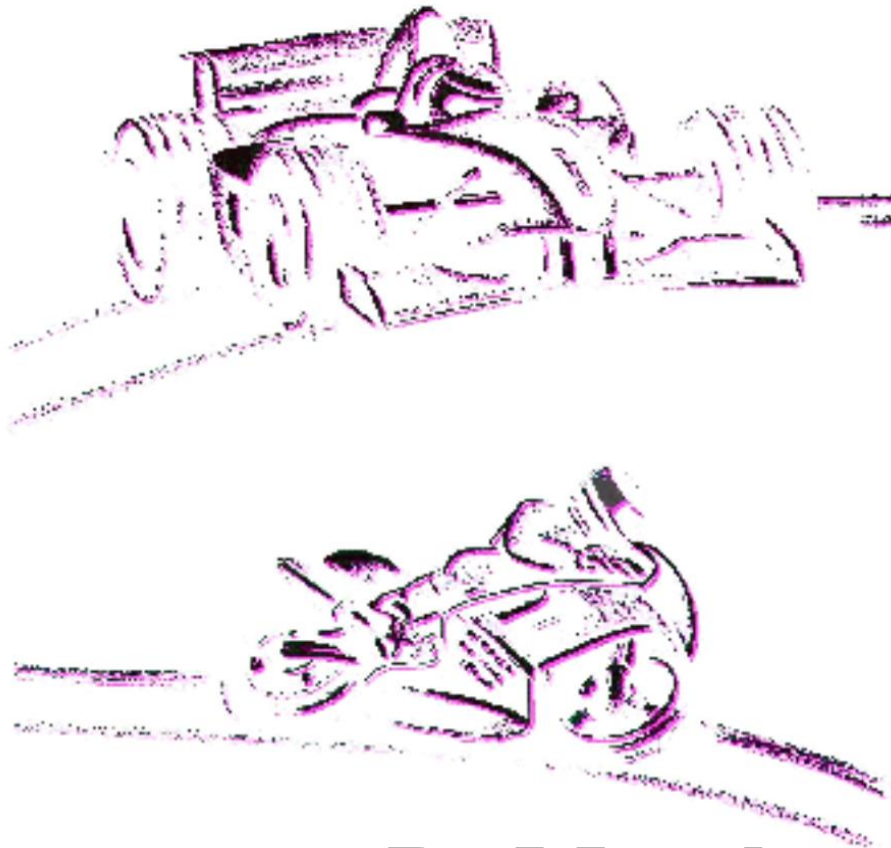


- English -



# 2D Modules

## Event channels

## 1 Revision History

| Revision | Description        | Release Date | Author |
|----------|--------------------|--------------|--------|
| 0        | Initial Release    | 2013-01-04   | MC     |
| 1        | Revision           | 2022-03-02   | FS     |
| 2        | Adding FAQ section | 2022-03-14   | FS     |

### Revision 0:

Initial release

### Revision 1:

Fundamental revision of the manual

### Revision 2:

Adding FAQ section to GAP event (4.3.3)

## **Table of content**

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>REVISION HISTORY .....</b>                                    | <b>2</b> |
| <b>2</b> | <b>NOTES AND SYMBOLS USED IN THIS MANUAL .....</b>               | <b>4</b> |
| <b>3</b> | <b>PREFACE .....</b>   | <b>5</b> |
| 3.1      | EVENT OVERVIEW .....   | 6        |
| <b>4</b> | <b>EVENTS .....</b>  | <b>7</b> |
| 4.1      | LAPTIME EVENT .....  | 7        |
| 4.2      | SECTIONTIME EVENT .....  | 9        |
| 4.2.1    | SecTime - Race Mode .....  | 10       |
| 4.2.2    | SecTime – Road mode .....  | 10       |
| 4.2.3    | SecTime - Endurance mode .....                                   | 10       |
| 4.2.4    | SecTime - Delta Sections Mode .....                              | 11       |
| 4.3      | GAP FUNCTION .....   | 12       |
| 4.3.1    | Configuration of GAP function: .....                             | 13       |
| 4.3.2    | Use cases GAP function .....                                     | 14       |
| 4.3.3    | GAP problems – FAQ .....   | 15       |
| 4.4      | COUNT .....  | 16       |
| 4.5      | SWITCHPAGE EVENT .....   | 17       |
| 4.6      | BUTTON EVENTS .....  | 20       |
| 4.7      | ALSTAT EVENT .....   | 22       |
| 4.8      | DIAG EVENTS .....  | 23       |
| 4.8.1    | Using the Diag event - Practical Example .....                   | 25       |
| 4.9      | REMAIN EVENT .....   | 28       |
| 4.10     | DORNA EVENTS .....   | 29       |
| 4.10.1   | DornaFlags .....   | 29       |
| 4.10.2   | DornaMessages .....  | 29       |
| 4.11     | LAPDETECT - SET FINISH LINE COORDINATES VIA MANUAL TRIGGER ..... | 30       |
| 4.11.1   | LapDetect .....  | 31       |
| 4.11.2   | LapDetectSpeed .....   | 31       |
| 4.11.3   | LapDetectStatus .....  | 31       |
| 4.12     | CPU LOAD .....   | 32       |
| 4.12.1   | CPU_Load_AVG .....   | 32       |
| 4.12.2   | CPU_Load_MAX .....   | 32       |
| 4.13     | EVENTTRG .....   | 33       |

## 2 Notes and symbols used in this manual



- In the paragraphs highlighted with this symbol, you will find tips and practical advice to work with the 2D-Software.



- Documentation reference to another manual



- In the paragraphs highlighted with this symbol, you will find additional information. It is very important that you follow the instructions given.

### 3 Preface

**Event Channels** offer special functions for different types of modules.

Beside **Lap-** and **Section-** timing event channels, there will be many other events which are especially related to other types of 2D modules.

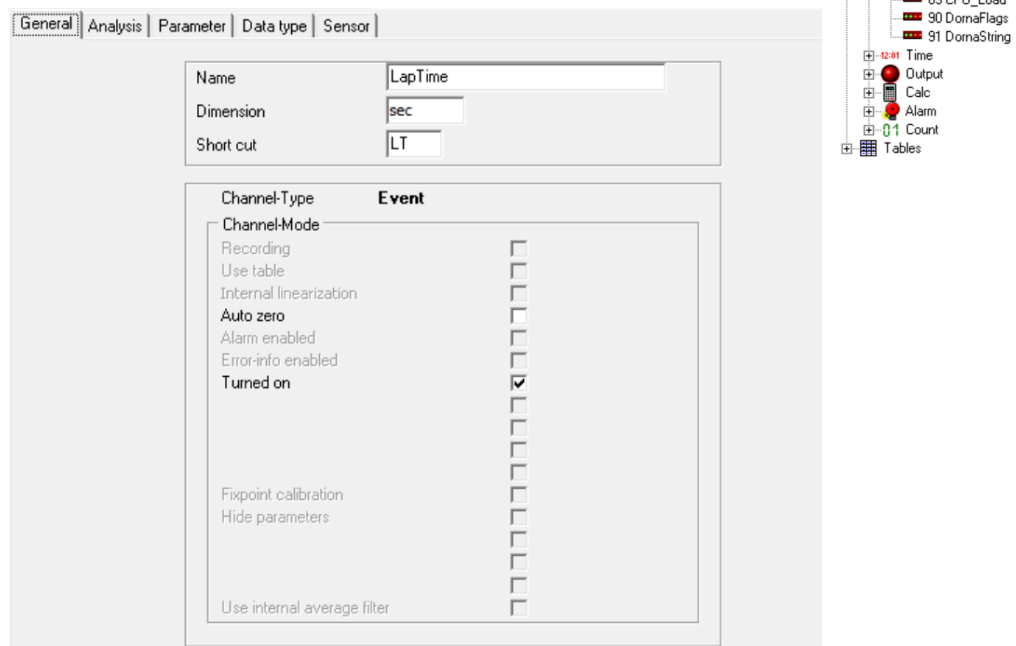
Thereby, Dashboard events like **Switch page** or Dorna events **DornaFlags** and **DornaString** are only available in 2D Dashboards.

This manual combines all descriptions of the 2D modules events.

The event channel group can be found in *Winit* when the respective module is loaded.

When an event is double clicked, the event settings are opened.

Each event is separated in different setting tabs which can be found on top of the window where the different properties of the event can be set.



