

## Content

<b>1</b>	<b>2D MOTO2™ KIT SYSTEM</b> .....	<b>4</b>
1.1	GENERAL STRUCTURE.....	4
1.2	GENERAL INFORMATION .....	5
1.3	SYSTEM OVERVIEW .....	5
1.3.1	2D Moto2™ data logger .....	7
1.3.2	Moto2™ engine interface module.....	7
<b>2</b>	<b>PUTTING YOUR SYSTEM INTO SERVICE</b> .....	<b>9</b>
2.1	CONNECTOR LAYOUTS/ASSEMBLY .....	9
2.1.1	Moto2™ front side connector layout .....	9
2.1.2	Moto2™ data logger back side connector.....	9
2.1.3	Assembly of “Tyco crimp contacts” into 34 pin AMP connector.....	10
2.1.4	Exchange or disassembly of a “Tyco contact pin” from the AMP plug .....	11
2.2	MOUNTING AND CONNECTING THE MODULES .....	11
2.2.1	Mounting the engine interface module to the data logger .....	11
2.2.2	Mounting the Moto2™ data logger to the bike.....	13
2.2.3	Mounting the LAF (lambda) sensor .....	13
2.2.4	Mounting the GPS mouse .....	14
2.2.5	Mounting the front speed sensor.....	14
2.2.6	Mounting the front suspension sensor.....	15
2.2.7	Mounting the rear suspension sensor.....	16
2.2.8	Mounting the brake pressure sensor .....	17
2.3	FIRST STEPS WITH THE SOFTWARE.....	18
2.3.1	Installation of the software .....	18
2.3.2	How to install the 2D USB drivers.....	18
2.3.3	Licensing of the 2D software .....	19
2.3.4	How to update the software via Internet.....	20
2.4	WINIT FOR MOTO2™.....	21
2.4.1	Configuration of the measured channels.....	22
2.4.2	Basic calibration steps (2 common examples).....	23
2.4.3	GPS module configuration .....	25
2.4.4	TMS configuration.....	25
<b>3</b>	<b>WORKING WITH THE SOFTWARE</b> .....	<b>27</b>
3.1	GENERAL INFORMATION ON THE DATA STRUCTURE .....	27
3.1.1	Create a new event directory.....	27
3.1.2	Change event directory .....	28
3.1.3	Communication with the logger - the program WinIt .....	29
3.2	HOW TO DOWNLOAD DATA.....	30
3.2.1	How to prepare the download .....	30
3.2.2	How to change the logger name?.....	31
3.2.3	Why to change the logger name? .....	32
3.2.4	How to start a download? .....	32
3.3	DATA ADMINISTRATION WITH THE 2D PROGRAM SPECVIEW .....	33
3.3.1	What is SpecView?.....	33
3.3.2	What are SpecSheet files? .....	33
3.3.3	What do the SpecSheet files do? .....	34
3.3.4	Naming of SpecSheet files .....	34
3.3.5	Correctly using your permanent info file.....	35
3.3.6	SpecView and using SpecSheet files.....	36
3.3.6.1	The mileage function .....	36
3.3.6.2	Default permanent info file for Moto2™.....	39
3.3.6.3	Printing SpecSheet data.....	41
3.3.6.4	Exporting SpecSheet data.....	42
3.3.6.5	Further explanation of the SpecSheet export .....	44
3.4	SUMMARY OF MEASURED CHANNELS .....	46
3.5	2D PROGRAM ANALYZER .....	49

