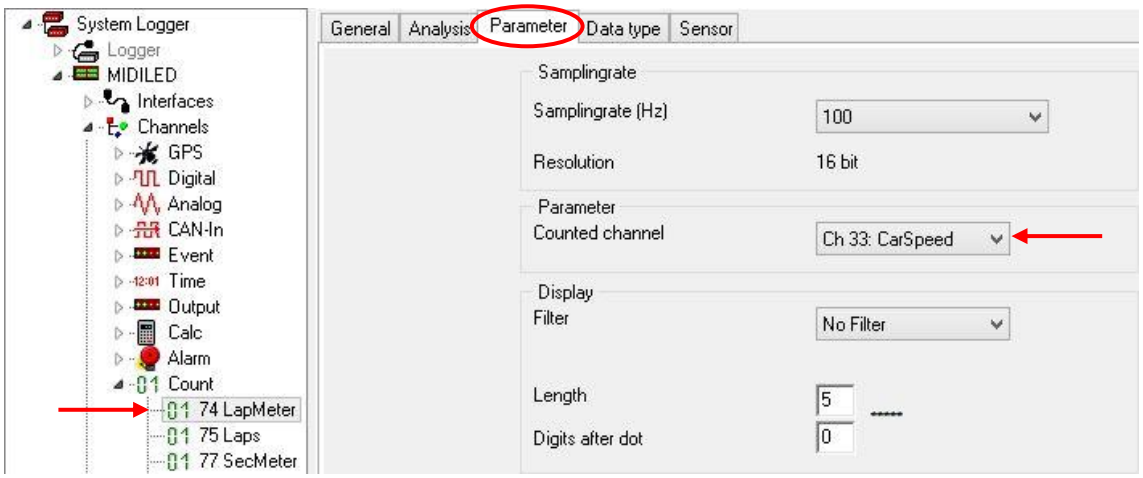
Available in **BigDash** (since version 112): If you enter a text to show like normal (***Alarmtext***) it will only be displayed on the rider pages (page 1 & 2).
If you want to show the text on all pages you have to enter the text like this: ***<Alarmtext>***

How to program the GAP function

The GAP function compares the actual driven lap to your best lap. It is assumed that you have already set the function of your laptime channel correctly.

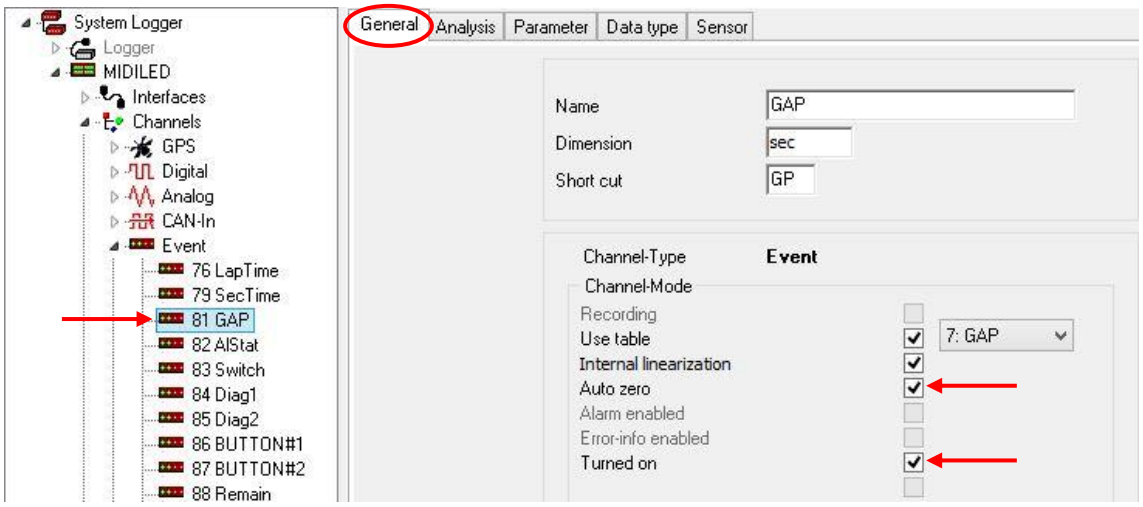
You can create a GAP table yourself while driving or use the Analyzer to create a GAP table[2] which you store inside your dash.

To be able to use this function you have to check if the channel “LapMeter” uses a valid speed channel. Therefore you select your dash in the system tree and go to “Channels”, “Count” ⇒ “LapMeter”. In tab “Parameter” you can select the speed channel (“Counted channel”).



Confirm your changes with <**Apply**>.

To program the GAP channel you go to “Channels”, “Event” ⇒ “GAP”. There you turn the channel on (tab “General”). If you want the GAP function to be self-updating, you activate “Auto zero”.



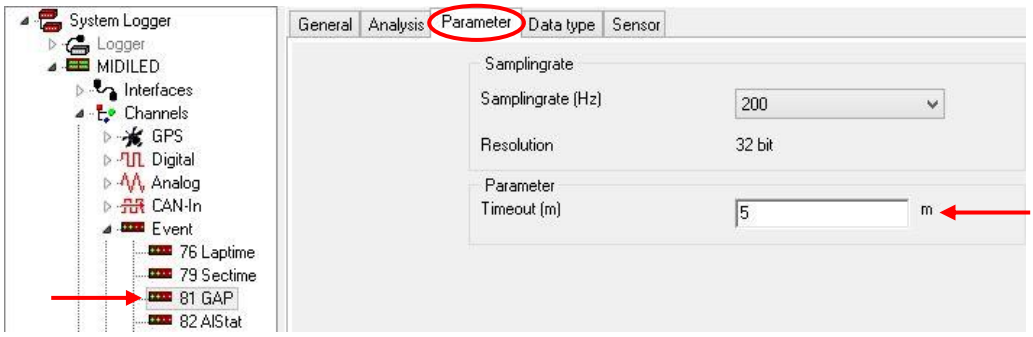
If you want to record a GAP table yourself, you have to activate „Auto zero“! Otherwise the function is not able to enter the values inside the table.

Auto zero active on channels LapTime and GAP: Power off/on will delete the GAP table!

Auto zero active on channel LapTime: Power off/on will delete the BestLap, but the GAP table is still available.