**All Event-channels can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

### 3.1 Event overview

|            | Dashboards |          |         |           | Logger  |             | CAN Modules  |
|------------|------------|----------|---------|-----------|---------|-------------|--------------|
|            | MiniDash   | MidiDash | BigDash | ColorDash | LED Bar | Sticklogger | GPS/GNSS2CAN |
| Laptime    |            |          |         | ✓         |         |             |              |
| Sectime    |            |          | ✓       |           |         | (✓)         |              |
| GAP        |            |          | ✓       |           |         | ✗           |              |
| Count      |            |          |         | ✓         |         |             |              |
| AIStat     |            | ✓        |         |           |         | ✗           |              |
| SwitchPage |            |          | ✓       |           |         | ✗           |              |
| Button     |            | ✓        |         |           |         | ✗           |              |
| DIAG       |            | ✓        |         |           |         | ✗           |              |
| Remain     |            | ✓        |         |           |         | ✗           |              |
| Dorna      |            |          |         | ✓         |         |             | ✗            |
| LapDetect  |            |          |         | ✗         |         |             | ✓            |
| CPU Load   |            |          |         | ✓         |         |             |              |
| EventTrg   |            |          |         | ✓         |         |             |              |

This document is subject to change at 2D decision. 2D assumes no responsibility for any claims or damages arising out of the use of this document, or from the use of modules based on this document, including but not limited to claims or damages based on infringement of patents, copyrights or other intellectual property rights.

## 4 Events

### 4.1 Laptime event



- Laptimes can also be set in Analyzer subsequently! Please see the manual *Laptiming via Analyzer* which can be found at:  
<http://2d-datarecording.com/downloads/manuals/>

The Laptime event is used to generate laptimes inside a module, which than can be recorded, displayed, or send to other modules via CAN.



- Loggers, Dashboards and GPS2CAN-modules are able to create laptimes!
- It is important that laptimes should always be calculated in just **one module of a system only** and, if available, is only used in other modules!

In general, the source which is used for creating a laptime must be distinguished in the following two different ways:

1. **GPS-positions** Defining finish line by GPS-coordinates
2. **Transponder X2-messages** Defining finish line by received X2-messages



- Detailed instructions on how to generate laptimes via GPS coordinates and TransponderX2-messages can be found at:  
<http://2d-datarecording.com/downloads/manuals/>

Depending on the behaviour of the trigger channel, what basically means the distinction between calculating and displaying laptime, four options exist for using the Laptime-Event in the correct way.



- Some of these options require the entry of a threshold value (digits).

Laptime is calculated, if...

- value smaller than** ... trigger channel becomes lower than threshold
- value bigger than** ... trigger channel becomes higher than threshold
- value changes** ... trigger channel value changes (no threshold entry)

Laptime is displayed, if...

- value @ change** ... trigger channel value changes (no threshold entry)



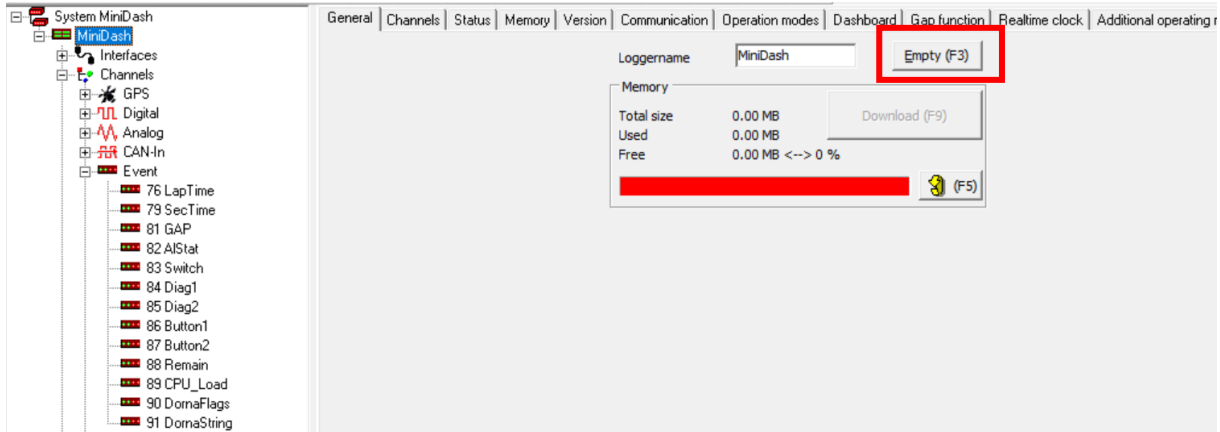
- The parameter **value @ change** must be used if a module receives the laptime via CAN-bus from another module and just needs to display the received laptime
- This procedure is important, because only then the laptime shown in the display will match the laptime calculated and recorded in another module of the system!

**The Laptime-Event-channel can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

This document is subject to change at 2D decision. 2D assumes no responsibility for any claims or damages arising out of the use of this document, or from the use of modules based on this document, including but not limited to claims or damages based on infringement of patents, copyrights or other intellectual property rights.

### Manually erase the fastest lapttime in the 2D display:

- Select the respective 2D Dash from the system tree
- Select the tab **<General>**
- Click the **<Empty>** button, and then click **<Apply>**.

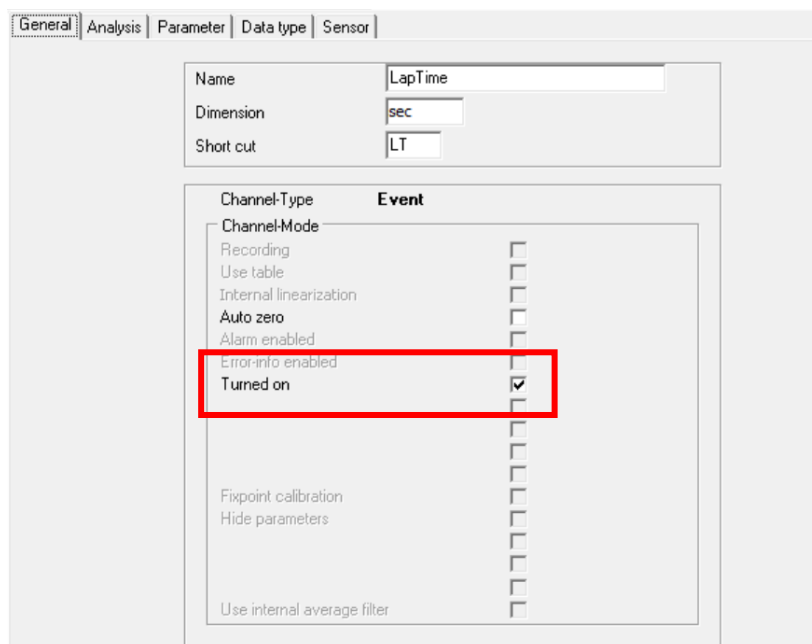


### Automatically erase the fastest lapttime in the 2D display:

Every time the display is switched on, the fastest lapttime is deleted from 2D dash's memory.

To automatically reset the Lapttime memory when the 2D Dash is powered-on:

- Select the Event channel Lapttime via **2D-Dash → Event → Lapttime**
- Select the tab **<General>**
- Check the **<Auto zero>** checkbox and then click **<Apply>**





## 4.2 Sectiontime event



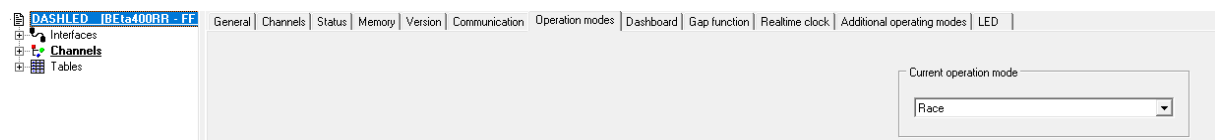
- Section times can also be set in Analyzer subsequently! Please see the manual *Laptiming via Analyzer* which can be found at: <http://2d-datarecording.com/downloads/manuals/>

The Sectime event is used to generate section times inside a module, which than can be recorded, displayed, or send to other modules via CAN.



- Loggers, Dashboards and GPS2CAN-modules are able to create section times!
- It is important that section times should always be calculated in just **one module of a system only** and, if available, is only used in other modules!