3.5.1	Fixed analysis layouts for Moto2™ kit license.....	50
3.5.2	Analysis tools for Moto2™ kit license.....	51
3.5.3	Google Earth™ Link.....	54
3.6	X2 SETTINGS .....	55
3.6.1	Receiving X2 transponder data.....	55
3.6.1.1	COMM_FIRST_CONTACT.....	56
3.6.1.2	FLAGS_MSG.....	56
3.6.1.3	INFO_MSG.....	57
3.6.2	Lap times.....	58
3.6.3	Section times.....	59
3.6.4	Flags.....	60
3.6.5	Dorna info messages.....	61
3.7	GPS CHANNELS .....	62
3.7.1	Generate lap times with GPS.....	63
3.7.1.1	Automatic setting of LapGps.....	63
3.7.1.2	Manual Setting of LapGps.....	65
4	APPENDIX .....	67
4.1	TRACK LOOPS .....	67
4.2	FLAG INDICATIONS .....	68
4.3	MEASURED CHANNELS GENERATION.....	69
4.3.1	Engine interface module.....	69
4.3.2	Moto2™ data logger.....	70
4.3.3	CAN IDs for Moto2™ data recording system.....	71
4.4	DETAILED INFO ABOUT SOFTWARE FOR MOTO2™ KIT SYSTEM .....	72
4.4.1	Calculated channels.....	72
4.4.1.1	Default calculated channels for Moto2™ .....	73
4.4.1.2	CalcTool entries for Moto2™ calculation channels .....	74
4.4.1.3	Making your own CAL files for analog channels Volt_1 and Volt_2.....	81
4.4.1.4	Adding a CAL file to the AutoCalc configurator list.....	83

## Symbols used in the text



These paragraphs contain tips and practical advice for working with the 2D software



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



Documentation reference

➤ A user manual reference number is provided so the user can seek further assistance

# 1 2D Moto2™ kit system

## 1.1 General Structure

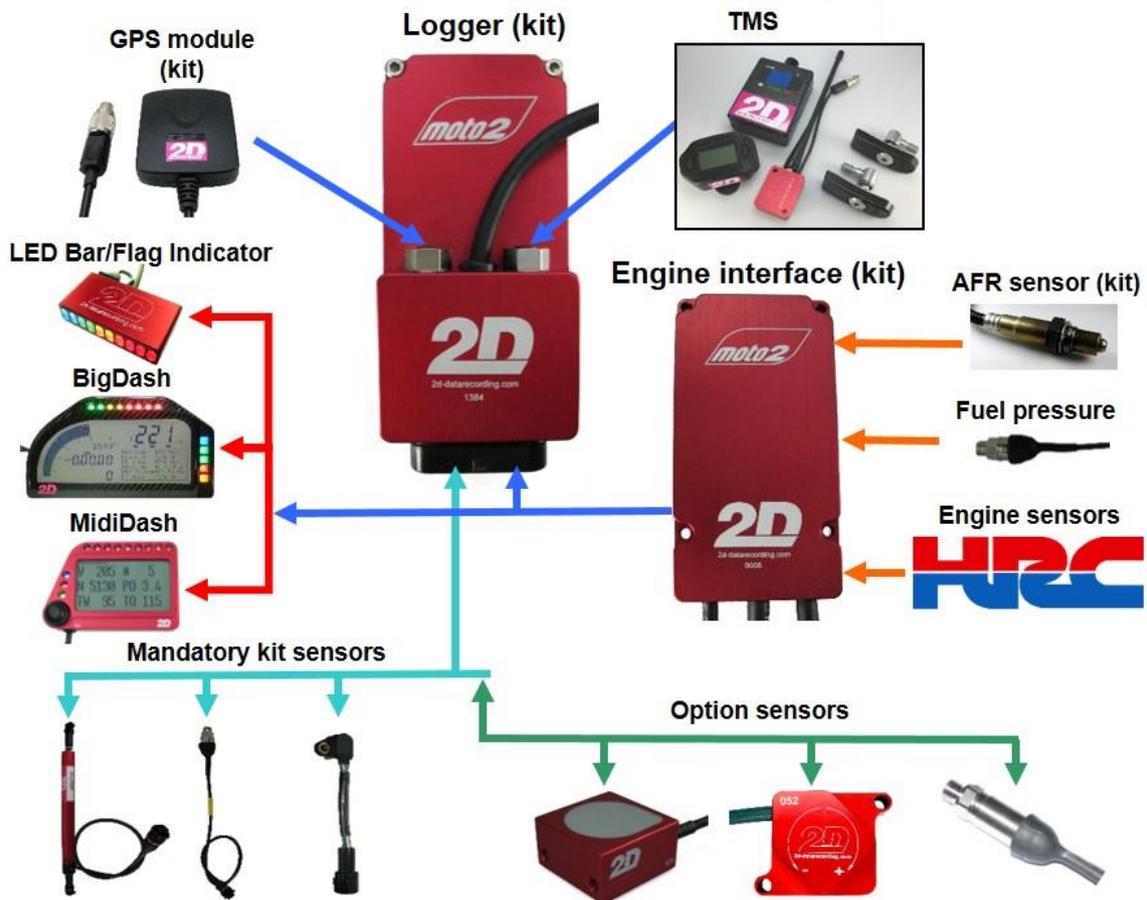
The 2D Moto2™ kit system is a modular data recording system built-up by several different devices. The system consists of a fixed setting data logger (*LG-μCAN11\_Moto2-211*), an engine interface module (*BC-LAF-Moto2-213*), a Tire Monitoring System (TMS) and a basic sensor kit that **must be used**.