[2] In the appendix you can find a short description on *How to create a GAP table with the Analyzer*

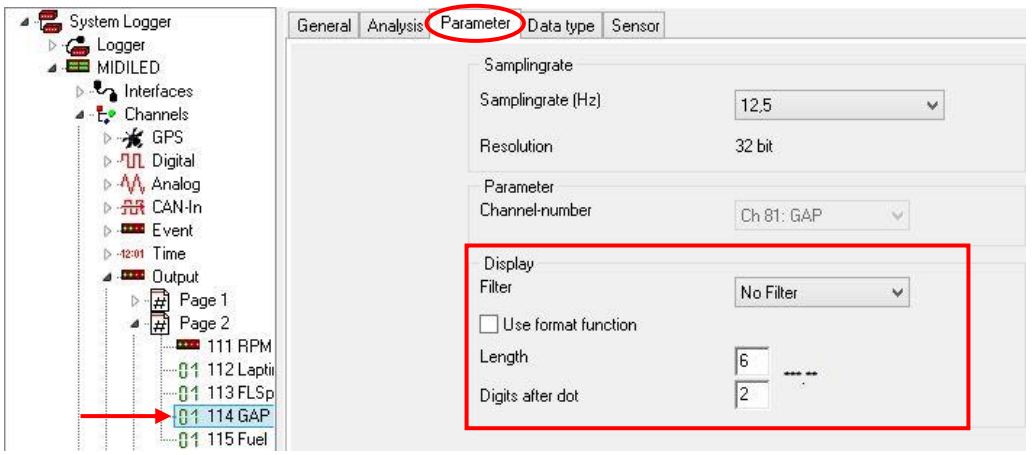
In tab “Parameter” you can enter a timeout value, which is the minimum track length. It is used to ignore faster lap times due to taking a short cut.



If your lap time is faster than your currently best lap, but the lap length is shorter than the entered minimum track length, then your lap time will not be used as a reference time for following laps.

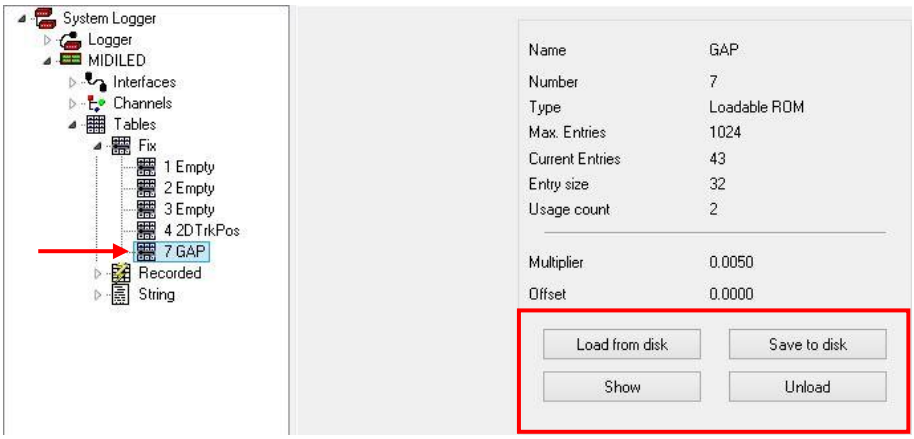
Confirm your changes with <**Apply**>.

If you put this channel on an output channel, you use it without a format function. The value will be displayed in seconds. Please remember by entering the length, that the “-” for indicating a faster time is a character, too.



Confirm your changes with <**Apply**>.

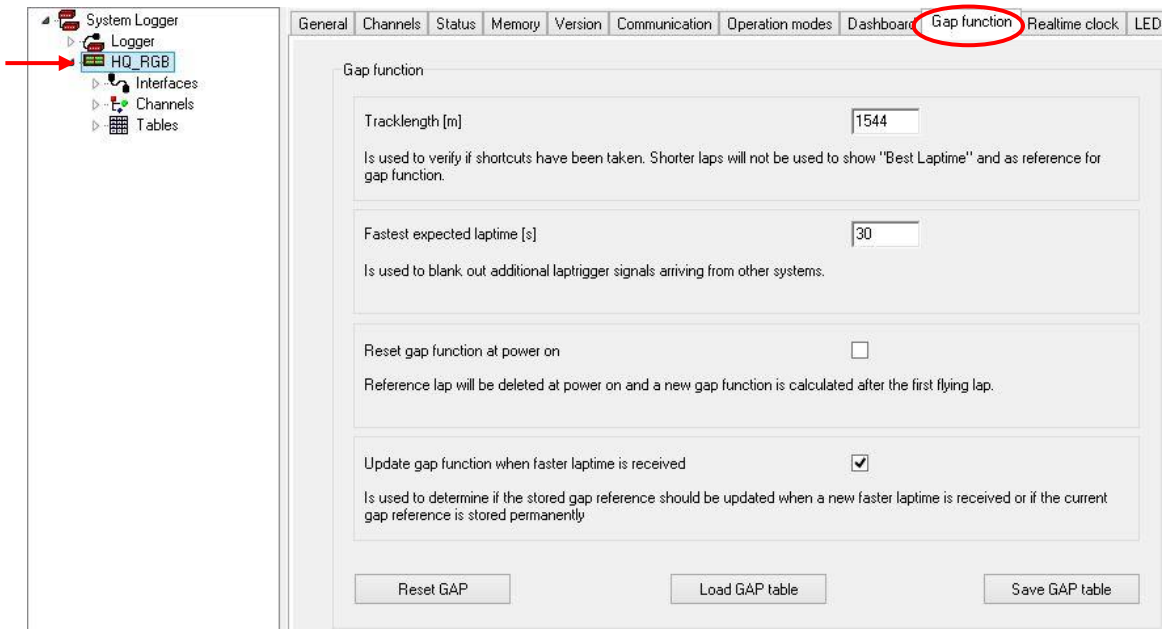
Within the table section (“Tables”, “Fix” ⇒ “GAP”) you can load a table from your PC, save your GAP table to your PC, have a look at the table or empty the table inside your dash:



Save your changes with <**Apply**>.

If you just want to modify the setting of the GAP function you can also use the tab “Gap function” of the dash:

Select the dash in the system tree and select the tab “Gap function”.



Tracklength [m]
Here you modify the timeout of the Gap channel:
Timeout of channel Gap = tracklength -10%

Fastest expected laptime [s]
Here you modify the timeout of the LapTime channel:
Timeout of channel LapTime = fastest expected laptime -10%

Reset gap function at power on
Here you can decide if you want to reset the GAP function each time you turn the dash´s power off. (Same as “auto zero function” of channel laptime.)

Update gap function when faster laptime is received
Here you can also enable the “auto zero function” of channel Gap.

Reset GAP
Unloads the GAP table.