In general, the SecTime event can be used in four different ways, which are defined at the 2D-modules tab *Operation modes*.



The different operation modes are:

- Race mode:  
→ elapsed time **between** two sec trigger signals
- Road mode:  
→ same as Race mode, but warnings cannot be accepted by pressing the display buttons
- Delta Section mode:  
→ gained or lost time **in the last completed section** to a previously defined reference section time
- Endurance mode:  
→ count down to 0 (from reference value) when a section trigger signal is received



- At Loggers no SecTime-mode can be chosen. Loggers always work in Race mode!

**The Sectiontime-Event-channel can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

#### 4.2.1 SecTime - Race Mode

In **Race Mode**, the event-channel *SecTime* shows the elapsed time **between** two sec trigger signals.



- *Timeout* value refers to time before the next section trigger is accepted and not to the Sectime value itself!

The parameters (Source channel, Timeout) of section time event must be set correctly!



- Detailed instructions on how to generate section times via GPS coordinates and TransponderX2-messages can be found at:  
<http://2d-datarecording.com/downloads/manuals/>

Counter channel Sec is not reset to 0 at laptrigger.

#### 4.2.2 SecTime – Road mode

Same as Race mode, but warnings cannot be accepted by pressing the display buttons.



- **If a warning occurs it will be shown as long as it persists!**

#### 4.2.3 SecTime - Endurance mode

In **Endurance Mode**, the event-channel *SecTime* is used as count down to 0 (from reference value) when a section trigger signal is received!



- *Timeout* value is used as start value of countdown
- This is useful where a minimum pitlane time is enforced by regulation. GT3 Endurance teams use this function to show the time left before the vehicle can leave the pits after a pitstop.

The parameters (Source channel, Timeout) of section time event must be set correctly!



- Detailed instructions on how to generate section times via GPS coordinates and TransponderX2-messages can be found at:  
<http://2d-datarecording.com/downloads/manuals/>

#### 4.2.4 SecTime - Delta Sections Mode

In **Delta Sections Mode**, the event-channel *SecTime* is used to display gained or lost time in the last completed section to a previously defined section time.



- *Timeout* value refers to time before the next section trigger is accepted and not to the *SecTime* value itself!



- After first section, **SecTime in Delta Section mode also contains the win/loss from previous sections!**
- If negative values at Delta Mode are expected, do not use **Laptime format (MM:SS.HHH)** for displaying *SecTime* because **Laptime format is not able to display negative times.**

The parameters (Source channel, Timeout) of section time event must be set correctly!



- Detailed instructions on how to generate section times via GPS coordinates and TransponderX2-messages can be found at:  
<http://2d-datarecording.com/downloads/manuals/>

In the tab *Reference section times* the target laptime for completion of each section can be defined for gain or lost time calculation.

| General   | Analysis | Parameter | Data type | Reference section times | Sensor          |        |                 |        |                 |        |                 |        |  |
|---|----------|-----------|-----------|-------------------------|-----------------|--------|-----------------|--------|-----------------|--------|-----------------|--------|--|
| <div>Reference section times</div> <table> <tr> <td>Section 1 [sec]</td> <td>10.000</td> </tr> <tr> <td>Section 2 [sec]</td> <td>15.000</td> </tr> <tr> <td>Section 3 [sec]</td> <td>18.000</td> </tr> <tr> <td>Section 4 [sec]</td> <td>20.000</td> </tr> </table> |          |           |           |                         | Section 1 [sec] | 10.000 | Section 2 [sec] | 15.000 | Section 3 [sec] | 18.000 | Section 4 [sec] | 20.000 |  |
| Section 1 [sec]   | 10.000   |           |           |                         |                 |        |                 |        |                 |        |                 |        |  |
| Section 2 [sec]   | 15.000   |           |           |                         |                 |        |                 |        |                 |        |                 |        |  |
| Section 3 [sec]   | 18.000   |           |           |                         |                 |        |                 |        |                 |        |                 |        |  |
| Section 4 [sec]   | 20.000   |           |           |                         |                 |        |                 |        |                 |        |                 |        |  |

#### Example:

Real Sectime of Section 3: 000:17.524 min  
 Reference Section time 3: 000:18.000 min  
 SecTime event: -000:00.476 min



- Only four reference section times can be defined!
- If *reference section times* 0 is entered for all four sections, the current laptime (LapRun) at section trigger is displayed at *SecTime* event.

Counter channel Sec is reset to 0 at laptrigger.

### 4.3 Gap function



- This configuration of GAP time function is valid when generating laptimes via GPS or TransponderX2!



- Please see the demonstration video on Youtube!  
<https://youtu.be/IYrxAfMdK5Y>

The GAP time function is used to constantly indicate the **current** time difference to the **same position** in a reference lap.

A **negative** GAP time means that at the current position the driver is ahead the current position at reference lap, so the laptime is **faster**.

A **positive** GAP time means that at the current position the driver is behind the current position at reference lap, so the laptime is **slower**.



- The GAP time is updated every 32 meters of a lap.
- If the current lap is longer than the reference lap used, the GAP time value from the last available 32-meter reference section is hold until next laptrigger.

Reference lap can be either created by the dashboard itself or created in *Analyzer* from a measurement in post-processing.

The reference lap can be defined in three different ways:

- Creating new reference lap when faster laptime is driven (Dashboard)
- Loading reference lap created from another dashboard (Dashboard)
- Creating reference lap in Analyzer in post-processing (Analyzer → File → Store Gap Table of current lap)

The idea GAP function is based on the creation of a GAP table (*GAP.tbl*) which contains the times after the each 32-meter section.

This table can be loaded from the dashboard to save it or exchange it between different dashboards and thus different bikes.

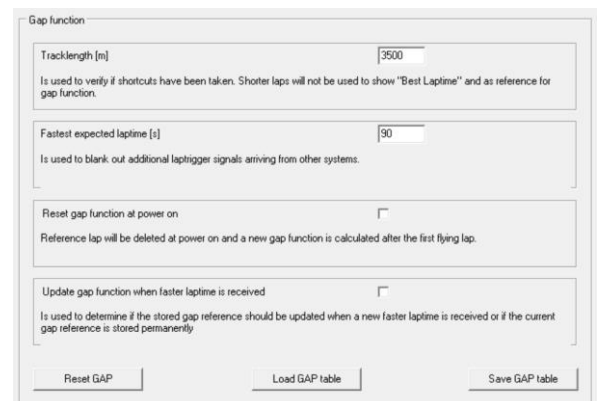


- By adding GAP time and current laptime in modules CALC-channels, an estimated laptime can be created!

#### 4.3.1 Configuration of GAP function:



- It is assumed that the laptime configuration is already done correctly!
- Select the group **<Count>** from system tree and select channel **<LapMeter>**
  - o In the tab **<General>**, select the **<Turned on>** check-box
  - o In the tab **<Parameter>**, select the **<Counted channel>** dropdown box and choose a valid and accurate speed channel. This is used to calculate lap progression for time comparison.
- Select the channel **<Gap function>** from the system tree
  - o In the tab **<General>**, select the **<Turned on>** check-box
- Select respective dashboard from system tree and navigate to tab **<GAP function>**
  - o **Tracklength [m]**  
Timeout of the Gap channel in respect to distance:  
 $Timeout\ of\ channel\ Gap = Tracklength - 10\%$
  - o **Fastest expected laptime [s]**  
Timeout of the LapTime channel in respect to Laptime:  
 $Timeout\ of\ channel\ LapTime = fastest\ expected\ Laptime - 10\%$
  - o **Reset gap function at power on**  
Here it can be decided if the GAP function is reseted each time the dash is powered on and if **<Apply>** is pressed after a setting change.  
This also resets the fastest laptime at power on!
  - o **Update gap function when faster laptime is received**  
The currently saved GAP table of reference lap is overwritten when driving a faster lap to again use fastest lap as reference.
  - o **Reset GAP**  
Unloads the GAP table.  
Unload only effective after pressing **<Apply>**
  - o **Load GAP table**  
Loads a GAP table from PC.
  - o **Save GAP table**  
Saves the current GAP table to PC.



- Press **<Apply>** to save settings on device

**The GAP-Event-channel can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

Beside the GAP live functionality, it is also possible to create and analyse the laptimes in post-processing in Analyzer.