The “heart” of the basic kit is represented by the data logger, which combines three devices in one housing (a data logger, a memory and a GPS receiver). By connecting a “GPS mouse” the reception of GPS coordinates at a rate of 10 Hz or 12.5 Hz becomes possible. A CAN connection between the data logger and the engine interface module enables the measurement and recording of Air/Fuel Ratio (AFR) and several important engine sensors.

Additional modules can be added easily, including any 2D display unit (BigDash, MidiDash or LED Bar/Flag Indicator).

Many more optional modules can be used for bike development, but **can only be used at private test sessions**. Options include using an **unfixed setting μCAN11\_Pro** or *\_Eng* data logger, strain gauge kit, thermocouple kit and a 2D infrared lap trigger.

Many options exist and feel free to ask 2D for more information on the available options.



## 1.2 General information

The data recorder is supplied as a “kit system” and many of the internal settings are fixed at the request of Dorna Sports. Despite being a fixed-setting system, the Moto2™ data recorder maintains the high standard of functionality that 2D has built into all its products for many years.

<p><b>Basic Moto2™ kit:</b></p>	<p>The basic Moto2™ data recording kit enables the user to measure essential variables of the bike and store them as data inside the data recorder memory. The recorded data can be downloaded to a PC and analyzed using the supplied Moto2™ kit software. <b>The following devices are included in the kit:</b></p> <ul style="list-style-type: none"> <li>• Moto2™ data logger, <i>LG-µCAN11_Moto2-211</i></li> <li>• Moto2™ engine interface module, <i>BC-LAF_Moto2-213</i></li> <li>• Moto2™ kit software CD, <i>RaceMoto2_14.0</i></li> <li>• 10 Hz GPS receiver module, <i>AC-GPS_Mouse_uC09-000</i></li> <li>• Cable for downloading data</li> <li>• Brake sensor adapter-cable</li> </ul>
<p><b>Standard kit sensors:</b></p>	<ul style="list-style-type: none"> <li>• Suspension sensor (150 mm), <i>SA-LP150D-200</i></li> <li>• Suspension sensor (75 mm), <i>SA-LP075D-200</i></li> <li>• Front brake pressure sensor (100 bar), <i>SA-PK100M10-200</i></li> <li>• Front wheel speed sensor, <i>SD-VI05-200</i></li> <li>• Lambda probe, <i>SA-LSU4.2-000</i></li> </ul> <p><b>Tire monitoring kit:</b> This is a kit that enables measurement of your tires’ air pressure and temperature during a race. The tire monitoring kit contains special sensors for fitting to your wheel and a short-range wireless transmitter module that connects to the CAN bus of your data logger. Data from the tires are then transmitted to a 2D receiver device and will then be recorded by the data logger.</p>
<p><b>Optional sensors:</b></p>	<ul style="list-style-type: none"> <li>• Oil pressure sensor (10 bar), <i>SA-PK010M10HT-200</i></li> <li>• Fuel pressure sensor (10 bar), <i>SA-PK010M10HT-200</i></li> <li>• Front/rear axle vertical acceleration sensor, <i>SA-AC010HQ1-200</i></li> <li>• Banking rate gyro, <i>SA-GY200V4-200</i></li> <li>• 12 Hz GPS receiver module, <i>AC-GPS_Mouse_12Hz-000</i></li> </ul>
<p><b>Optional race modules:</b></p>	<p><b>2D display modules:</b> 2D can provide additional display modules that connect to the Moto2™ kit system for displaying further important bike information to the rider. The displays currently available are:</p> <ul style="list-style-type: none"> <li>• MidiDash, <i>DI-Midi11-201</i></li> <li>• BigDash, <i>DI-Dash_6/8_2C-200</i></li> </ul> <p>There are also special devices for displaying flag signals available:</p> <ul style="list-style-type: none"> <li>• LED Bar (with flag recording), <i>DI-FLAGSig-000</i></li> <li>• Flag Indicator (pure display unit), <i>DI-FLAGSig_Base-000</i></li> </ul>
<p><b>Optional modules for TESTING ONLY:</b></p>	<p><b>Expansion of system with extra µCAN data logger:</b> An additional µCAN data logger (engineer or professional specification) can be connected to the CAN bus of your existing Moto2™ data logger. This lets you extend the system capabilities and use many thermocouples, strain gauges and other sensors on the bike for development purposes.</p>