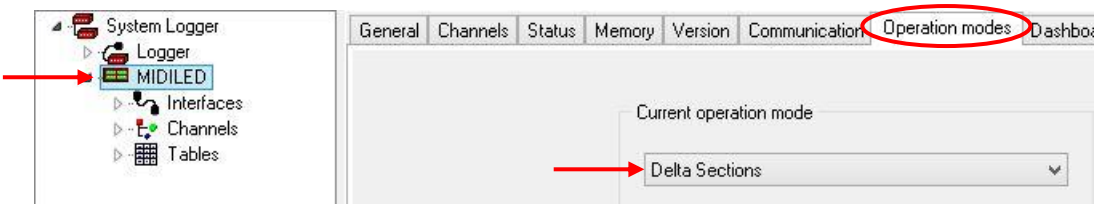
Load GAP table
Loads a GAP table from your PC.

Save GAP table
Saves the current GAP table to your PC.

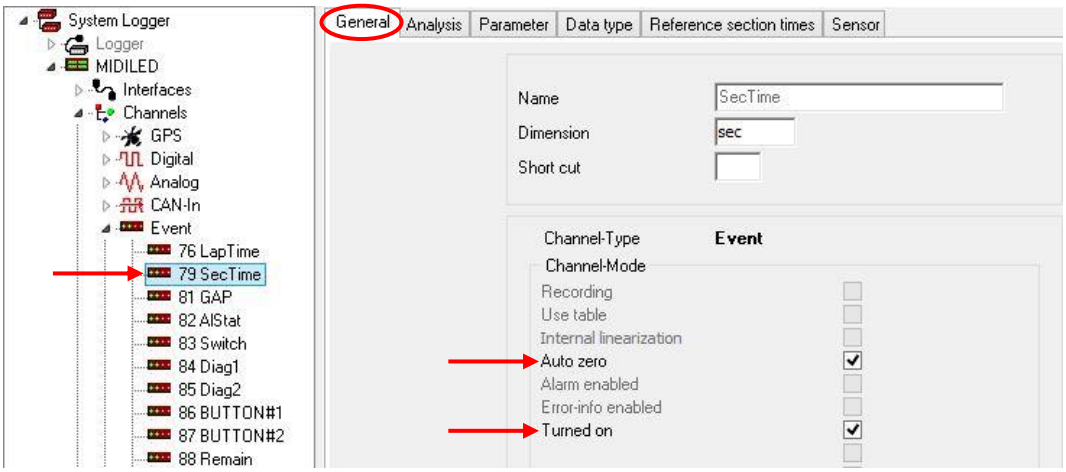
How to use the Delta Sections function

With the Delta Sections function the dashboard compares your section/lap times with previously entered values or with the section times of your best lap. It updates its value on the screen each time when you pass a trigger signal for lap or section times. It is assumed that you have already set the function of your lap time and section time channels correctly.

To enable the Delta Sections function you have to change the operation mode of your dash. Therefore you select the dash in the system tree and go to tab “Operation modes”. There you select the mode “Delta Sections”.

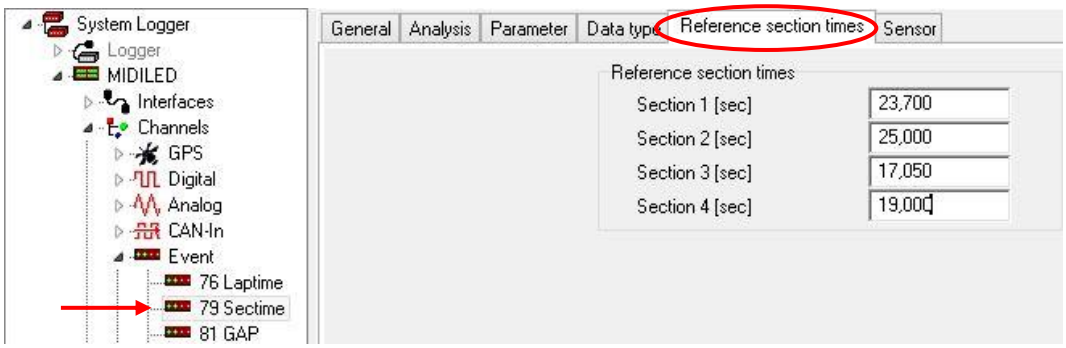


If you want the DeltaSection function to update the reference section times itself, you have to activate the “Auto zero” in tab “General” of channel Sectime. Otherwise it won´t update the reference section times itself.



Enter the section times to compare. Go to “Channels”, “Event” ⇒ “Sectime” and select tab “Reference section times”.

Even if you´re using the self-updating funcitonality you should enter some values. To update itself the lap time has to be faster than the sum of the reference section times and the BestLap[3].

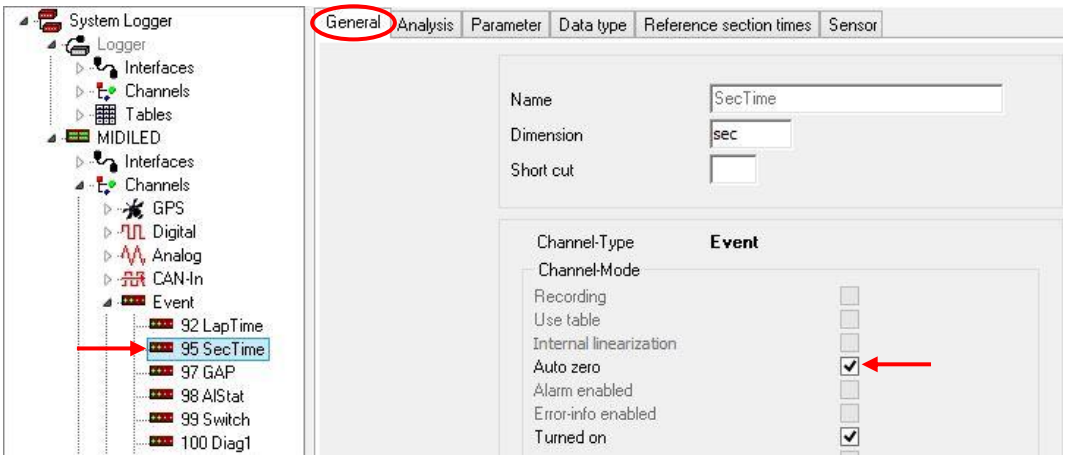


Confirm your changes with <**Apply**>.

[3] The BestLap depends on the lap which is stored inside the GAP-table (assumed channel GAP is turned on) until you go faster.