- If very small values are used for *Tracklength* and *Fastest expected laptimes* a shortcut on track will create faster lap and thus new GAP reference time! It must be taken care by user that correct values are used!

#### 4.3.2 Use cases GAP function

##### Creating new reference lap at every outing

Reference lap is reseted at each power-on and will be updated if a faster laptime than currently stored reference lap is received.

|  |                                     |
|--|-------------------------------------|
| Reset gap function at power on   | <input checked="" type="checkbox"/> |
| Reference lap will be deleted at power on and a new gap function is calculated after the first flying lap.   |                                     |
| Update gap function when faster laptime is received  | <input checked="" type="checkbox"/> |
| Is used to determine if the stored gap reference should be updated when a new faster laptime is received or if the current gap reference is stored permanently |                                     |

##### Using a reference lap and updating if faster laptime is received

One reference lap is used and will be updated if a faster laptime than currently stored reference lap is received.

This is very helpful when the reference lap is used by another event or by another rider for training purposes.

|  |                                     |
|--|-------------------------------------|
| Reset gap function at power on   | <input type="checkbox"/>            |
| Reference lap will be deleted at power on and a new gap function is calculated after the first flying lap.   |                                     |
| Update gap function when faster laptime is received  | <input checked="" type="checkbox"/> |
| Is used to determine if the stored gap reference should be updated when a new faster laptime is received or if the current gap reference is stored permanently |                                     |

##### Reference lap only

One reference lap is used and will **not** be updated if a faster laptime than currently stored reference lap is received.

This is very helpful when the reference lap is used by another event or by another rider for training purposes for continuous riding.

|  |                          |
|--|--------------------------|
| Reset gap function at power on   | <input type="checkbox"/> |
| Reference lap will be deleted at power on and a new gap function is calculated after the first flying lap.   |                          |
| Update gap function when faster laptime is received  | <input type="checkbox"/> |
| Is used to determine if the stored gap reference should be updated when a new faster laptime is received or if the current gap reference is stored permanently |                          |

### 4.3.3 GAP problems – FAQ

#### GAP event is consistently showing value 0

- ➔ Check, if GAP event is switched on
- ➔ Check if Tracklength-TimeOut is set correctly
- ➔ Check if Fastest-expected-Laptime-TimeOut is set correctly
- ➔ Check if speed channel used in Dashboards LapMeter-Count channel is received correctly
- ➔ Check if speed channel is set correctly in Dashboards LapMeter-Count channel

#### GAP event shows non-usable values

- ➔ Check if speed channel used in Dashboards LapMeter-Count channel is received correctly
- ➔ Check if speed channel is set correctly in Dashboards LapMeter-Count channel
- ➔ Check if GAP event reset and update settings are set correctly
- ➔ Reset GAP table and try again (GAP event can also tested by using modules CALC channels)

## 4.4 Count

Because the count channels are related to the LapTime and SecTime events they are also described in Events channels manual.

**The Count-channels can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

**LapMeter:** Integration of speed channel, which results in the up-counting meters per lap and reset to 0 at LapTrigger.

**Laps:** Lap-count channel

**SecMeter:** Integration of speed channel, which results in the up-counting meters per section and reset to 0 at SecTrigger.

**Secs:** Section-count channel



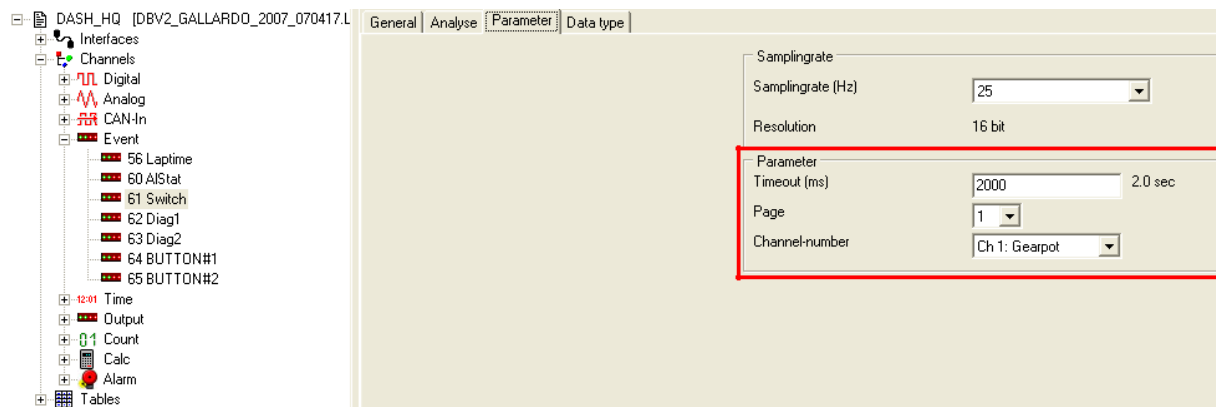
## 4.5 SwitchPage event

With the *SwitchPage* event, the displayed page can be changed in various ways with the aid of a query whether a selected "trigger channel" is greater than zero.

The various ways are described in this chapter later.



- *Button* event always has priority over the *SwitchPage* event!
- *SwitchPage* event is often used to display defined pages when the bike is inside the box



This function can be found inside the system tree via; **<Channels> → <Event> → <Switch>**.

To turn on this function, open tab **<General>**, then select the **<Turned on>** check-box.

To define the triggering channel, open the tab **<Parameter>**, select the correct trigger channel from the **<Channel-number>** dropdown-box.

After the trigger channel returns to 0, the "switched page" will remain displayed until the **<Timeout>** (set the time in milliseconds) that has elapsed.

The frequency at which the triggering channel is checked for changes is defined by the **<Samplingrate>**.

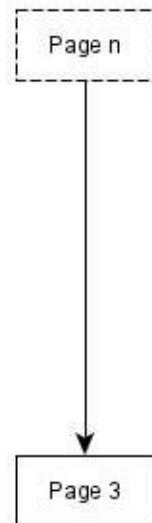
Inside the **<Page>** dropdown box, the page to which the dash will switch to is assigned. If one of the pages 1-5 are selected here, the dashboard displays the selected page when "trigger channel" is greater than zero. At dropdown box **<Page>**, also the parameter *Value input channel* can be selected, what means, that the page number can be changed dynamically by using the value of the input channel, e.g. over a Calculation channel to jump between only 2 of 4 pages.

### Example 1:

Switching to a defined page when V\_Front is bigger than 0 and should be displayed minimum 2 seconds.



- Page n corresponds to the currently displayed page



When V\_Front is greater than zero, page 3 is displayed at minimum for 2 seconds.

### SwitchPage event:

In SwitchPage event, channel *V\_Front* is used at *Channel-number* at dropdown box with 3 at Page dropdonw menu and Timeout *2000 ms*.

|                |   |         |
|----------------|---|---------|
| Parameter      |   |         |
| Timeout (ms)   | <input type="text" value="2000"/>             | 2.0 sec |
| Page           | <input type="text" value="3"/>                | ▼       |
| Channel-number | <input type="text" value="V_Front (Ch 133)"/> |         |



- Useful trigger channels: RPM, Speed

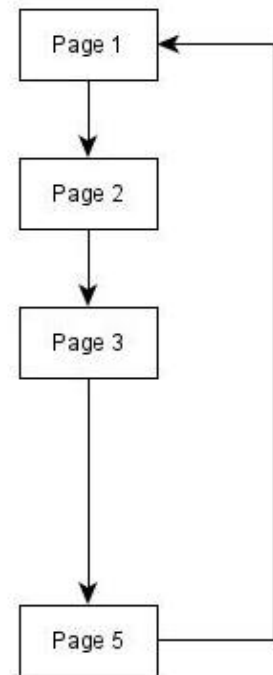
### Example 2:

Continuously scrolling through certain pages by using calculation channel

When “trigger channel” is greater than zero, the page depends on the value of the input channel. Page 4 must be hidden by using calculation channels.



- When calculation channel is used for switching pages, different switching times can be realized!