## 1.3 System overview

The Moto2™ data recorder system has been developed so that all 2D devices (data logger, engine interface module, display) can function correctly with the Moto2™ standard Honda ECU (electronic

control unit) and the associated wiring harness. All measurements made by the system are made possible by the sensors fitted to the bike/engine.



2D supplies the sensors used on your Moto2™ **bike** and several **engine** sensors. Other **engine** sensors must be supplied by Honda/Geo Technologies.

- The 2D “**bike sensors**” (suspension front and rear, brake pressure front, Gyro, etc.) are connected directly to the Moto2™ data logger (see connection instructions section 2.1.1). The data from these sensors are recorded by the data logger.
- The 2D “**engine sensors**” (lambda probe, probe temperature and heating power) are connected directly to the 2D engine interface module, from which the accumulated data is sent to the data logger via CAN bus.
- Additional engine sensors are supplied with the Moto2™ engine by Honda/Geo Technologies. The engine sensors are all connected to the standard Moto2™ ECU. Engine sensors for RPM (engines crank-shaft rotation speed) and RPSprkt (gearbox output shaft rotation speed for the engine) are also connected directly to the Moto2™ data logger.
- Standard engine sensors for throttle, T\_water (temperature of the engine’s coolant fluid (water)), IMAP (air pressure inside the air box) and T\_Air (air temperature inside the air box) are connected to both the Honda ECU **and** the 2D engine interface module. The “*ECU Error*” output of the Honda ECU is also connected with the engine interface module to enable analysis of this. Additionally the option exists for connecting a 2D fuel pressure sensor.

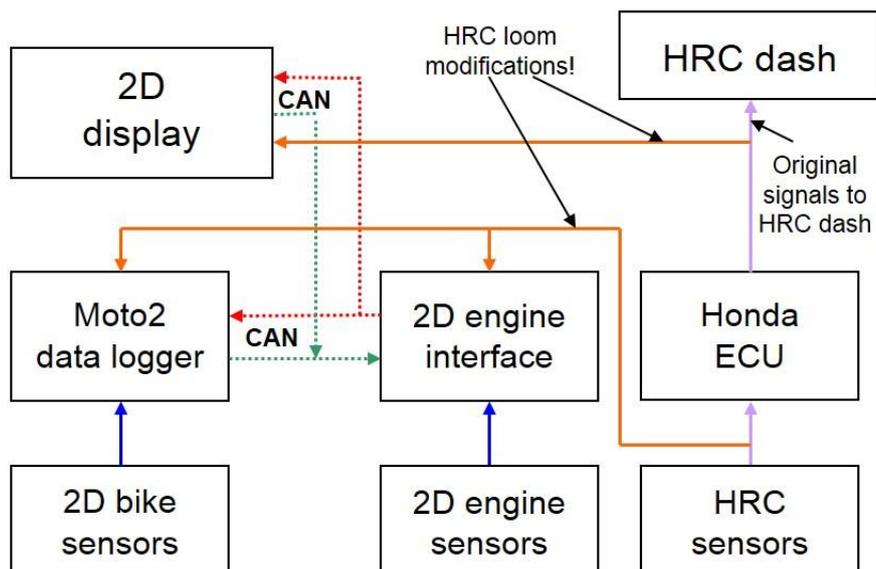


Since 2011 the option exists to measure fuel pressure using a 10 bar 2D sensor. This is connected to analog input 6 of the 2D engine interface module with data sent to the datalogger via CAN bus.