If you put this channel on an output channel, you use it without a format function. The value will be displayed in seconds. Please remember by entering the length, that the "-" for indicating a faster time is a character, too.

The Delta Sections function is able to update the previously entered values itself. Therefore the lap time has to be faster than the entered values and the lap length has to be more than the entered minimum track length of the GAP channel[4]. To enable this automatic update function select tab "General" of your Sectime channel and put a tick by "Auto zero":



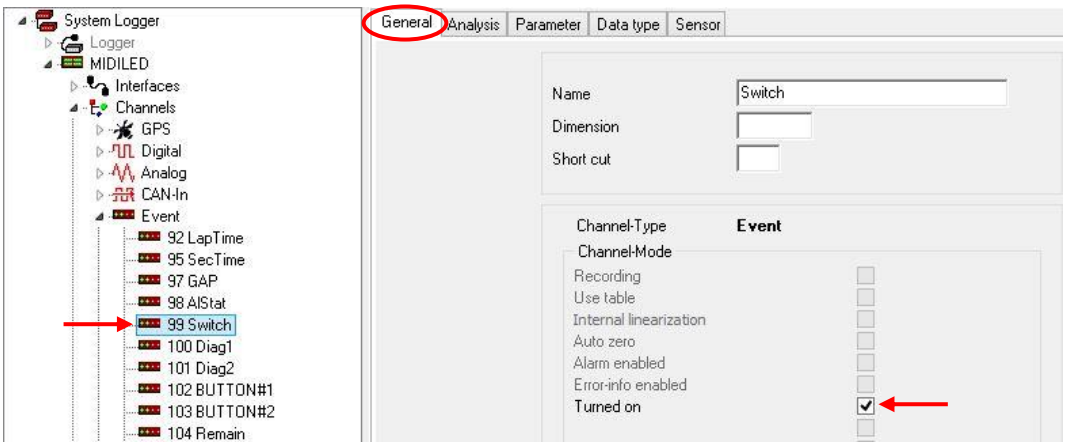
Confirm your changes with <**Apply**>.

How to program the Switch function

The switch function changes the currently displayed page automatically. There are two possibilities in this function to change the page:

1. You select a trigger channel and a page to display.
2. You select a channel with information of the page number to display.

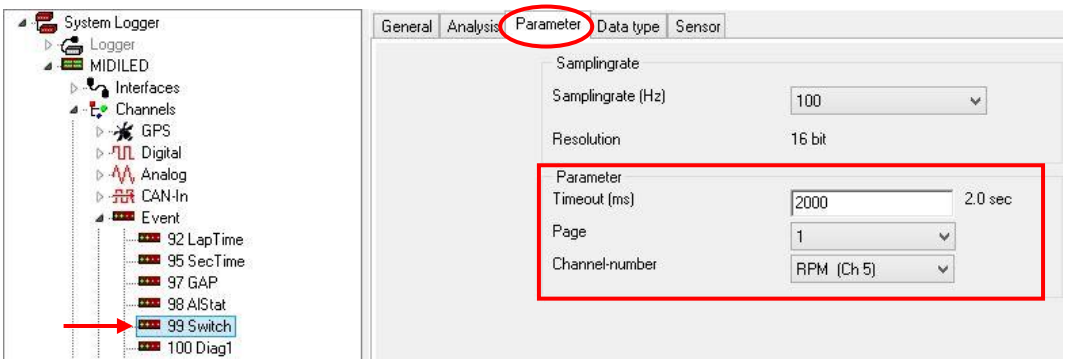
Go to "Channels", "Event" ⇒ "Switch" and select tab "General" to turn this function on.



On tab "Parameter" you can select when this function will change the page and which page will be displayed then. In field "Parameter" you select a timeout, the page and the channel which triggers the change.

[4] If the GAP channel is turned off, the minimum track length is not taken into account.

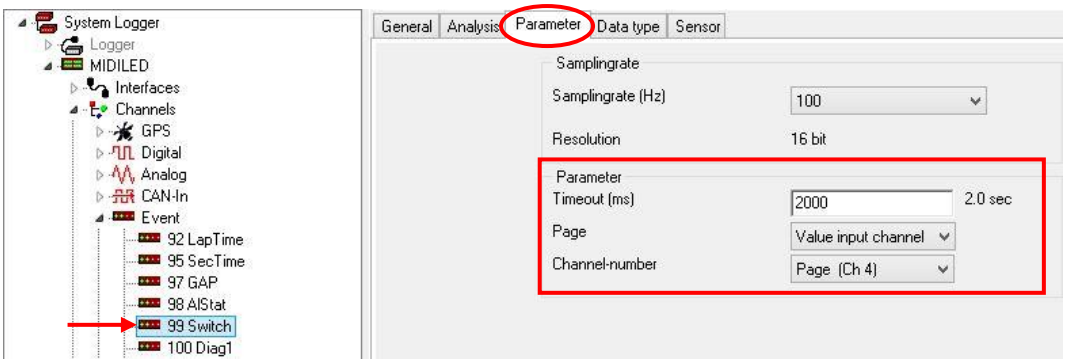
Possibility #1:



Select page and a trigger channel. As soon as (considering the timeout) the value of this channel is higher than “0” then the display will switch to the selected page.

For example: If the engine is running (RPM>0) the display will change the currently displayed page to page 1. And when the engine is turned off (RPM=0) the displayed page will switch back.

Possibility #2:



Select “Value input channel” from the drop-down list as the page and select the channel with information of the page number to display. This can be a CAN channel or a Calc-channel or any other kind of channel. But it is important that the value of this channel contains the page number.

As soon as (considering the timeout) the value of this channel changes, the display will switch to the page of the channel value.

For example: If the channel has the value “4” then page 4 will be displayed. When the channel value changes to “1”, page 1 will be displayed. If the page of the channel value is not used or if there is no page to that channel value, then the page will not be changed.