### Calculation channels

Hiding page 4

|                                     |                                     |                    |   |   |       |
|-------------------------------------|-------------------------------------|--------------------|---|---|-------|
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | cnt                | if(x<65535, x+1,0)                            | 1 | 1.000 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | ScrollPage         | (#cnt%5) + 1                                  | 1 | 1.000 |
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | ScrollPage_Trigger | if(#ScrollPage==4, #ScrollPage+1,#ScrollPage) | 1 | 1.000 |

Cnt 1 Hz counter from 0 to 65535 Changing page every second

ScrollPage 1 Hz counter from 1 to 5 Pages 1 to 5

ScrollPage\_Trigger Skip 4 at counter 1 to 5 Hide page 4

### SwitchPage event

In *SwitchPage* event, channel *ScrollPage\_Trigger* is used at *Channel-number* at dropdown box with *Value input channel* at Page dropdown menu.

Parameter

Timeout (ms)

0

0.0 sec

Page

Value input channel

Channel-number

ScrollPage\_Trigger (Ch 146)



- A further possibility to switch pages can be found in following chapter *Button events!*
- Detailed instructions on how to use *Online Calculation channels* can be found at: <http://2d-datarecording.com/downloads/manuals/>

#### 4.6 Button events

Some 2D dashboards provide built-in button(s) to scroll through the pages of the display or confirming *Alarms* to get back to normal page display.



- *Button* event always has priority over the *Switch* event!



More detailed information about *Alarms* can be found in the Dashboard manual which can be found on our website:

<http://2d-datarecording.com/downloads/manuals/>

The values of the button(s) can be checked in *Button* events.



- If dynamical switching of pages or jumping to defined pages is required, please see *SwitchPage* event at chapter 4.5.

In the tab **<Parameter>** next to **<Channel-number>** the channel to trigger the page switching is assigned.

Any Analogue or CAN-channel can be used and will change the displayed page when its physical value rises above 50% of its maximum (full scale) value.

**<Timeout>** defines the threshold time before the display page is switched.



- The default trigger channel is the input channel of the external hardware "Button".
- Analogue or CAN-IN channels can be used as input channels only!

**The Button-Event-channel can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

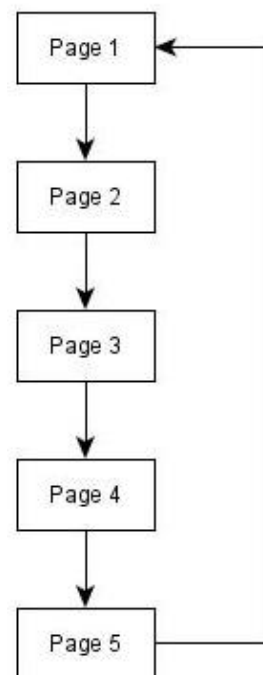
### Example:

Continuously scrolling through pages by using calculation channel

Like mentioned before, the button event scrolls through the pages when the physical value of **<Channel-number>** rises above 50% of its maximum (full scale) value.

Instead of using the respective analogue channel, which is connected to hardware button on Dashboard, other channels, like calculation channels can be used for scrolling.

This makes it possible to use a calculation channel that can be set at any time intervals, increasing its value above 50% of its maximum value (full scale) to force scrolling to next page.



### Calculation channels

Dashboard calculation channels which scroll pages every 5 seconds (%5).

|                          |                                     |                    |                       |   |       |
|--------------------------|-------------------------------------|--------------------|-----------------------|---|-------|
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | cnt                | if(x<65535, x+1, 0)   | 1 | 1.000 |
| <input type="checkbox"/> | <input checked="" type="checkbox"/> | ScrollPage_Trigger | if((#cnt%5)==0, 1, 0) | 1 | 1.000 |

*Cnt* 1 Hz counter from 0 to 65535

*ScrollPage\_Trigger* Trigger signal every 5 sec Change page every 5 sec

### Button event

In *Button* event, channel *ScrollPage\_Trigger* is used at *Channel-number* at dropdown box.

Parameter  
 Timeout (ms)  0.0 sec  
 Channel-number



- Detailed instructions on how to use Online Calculation channels can be found at: <http://2d-datarecording.com/downloads/manuals/>

#### 4.7 AStat event

The AStat event contains binary information to indicate when each of the 16 alarm channels was triggered.

Since alarms event is only available in Dashboards, the **AStat** event should be sent from the dashboard via CAN to the logger if necessary, so that it can be traced later which alarms were active and when.

This can provide information about whether the alarms were active and whether the driver overlooked any.



- If an alarm is triggered the corresponding bit is set to 1 and back to 0 when the alarm is turned off.

The AStat event does **not** contain any further information for each alarm as to whether the alarm was terminated and if so, how (timeout, button press).

If detailed information is required for a specific alarm (how the alarm was cancelled, activated text on display, flashing LED activated, etc.), this alarm must be recorded (in a logger).



More detailed information can be found in the Dashboard manual in the chapter *Alarms*:

<http://2d-datarecording.com/downloads/manuals/>

**The AStat-Event-channel can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

#### 4.8 Diag events

The diagnostic channels make it possible to display predefined messages on the dashboard when the assigned channel has a certain value.



- The displayed messages are predefined inside a table, which is then uploaded to the dashboard and saved inside.

##### Examples for using of Diag-functions:

- Display the exact meaning of a “fault channel” signal from ECU
- Count down the final laps of the race with text on screen
- Give a text-based multi-stage alert according to the value of a particular channel, e.g. “rear tyre COLD”, “rear tyre OK”, “rear tyre HOT”

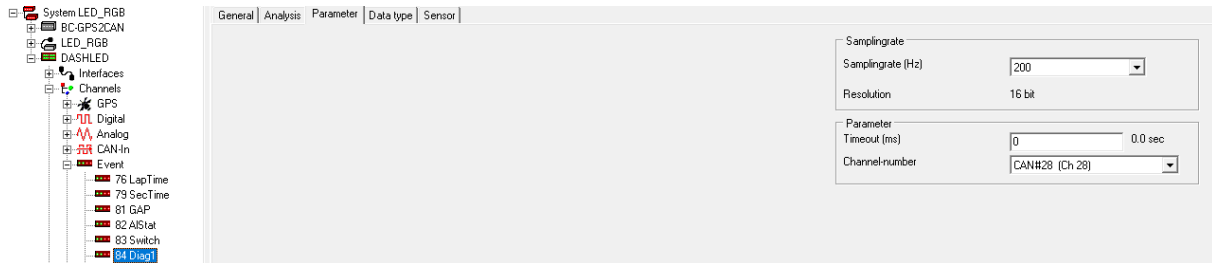
##### Differentiation of Diag-functions:

**Diag1** Displays the messages defined by the loaded string table “STR1” **on all pages**

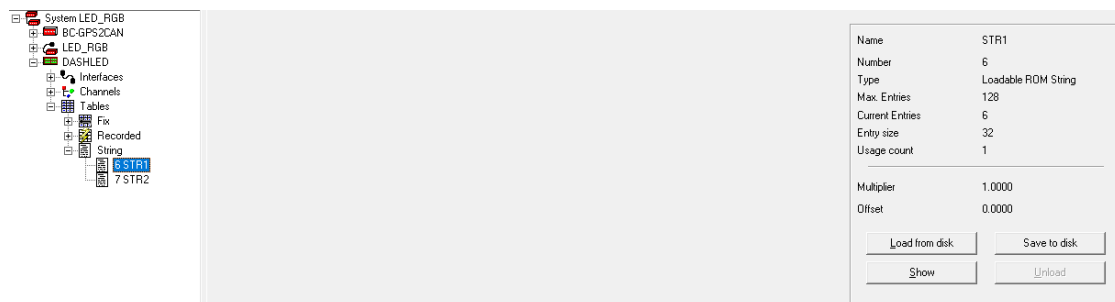
**Diag2** Displays the messages defined by the loaded string table “STR2” but **only on engineering page**

### Activate the "Diag" function:

- Open the tab **<General>** for the Diag channel, then select the **<Turned on>** check-box
- Select the source channel for the Diag event from the **<Channel-number>** dropdown-box. The selected channel will be used in collaboration with the loaded string table to generate the correct display message on the screen.



- Assign an appropriate value to **<Timeout>**, this defines the threshold time before the diagnostic output is displayed
- Define a String (STR) lookup table that defines the messages to be displayed on the screen when the corresponding source channel value is input to the string table.



- Load the String lookup table to table location **<STR1>** or **<STR2>** of the 2DDash.



- STR1 → Diag1, STR2 → Diag2



#### 4.8.1 Using the Diag event - Practical Example

Consider the following example where a multi-stage alert is created to give information on water temperature channel "T\_Water" as the vehicle is being warmed up. The messages should only displayed on engineer page (last page) of the Dash. Therefore, the Event Channel "**Diag2**" must be used to display the messages only engineering page.