All data accumulated by the engine interface module are sent to the data logger by CAN bus and recorded by the data logger for later offline analysis.



All data inside the 2D data recording system can be displayed to the rider (or engineer) by sending them to a 2D display (BigDash/MidiDash). Alternatively the standard HRC dashboard can be used, however this can only display basic channels for RPM and water temperature.



The diagram illustrates the “flow” of data within the Moto2™ data recording system.



The HRC dash does not have a CAN bus! Therefore the only data that can be displayed to the rider using the HRC Dash is the data generated by the HRC sensors and routed to the dash by many wires.

To maximize the quality of your Moto2™ system, it is recommended to use a 2D display module. 2D can currently provide the BigDash and MidiDash for use on Moto2™. All devices receive data via CAN bus and provide a highly customizable visual interface for the rider.

The CAN bus communication enables any channel measured by the system to be displayed to the rider (or an engineer in the pit box). Also many **alarms** and **calculation channels** can be defined by the user to display to the rider (or engineer). These can be combined to provide **logic based alarm functions** to inform the rider of low oil pressure! Contact 2D for more information on display modules.

### 1.3.1 2D Moto2™ data logger

The main functions of the Moto2™ data logger device are summarized as follows:

- Enable direct connection of many bike sensors
- Provide power supply for connected sensors
- Enable connection of a GPS antenna module
- Enable configuration and calibration of all data channels
- Recording all data channels according to defined configurations
- Receive data from 2D engine interface module via CAN bus
- Send and receive data from 2D digital display via CAN bus
- Make connection with PC/laptop for data download and analyzing



The data logger has two completely independent CAN buses: **CAN\_2D** (CAN-1) and **CAN\_EXT** (CAN-2).

- CAN 2D (CAN-1) is used to exchange the bike data.
- CAN EXT (CAN-2) is used by Dorna Sports for delivery of bike data to an on-board camera during TV Broadcasts.

### 1.3.2 Moto2™ engine interface module

The engine interface module has three important functions.

#### 1. An Air/Fuel Ratio (AFR) sensor control and measurement system to perform:

- measurement of the voltage generated by the lambda probe
- measurement of lambda probe temperature
- actuation of lambda probe heating element to control probe temperature



The lambda probe heating element requires a 12 V power supply **and uses a significant portion of the bikes battery power!**

To preserve battery power the heating element of the lambda probe is only active, when the engine is running (RPM > 0).

#### 2. A junction into which the following engine sensors are connected:

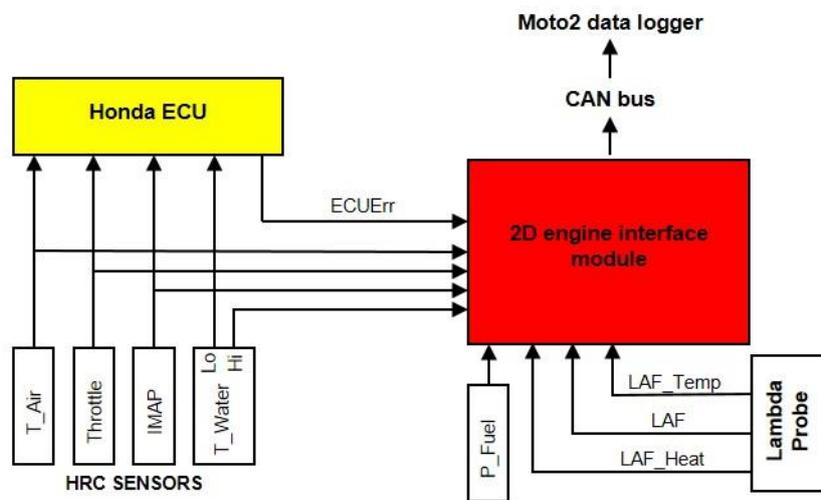
- HRC throttle position
- HRC engine coolant temperature (high temperature element)<sup>1</sup>
- HRC intake manifold air pressure (IMAP)
- HRC air intake temperature

<sup>1</sup> HRC water temperature sensor has two elements: one designed for accurate low temperature measurement and one for high temperature measurement. Consult HRC for further details on this device.