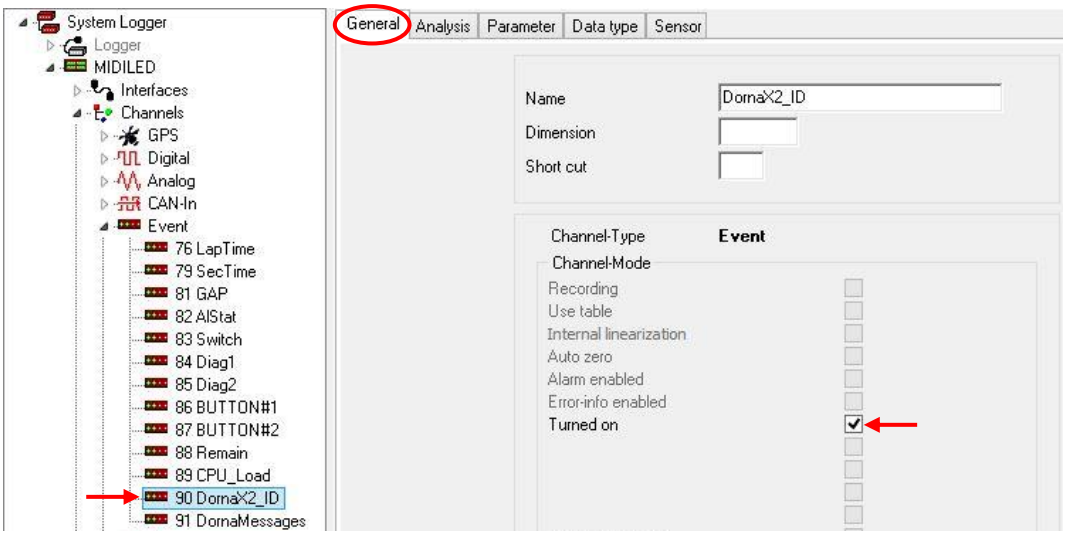
X2 settings[5]

How to check the availability of the X2 transponder

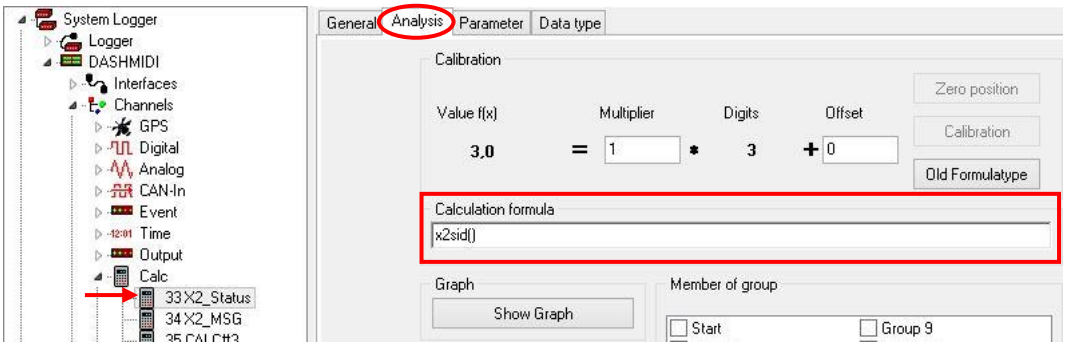
Since 2017 it is requested by the Dorna to check if the X2 transponder is connected correctly to the Datarecording system. This check happens when the system is turned on.

It takes three steps to find out and display if an X2 transponder is available:

1. Request the transponders ID by turning on the dash´s channel DornaX2_ID



2. Check the outcome of the request with a calc-channel. In this example this channel is named "X2_Status" and programmed with the calculation formula "x2sid()"

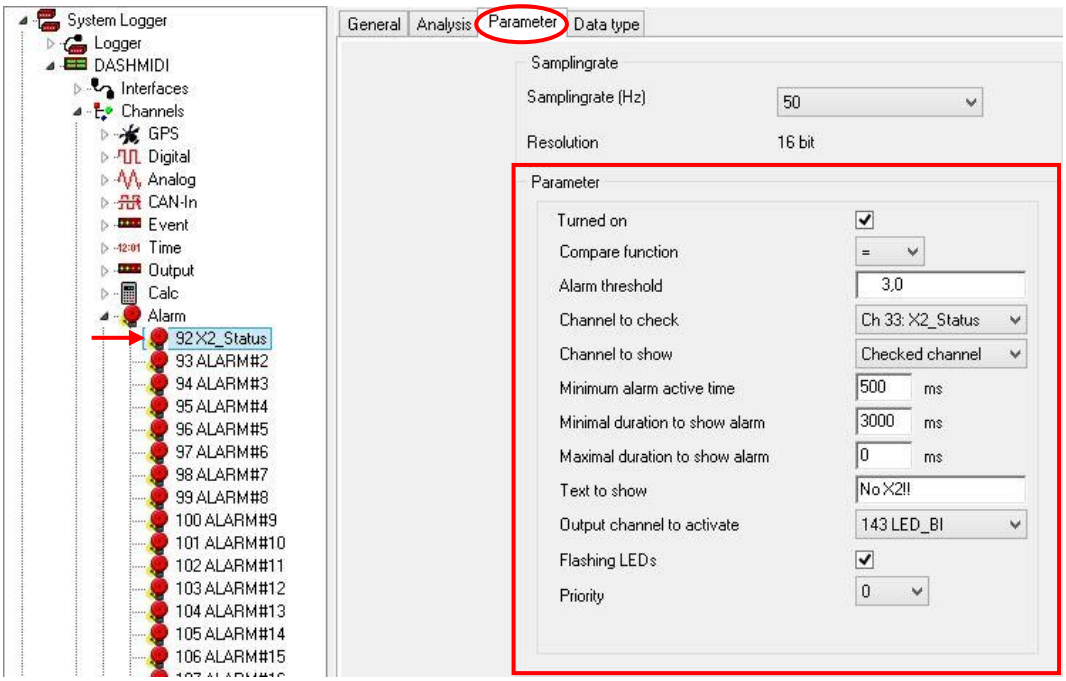


The possible results are:

0 ⇒ Channel DornaX2_ID not turned on
1 ⇒ The X2 transponder´s ID is requested
2 ⇒ The X2 transponder´s ID is received
3 ⇒ After 10s there is still no ID received ⇒ no transponder connected