In practice, the first step is to define the channel conversion table. At this stage *threshold values* must determine at which the display messages should be activated.

The following information should be displayed depending on the current water temperature:

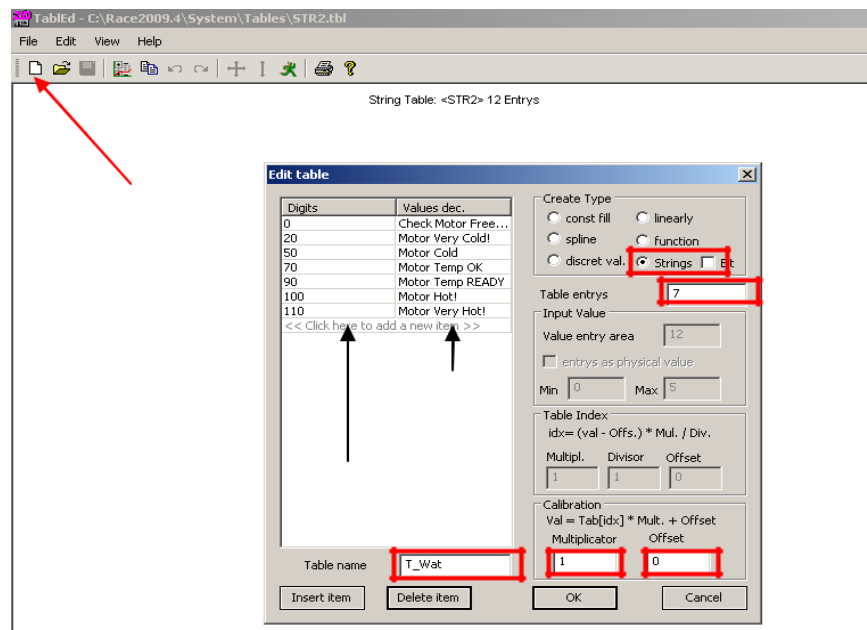
| Water temperature [C°] | Message to display  |
|------------------------|---------------------|
| -40 to 0               | Check Motor Freeze! |
| 0 to 50                | Motor V Cold!       |
| 50 to 70               | Motor Cold          |
| 70 to 90               | Motor OK            |
| 90 - 100               | Motor Ready         |
| 100 - 110              | Motor Hot!          |
| 110 - ____             | Motor Very Hot!     |

In the following it will be explained how to transfer the defined channel conversion table to a String-table:

- Open the 2D WinIt program "TabEd.exe" which can be found in the Race\_xx.y (or similar) installation folder
- Click <New> to open the "Create Table" Window
- Set Create Type to "Strings"
- Set Table entries to 7 (the number of entries you have made)
- Set "Multiplier" to 1 and "offset" equal to 0
- Name the table in the Table Name field e.g. 'T\_Wat', then click <OK>
- **In the column "Digits" enter the threshold temperature values above which you want a particular display message to be made**
- **In the column "Values dec", enter the text you want to display on the dash when the Diag input channel has the value entered to the digits column**

To achieve the desired settings that were already introduced, the entries below (in GOLD) should be entered into the table.

| Water temperature [C°] | Message to display  | Digits |
|------------------------|---------------------|--------|
| -40 to 0               | Check Motor Freeze! | 0      |
| 0 to 50                | Motor V Cold!       | 20     |
| 50 to 70               | Motor Cold          | 40     |
| 70 to 90               | Motor OK            | 50     |
| 90 - 100               | Motor Ready         | 70     |
| 100 - 110              | Motor Hot!          | 90     |
| 110 - ____             | Motor Very Hot!     | 110    |



Also ensure that the following settings are made before completing the string lookup table:

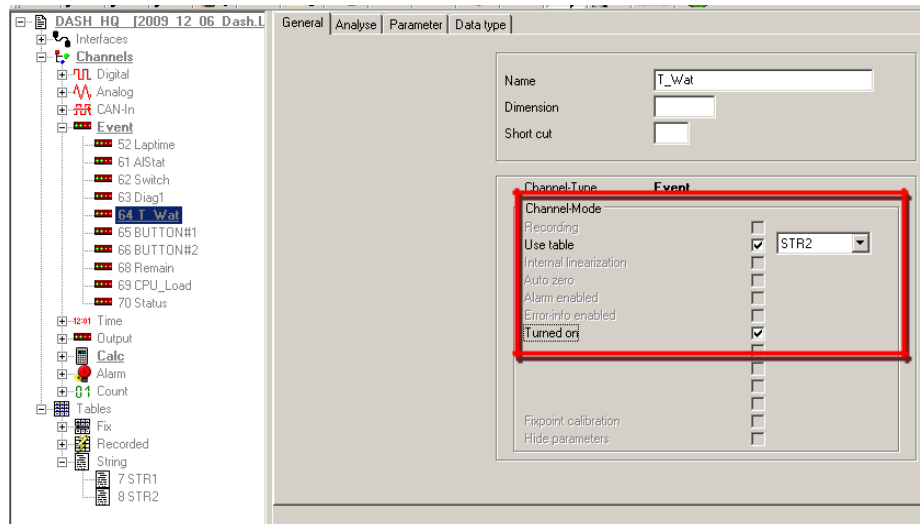
- ☐ Go to <**File/Save**> to permanently store this table inside your computer
- ☐ Save the file table with the same name e.g. 'T\_Wat', **and be sure of the directory into which it is saved.**

**Step 2** - To load the defined string table into 2D Dash:

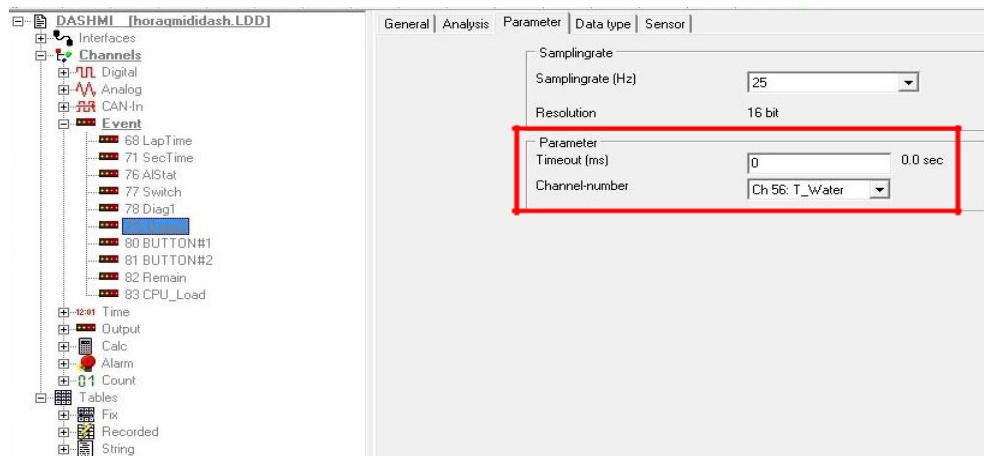
- ☐ Open the 2D program WinIt and make communication with the 2D Dash
- ☐ Select the node "**Tables**" from the system tree
- ☐ Expand the "**String**" section within the "**Tables**" node
- ☐ Select the table position named "**STR2**", this is used for the Diag2 function.
- ☐ Click <**Load from disk**> on the main window
- ☐ Locate the directory into which the "**STR2**" table was saved
- ☐ Select the table "**STR2**" and click <**Open**>

**Step 3** - To define the settings of the Event Channel Diag2:

- ☐ Open the **“Event”** node of the system tree, select **“Diag2”**
- ☐ Check the box for **“Use Table”**, and choose the previously created table **“STR2”**
- ☐ Check the box for **“Turned on”**, this will activate the Diag2 function



- ☐ Select the tab **<Parameter>** and nominate the data channel (from **<Channel-number>** dropdown box) to use as input for the created string lookup table. In this example the



analog input channel **“T\_Water”** is selected.

For this example a **<Timeout>** of “0” is used, meaning the message is always displayed while the string table and input channel value persist. If a timeout of “1000” was defined, the displayed message would disappear from Page 3 of the screen after 1 second (provided the displayed message does not require to change in that time!). The message will come back to the screen if or when the data channel value changes enough to make the displayed message also change.