- HRC ECU Error signal
- 2D fuel pressure sensor (recommended option)

**A CAN bus device to send the following engine data to the data logger for recording:**

- LAF measured value
- LAF temperature
- LAF heating
- Throttle position
- Engine coolant temperature (high temperature element)<sup>1</sup>
- Intake manifold air pressure
- Air intake temperature
- ECU Error signal
- Fuel rail pressure



Connections made by the engine interface module

The engine interface module has 6 analog input channels dedicated to the engine sensors of the bike. These are occupied by *Throttle*, *T\_Water*, *IMAP*, *T\_Air*, *ECUErr* and *P\_Fuel*. The engine interface module has no memory to record data. It performs as an “intermediate device”, taking measurements of the engine sensors and sending the values to the data logger via CAN bus.

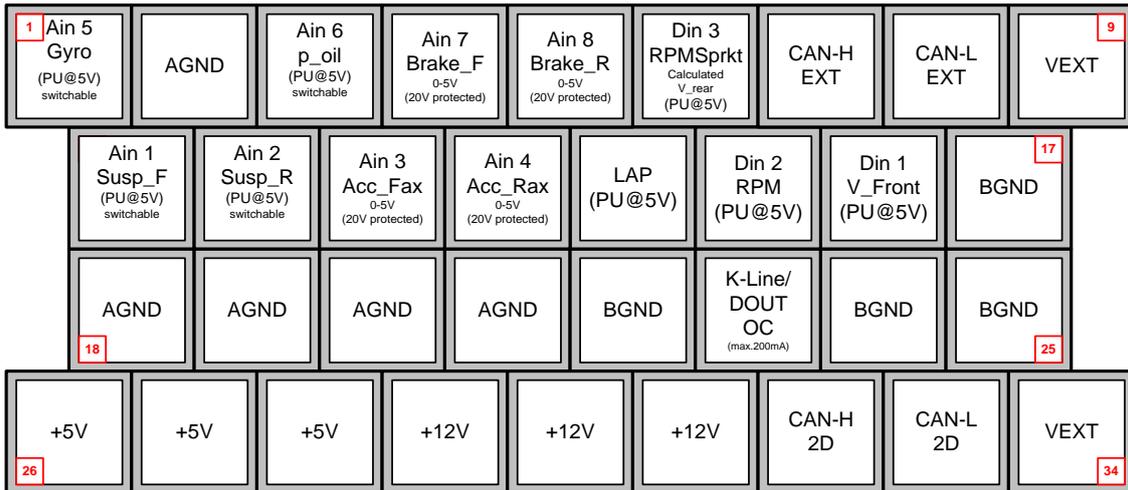
## 2 Putting your system into service

### 2.1 Connector layouts/assembly



The wires delivered by 2D are with the “Tyco contact pins” already crimped. Just insert the individual contacts into the 34 pin AMP connector.

#### 2.1.1 Moto2™ front side connector layout

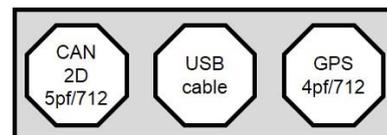


34 pin connector for Moto2™ logger



Front side of Moto2™ data logger

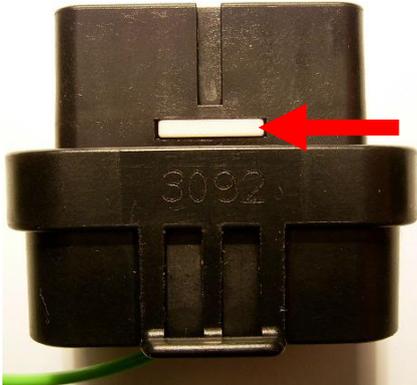
#### 2.1.2 Moto2™ data logger back side connector