[5] Since 2017, Midi/MiniDash Firmware-Version 46, BigDash Firmware-Version 120

3. To show the result on the display, you use an alarm channel:

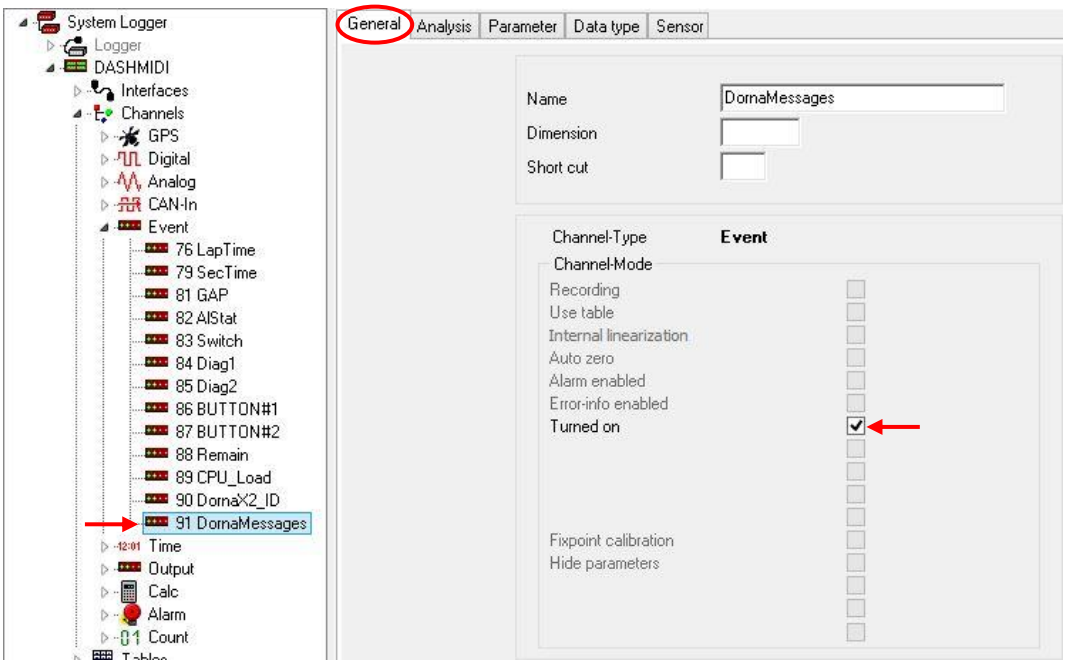


Confirm your changes with <**Apply**>.

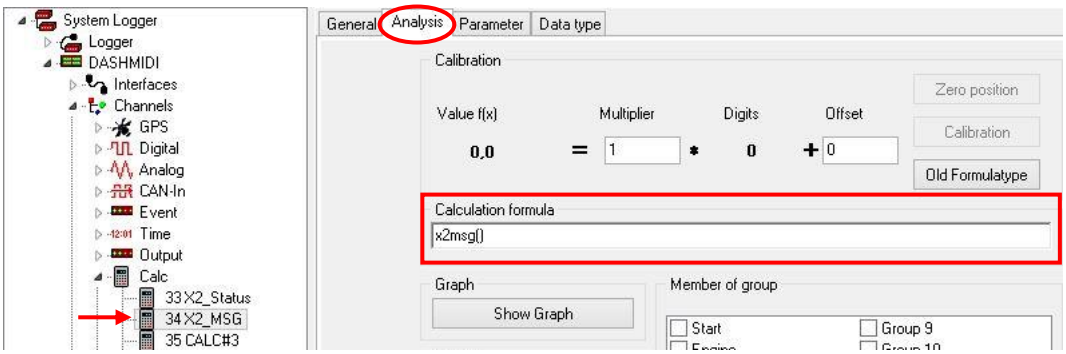
How to display flags and other X2 related messages

Since 2017 the dashes should be able to display more X2 related information than just the flags and Dorna info messages. To handle this amount of information the dash´s programming has changed.

To display the Dorna flags, info messages, box call and team notification you only have to turn on the channel DornaMessages. This channel receives all message information and displays them automatically until the signal is taken back.



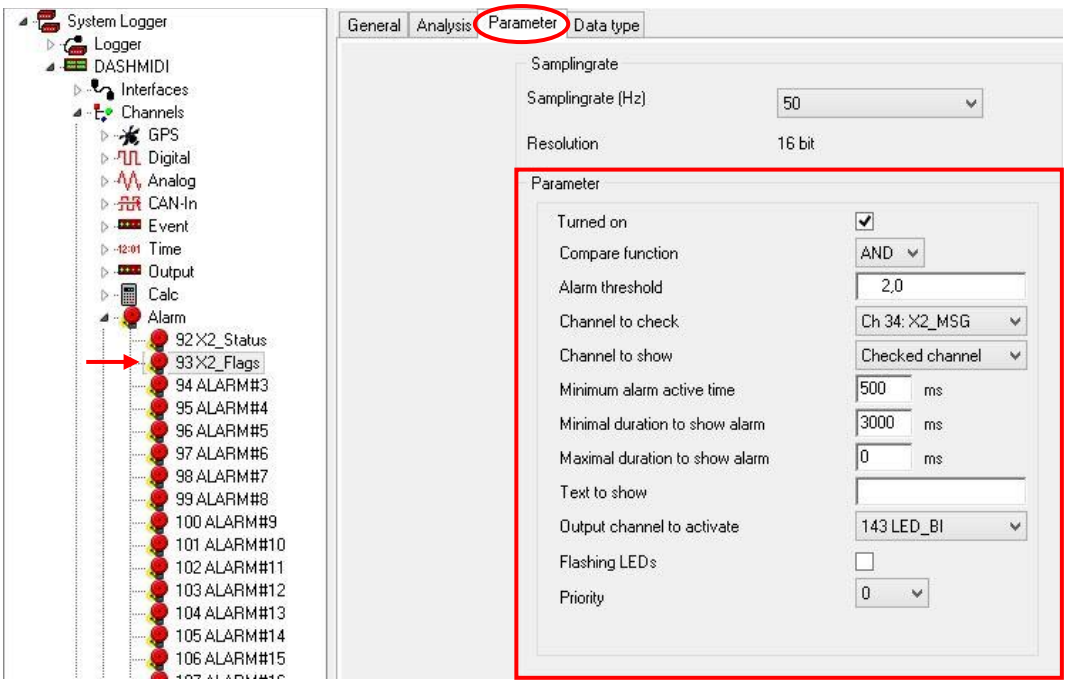
As it is mandatory to use the flags with an LED, these signals have to be separated from the others. To do this, you program a calc-channel, which indicates what kind of signal is received. In this example the channel is named “X2_MSG” and programmed with the calculation formula “x2msg()”:



The information of this channel is bit coded:

Bit 1 ⇒ value 1 ⇒ ID received
Bit 2 ⇒ value 2 ⇒ flag received
Bit 3 ⇒ value 4 ⇒ Dorna info message received
Bit 4 ⇒ value 8 ⇒ box call received
Bit 5 ⇒ value 16 ⇒ team notification received

To light up the LED you program an alarm channel. Use the calc-channel “X2_MSG” as source channel, the trigger condition “AND 2” and set the other parameters (duration to show alarm, LED for indication, priority) as usual:



Confirm your changes with <**Apply**>.