#### 4.9 Remain event

The channel "**Remain**" shows the difference between the current date and time to a specified time.



- It can be used as a countdown to show the rider/driver the remaining time to the end of a practice session or any other time related event.

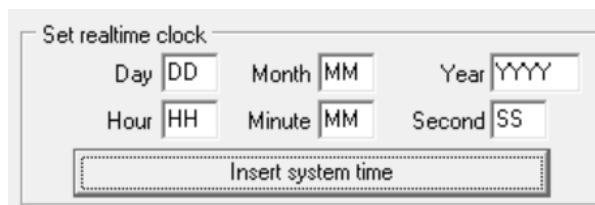
The Remain channel is configured as shown below:

1. Select *Remain* event in the system tree and navigate to tab **<General>**
2. Select the **<Turned on>** check-box to switch on Remain event.
3. Navigate to tab **<Parameter>**, enter the required end time in the **<End time [hh:mm]>** field.



- Unless the end time is modified, the Remain event will count down to this exact time every day. After the time has elapsed, the value of "Remain" stays at 0 until the next day begins.

4. Select respective module in the system tree and navigate to tab **<Realtime Clock>**
5. Insert the correct system time or press button **<Insert system time>** to use time of computer used and press **<Apply>**




- After pressing **<Apply>** the values of area *Set realtime clock* are reseted to placeholders again.

6. Select group *Digital* in the system tree and check channels *#RTC\_...* if realtime clock was set correctly



- If remain time is less than 10 minutes, the format of remain time is switched automatically from MM:SS to M:SS.H

**The Remain-Event-channel can be sent via CAN to other 2D CAN modules (e.g. Dashboards) or recorded like other channels!**

#### 4.10 Dorna Events

The TransponderX2 system is used in many racing classes for lap and section timing and for further communication between race control (Dorna) and the drivers.

For this purpose, the beacon loops are embedded in the asphalt on many tracks.

Beside transmitting the lap- and sectiontime-triggers to the measurement systems on the bike, the TransponderX2 system can also be used for sending messages, penalties, warnings and flags to the riders dashboard during the race.



- For more information about TransponderX2 please see the respective manual which can be found at our website:

[www.2D-Datarecording.com/manuals/](http://www.2D-Datarecording.com/manuals/)

##### 4.10.1 DornaFlags

At older 2D modules (with *DornaString* instead of *DornaMessages* event), the DornaFlags-CAN-In channel must be linked manually to modules *DornaFlags* event in *Parameter* tab.



- Newer 2D modules (with *DornaMessages* Event) does not need any further actions with linking the DornaFlags-CAN-In channel because this is already done in firmware!

##### 4.10.2 DornaMessages

At older 2D modules (with *DornaString* instead of *DornaMessages* event), the DornaMessages-CAN-In channel must be linked manually to modules *DornaString* event channel in *Parameter* tab.



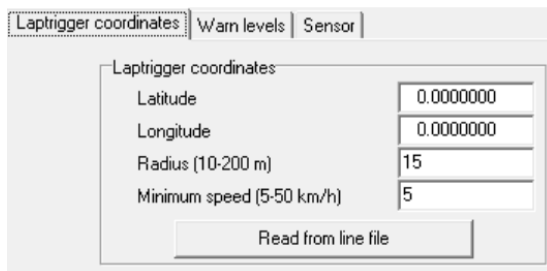
- Newer 2D modules (with *DornaMessages* Event) does not need any further actions with linking the DornaMessage-CAN-In-channels because this is already done in firmware!

#### 4.11 LapDetect - Set finish line coordinates via manual trigger

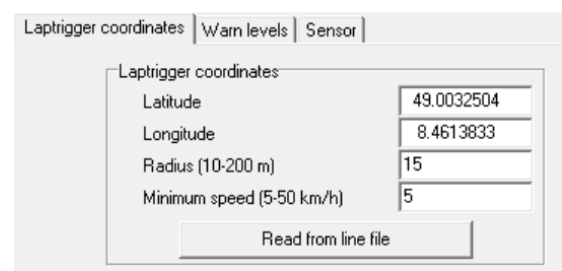


- The LapDetect events are only available in GPS/GNSS2CAN modules!

The *LapDetect* event can be used to manually set the current Latitude and Longitude coordinates as laptrigger GPS coordinates of channels #LapGPS **online on track**.



Laptrigger coordinates | Warn levels | Sensor  
 Laptrigger coordinates:  
 Latitude: 0.000000  
 Longitude: 0.000000  
 Radius (10-200 m): 15  
 Minimum speed (5-50 km/h): 5  
 Read from line file

Laptrigger coordinates | Warn levels | Sensor  
 Laptrigger coordinates:  
 Latitude: 49.0032504  
 Longitude: 8.4613833  
 Radius (10-200 m): 15  
 Minimum speed (5-50 km/h): 5  
 Read from line file

Via *Laptrigger coordinates* are then used for Laptimes event to create laptimes via GPS coordinates.



- Detailed instructions on how to generate laptimes via GPS coordinates be found at: <http://2d-datarecording.com/downloads/manuals/>

As laptrigger-set-condition any channel, e.g. analogue channel connected to light switch or horn button, can be used.



- The *LapDetect* event is not setting **laptriggers** directly but writes the current Latitude and Longitude coordinates as the **GPS coordinates** of channel #LapGPS!
- When *LapDetect* event is used, GPS laptime must be **calculated** in GPS/GNSS2CAN module and then **only send** to other devices for recording/displaying!

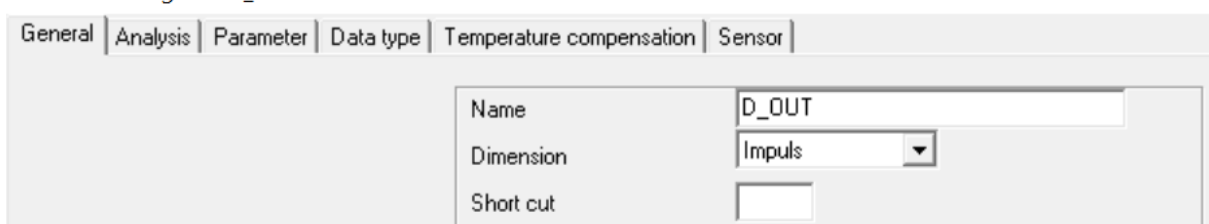
The LapDetect event consists of three single events:

- LapDetect: Set trigger conditions for writing GPS coordinates to #LapGPS
- LapDetectSpeed: Set minimum speed for valid trigger condition
- LapDetectStatus: Showing current state of *LapDetect* event

#### Important information:

*LapDetect* event can only be activated if dimension of output channel *D\_OUT* is set to *Impuls*.

Channel-Setting 115 D\_OUT



General | Analysis | Parameter | Data type | Temperature compensation | Sensor  
 Name: D\_OUT  
 Dimension: Impuls  
 Short cut:

This document is subject to change at 2D decision. 2D assumes no responsibility for any claims or damages arising out of the use of this document, or from the use of modules based on this document, including but not limited to claims or damages based on infringement of patents, copyrights or other intellectual property rights.

#### 4.11.1 LapDetect

Setting the trigger conditions for writing laptrigger GPS coordinates to channel *#LapGPS*.

|                   |                    |
|-------------------|--------------------|
| Parameter         |                    |
| Timeout (ms)      | 1000 1.0 sec       |
| Channel-number    | HornSwitch (Ch 59) |
| Trigger threshold |                    |
| Trigger when      | value smaller than |
|                   | 100.0              |

**Timeout (ms):** Setting time the trigger channel must be active

**Channel-number:** Selecting the trigger channel for writing GPS coordinates to *#LapGPS*

**Trigger when:** Setting trigger condition

#### 4.11.2 LapDetectSpeed

Setting the minimum speed for valid trigger condition which is selected in event *LapDetect*.



- Only if event *LapDetect* and event *LapDetectSpeed* are both TRUE, the current Latitude and Longitude coordinates are written to laptrigger GPS coordinates of *#LapGPS*
- The new laptrigger coordinates are just effective after a power-cycle of the module!

|                   |                   |
|-------------------|-------------------|
| Parameter         |                   |
| Timeout (ms)      | 0 0.0 sec         |
| Channel-number    | V_Sat (Ch 73)     |
| Trigger threshold |                   |
| Trigger when      | value bigger than |
|                   | 40.00             |

**Timeout (ms):** Setting time the speed channel must be minimum above a threshold

**Channel-number:** Selecting the speed channel to check

**Trigger when:** Setting minimum speed



- At *LapDetectSpeed* event only the condition *value bigger than* can be used!