Calculation functions of the calc-channels

Evaluating calc-formulas

The formula for a channel is evaluated in a strict time sequence given by the sampling rate of that channel. E.g., if the rate is set to 200Hz then every 1/200 seconds (cycle time) the formula is evaluated once and the result value is stored.

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 the 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* (*expr1*^{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, if *expr* is positive;
		-1 if *expr* is negative;
		0, if *expr* is zero.
			`sig(-2)`
			`sig(#4)`
			`sig(#speed)`

Absolute value	`abs(`*expr*`)`	Result is the absolute value of *expr*.	`abs(-2)`
			`abs(#4)`
			`abs(#speed)`

Derivation	`der(`*expr*`)`	Result is the rate of change between the values of *expr* computed in the previous cycle and the current value. (time derivative)	`der(#4)`
			`der(#dist)`
			`der(#4-#speed)`

Sum over time	`sum(`*expr*`)`	Result is the sum of all the values of *expr*.	`sum(1)`
			`sum(#4)`
			`sum(#dist)`

Integration	`i(`*expr*`)`	Result is the integrated values of *expr* over time.	`i(#speed)` `i(#acc)`

Relational expressions

Smaller	*expr1* `<` *expr2*	Result is 1 if *expr1* is smaller than *expr2*. 0, otherwise.	`#4 < #10`
			`#4 < 10.7`
			`10 < #speed`

Smaller or equal	*expr1* `<=` *expr2*	Result is 1 if *expr1* is smaller than or equal to *expr2*. 0, otherwise.	`#4 <= #10`
			`#4 <= 10.7`
			`10 <= #speed`

Greater	*expr1* `>` *expr2*	Result is 1 if *expr1* is greater than the value of *expr2*. 0, otherwise.	`#4 > #10`
			`#4 > 10.7`
			`10 > #speed`

Greater or equal	*expr1* `>=` *expr2*	Result is 1 if *expr1* is greater than or equal to *expr2*. 0, otherwise.	`#4 >= #10`
			`#4 >= 10.7`
			`10 >= #speed`

Equality	*expr1* `==` *expr2*	Result is 1 if *expr1* is equal to *expr2*.
		0, otherwise.
			`#4 == #10`
			`#4 == 10.7`
			`10 == #speed`

Inequality	*expr1* `!=` *expr2*	Result is 1 if *expr1* is not equal to *expr2*. 0, otherwise.	`#4 != #10`
			`#4 != 10.7`
			`10 != #speed`

Logical expressions

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

Bit expressions

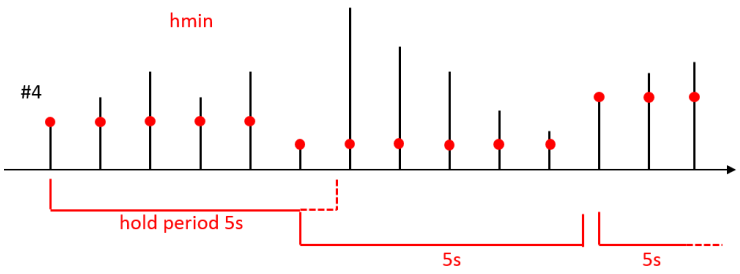
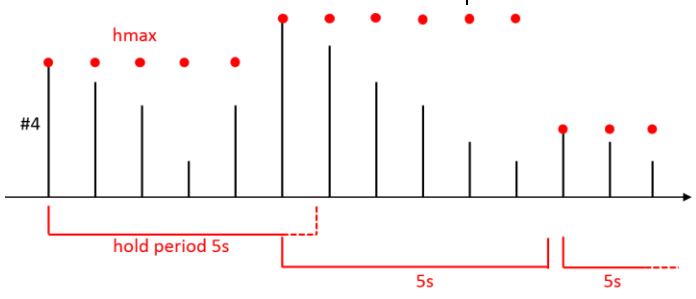
Bitwise AND	*expr1* `&` *expr2*	Result is the bitwise AND of *expr1* and *expr2*.	`#4 & 0xFF`
Bitwise OR	*expr1* `|` *expr2*	Result is the bitwise OR of *expr1* and *expr2*.	`#4 | 0x80`
Bitwise complement	`~` *expr*	Result is the bitwise complement of *expr*.	`#4 & ~0x00`

Trigonometric expressions

Rad	`rad`(*expr*)	Converts the value of *expr* from degrees to rad.	
Degree	`deg`(*expr*)	Converts the value of *expr* from rad to degrees.	
Sine	`sin`(*expr*)	Result is the sine value of *expr.* (*expr* has to be in radians).	`sin(#4)` `sin(1)`
Cosine	`cos`(*expr*)	Result is the cosine value of *expr.* (*expr* has to be in radians).	`cos(#4)` `cos(1)`
Tangent	`tan`(*expr*)	Result is the tangent value of *expr.* (*expr* has to be in radians).	`tan(#4)` `tan(1)`
Arc sine	`asin`(*expr*)	Result is the arc sine value of *expr*. (*expr* has to be in radians).	`asin(#4 / #5)`
Arc cosine	`acos`(*expr*)	Result is the arc cosine value of *expr*. (*expr* has to be in radians).	`acos(#4 / #5)`
Arc tangent	`atan`(*expr*)	Result is the arc tangent value of *expr*. (*expr* has to be in radians).	`atan(#4 / #5)`

Sine (degree)	`dsin(`*expr*`)`	Result is the sine value of *expr*. (*expr* has to be in degree).	`dsin(70)`
Cosine (degree)	`dcos(`*expr*`)`	Result is the cosine value of *expr*. (*expr* has to be in degree).	`dcos(70)`
Tangent (degree)	`dtan(`*expr*`)`	Result is the tangent value of *expr*. (*expr* has to be in degree).	`dtan(70)`
Arc sine (degree)	`dasin(`*expr*`)`	Result is the arc sine value of *expr*. (*expr* has to be in degree).	`dasin(70)`
Arc cosine (degree)	`dacos(`*expr*`)`	Result is the arc cos value of *expr*. (*expr* has to be in degree).	`dacos(70)`
Arc tangent (degree)	`datan(`*expr*`)`	Result is the arc tangent value of *expr*. (*expr* has to be in degree).	`datan(70)`

Min-Max 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	`hmin(`*expr1, expr2*`)`	*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. (see below) 	`hmin(#4, 5)`
Hold maximum	`hmax(`*expr1, expr2*`)`	*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. (see below) 	`hmax(#4, 5)`

Filter expressions

Filter	`flt(`*expr1,expr2*`)`	Computes a τ-filter over the values of *expr1* with a window size of *expr2* seconds.	`flt(#4, 5)`

Variables

m1, m2, … 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.)	

p1, p2		Variables which can only be incremented. Cannot be reset to 0.	`p1 = p1+1`
x		Holds the value from the previous computation. Each channel has its own variable x. Reset to 0 at power on.	

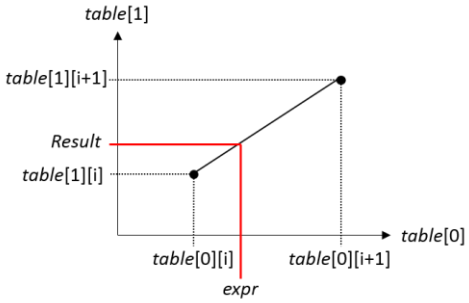
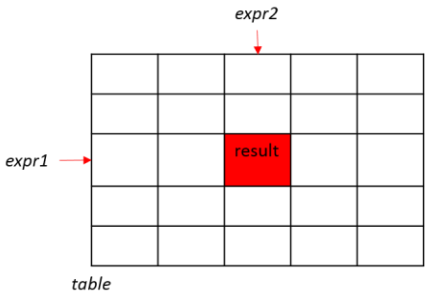
Conditional expressions

Conditional execution	`if(`*expr1*`,`*expr2*`,` *expr3*`)`	The result is *expr2* If *expr1* is not zero Otherwise, the result is *expr3*.	`if (#3==0,#4,#5)`

Assignment

Assign	*variable* = *expr*	Stores the value of *expr* in a *variable*. The result of the assignment is the value of *expr*.	`m1 = max(#3, #4)`

Table expressions

Table	`tab(`*table*`,`*expr*`)`	*table* is a table with two rows, *table*[0] and *table*[1]. The result of the tab-function is computed by linear interpolation as shown below:  	
Linearize	`lin(`*expr1*`,`*expr2*`,` *table*`)`	*table* is a 2-dimensional table. Result is the value *table*[*expr1*][*expr2*].  	

Syntax

expr := *numerical constant* or *variable or channel*

expr := *expr operator expr*

expr := *function* "(" *expr* ")"

numerical constant := positive or negative decimal number

numerical constant := hexadecimal number // "0x" or "0h" followed by hexadecimal digits 0..F

numerical constant := binary number // "0b" followed by binary digits (0 and 1)

channel := "#"<channel number> or "#"<channel name>

variable := m1 or m2 or …, or m6 or p1 or p2 or x

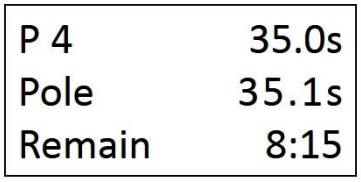
All possible operators and functions are described in the table above.

Appendix

Portrayal of X2 Team notification and Box call

The X2 transponder´s team notification and box call are a possibility to have a virtual pit board on the bike´s dash. To display the information without further programming, the settings are preprogrammed and fixed.

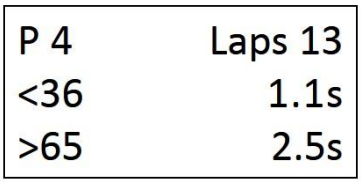
Practice Info:



Own position

Target time (only seconds)
P1 Time (only seconds)
Remaining time

Race Info:

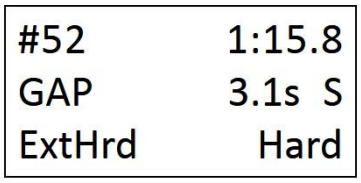


Own position — Remaining laps

Racer ahead — Gap ahead

Racer behind — Gap behind

Opponent Info:



Rider — Laptime

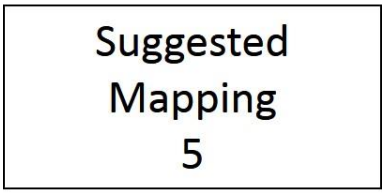
Gap — Tire option

Front tire — Rear tire

Tire options: Slick (S), Intermediate (I), Wet (W)

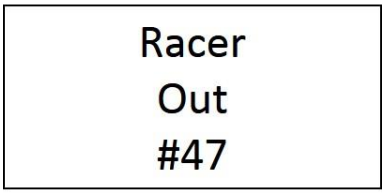
Tire type: Extra soft (ExtSof), Soft, Medium, Hard, Extra hard (ExtHrd)

Suggested mapping:

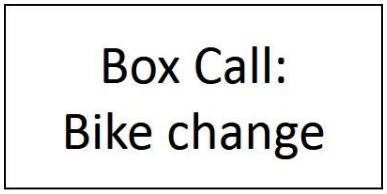


Being followed:

Racer out:



Box Call:

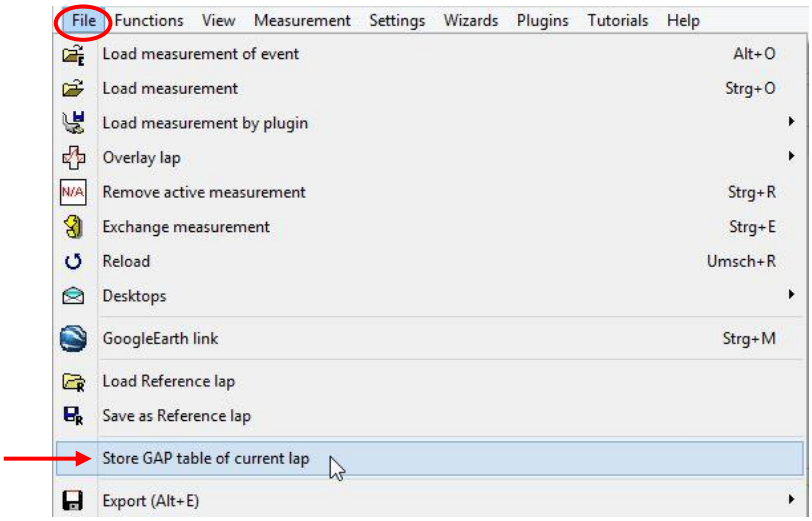


Box Call:

Call reason

How to create a GAP table with the Analyzer

1. Load a measurement in the *Analyzer*.
2. Go to a lap which you want to use for the GAP table.
3. Select “File” ⇒ “Store GAP table of current lap”.



4. Enter a name and save the new GAP table.