#### 4.11.3 LapDetectStatus

**1:** LapDetect event switched on

**2:** LapDetect event active (Trigger condition TRUE)

**16:** Trigger condition AND minimum speed both TRUE → Latitude and Longitude GPS coordinates are written to *#LapGPS*

#### **4.12 CPU Load**

##### **4.12.1 CPU\_Load\_AVG**

Delivers the average CPU load of the module.

##### **4.12.2 CPU\_Load\_MAX**

Delivers the maximum CPU load of the module.



#### 4.13 *EventTrg*

The *EventTrg* event channels contains every information about the current state of the recording device and about exact time information of all events occurring during recording. Because *EventTrg* channels logs the exact time information of the occurring event, it can be recorded with very low sampling rate.



- The *EventTrg* information is very important for debugging purposes.

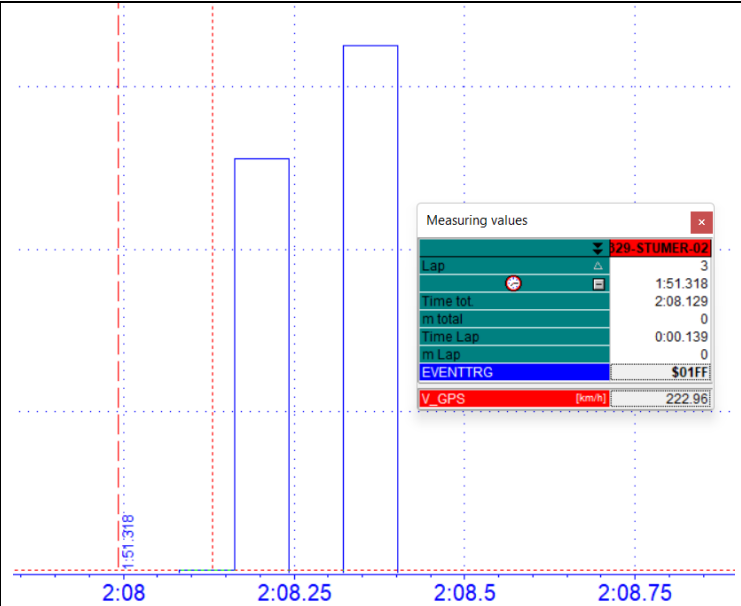
By the time information the exact time of occurrence can be recalculated with 1ms resolution!

One *EventTrg* information always consists of three different channel states.

1. Type of current *EventTrg*
2. Time information of current *EventTrg*
3. Cancellation of current *EventTrg*

**Example:**

Type: Laptrigger (0x01FF)



Exact Time information **where event was occurring** EventTrg.

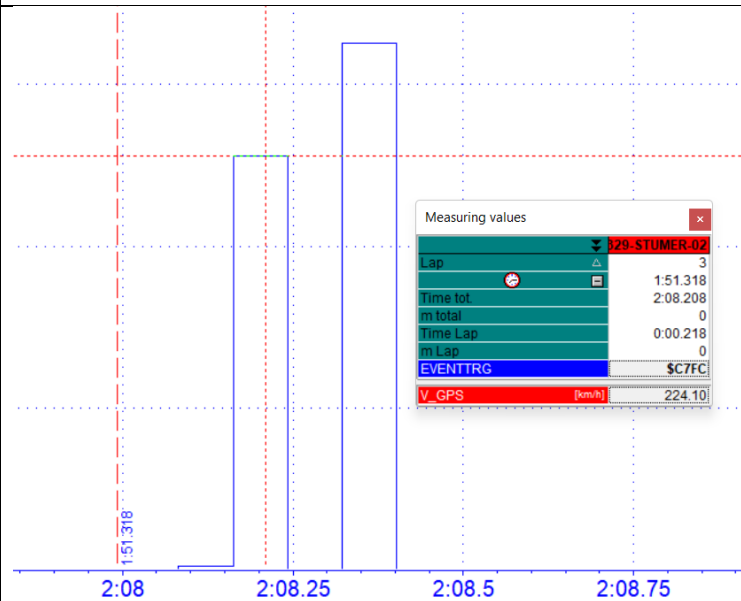


Time information must be multiplied with *Multiplier* and *Offset* from following table.

0xC7FC → 51196

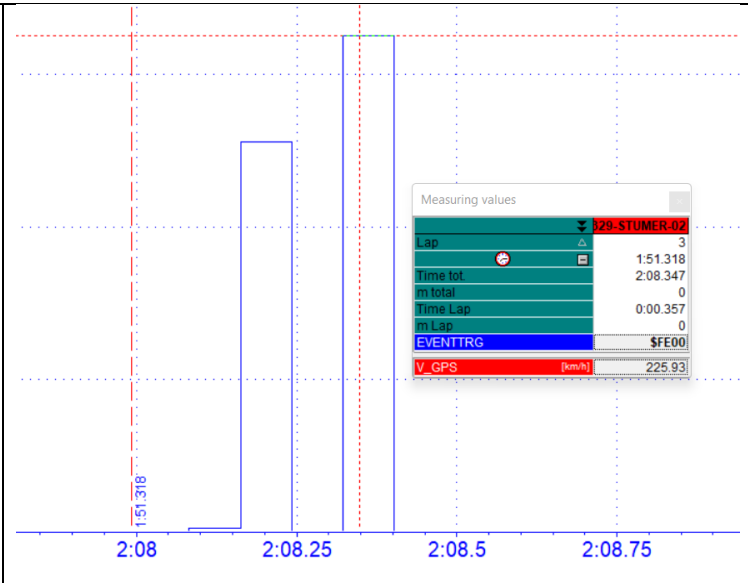
$51196 \cdot 0.0025 = 127.990 \text{ sec}$

→ 2:07.990 min



EventTrg end: Inverse value of Type

0x01FF → inverse → 0xFE00



| Type   | Data information              | Description                  | Note                   |
|--------|-------------------------------|------------------------------|------------------------|
| 0x01FF | LapTime [s] - M=0.0025 Off=0  | Laptrigger                   |                        |
| 0x0200 | SecTime [s] - M=0.0025 Off=0  | Sectrigger                   |                        |
| 0x01FD | RecTime [s] - M=0.0025 Off=0  | On/Off Trigger               |                        |
| 0x01FC | RecTime [s] - M=0.0025 Off=0  | Stopp/Start der Aufzeichnung |                        |
| 0x01FB | RecTime [s] - M=0.0025 Off=0  | Top Speed                    | <b>Dashboards only</b> |
| 0x01FA | RecTime [s] - M=0.0025 Off=0  | Pit Entry                    | <b>Dashboards only</b> |
| 0x01F9 | RecTime [s] - M=0.0025 Off=0  | Pit Exit                     | <b>Dashboards only</b> |
| 0x01F0 | Error Bits                    | Error Status Logger          |                        |
| 0x01D0 | X2 FLAGS_MSG                  | X2 Flags                     | Dorna X2               |
| 0x01D1 | X2 INFO_MSG                   | X2 Info String               | Dorna X2               |
| 0x01D2 | X2 BOX_CALL/CLEAR             | X2 Box Call                  | Dorna X2               |
| 0x01D3 | X2 TEAM_NOTIFICATION          | X2 Team Notification         | Dorna X2               |
| 0x01D4 | X2 COMM_TRANSMIT_QUEUE_STATUS | X2 Queue status              | Dorna X2               |
| 0x01D5 | X2 MYLAPS_PROPRIETARY         | X2 Mylaps proprietary        | Dorna X2               |
| 0x01D6 | X2 COMM_ID                    | X2 Transponder UID           | Dorna X2               |
| 0xF1   | CAN Error - [Bus][Error]      |                              |                        |
|        | X = CAN-Bus [1..6]            |                              |                        |
|        | Y = 0x01 - STUFF Bit Error    |                              |                        |
|        | Y = 0x02 - No ACK             |                              |                        |
|        | Y = 0x03 - CRC                |                              |                        |
|        | Y = 0x04 - Form Error         |                              |                        |
|        | Y = 0x05 - Bit Error          |                              |                        |