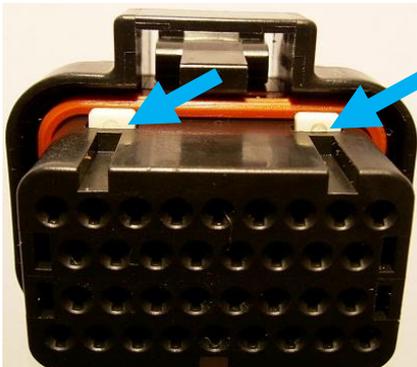
### 2.1.3 Assembly of "Tyco crimp contacts" into 34 pin AMP connector



"Tyco contact pin" crimped onto wire



**First of all the plug locking device must be released.** Therefore the rectangular "white plastic tip" (marked with red arrow) has to be pushed with gentle pressure into the plug. Use a small tool, e.g. a screwdriver or the point of a knife. You shall hear a "click" when the locking device opens.

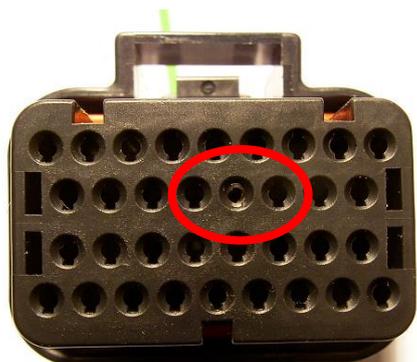


**Make sure that the 34 pin AMP connector is not locked:** the two small rectangular "white plastic tips" (marked with blue arrows) have to protrude from the connector's body. **Only in this position the "Tyco contact pins" may be slid into the AMP plug and also be removed again.**

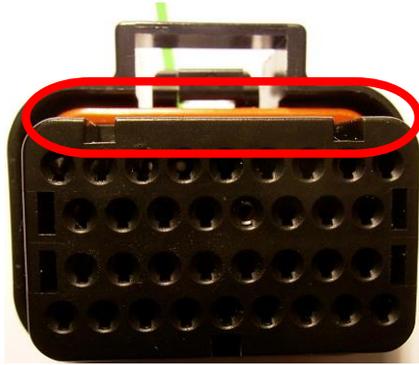
In the next step the "Tyco contact pin" must be inserted into the correct position. The position of the individual contacts can be determined from the table in section 2.1.1.



Press the "Tyco contact pin" into the back side of the AMP plug...

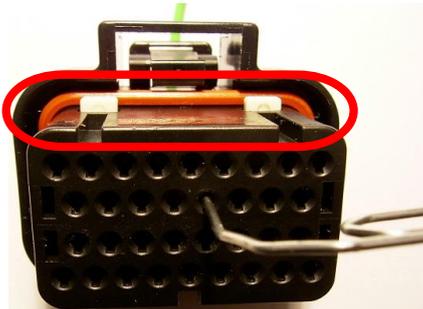


...until it becomes visible from the front side of the connector.

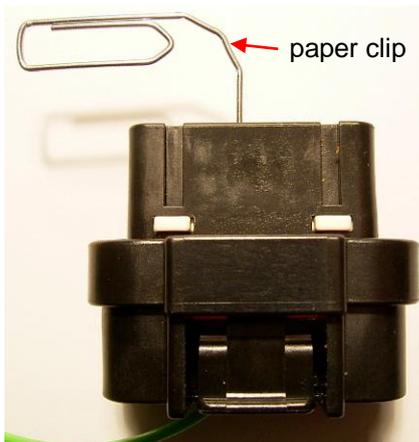


When all contacts are correctly installed, the AMP plug must be locked again. You might use a screwdriver for that purpose. **Press the two “white plastic tips” into the plug until they become even with the surface of the connector.** The locking device engages audibly with a “click”.

### 2.1.4 Exchange or disassembly of a “Tyco contact pin” from the AMP plug



**Make sure that the 34pin AMP connector is not locked:** the two small rectangular “white plastic tips” have to protrude from the connector’s body. **Only in this position the “Tyco contact pins” may be slid into the AMP plug and also be removed again.**



To extract the “Tyco contact pins” you might use a paper clip or a similar tool: just push the pin carefully out.



Please pay attention not to use a tool that is too thick. Otherwise you might damage the contact pins by widening them. In this case you will not be able to re-use this pin because a correct connection cannot be assured any longer!

## 2.2 Mounting and connecting the modules

### 2.2.1 Mounting the engine interface module to the data logger

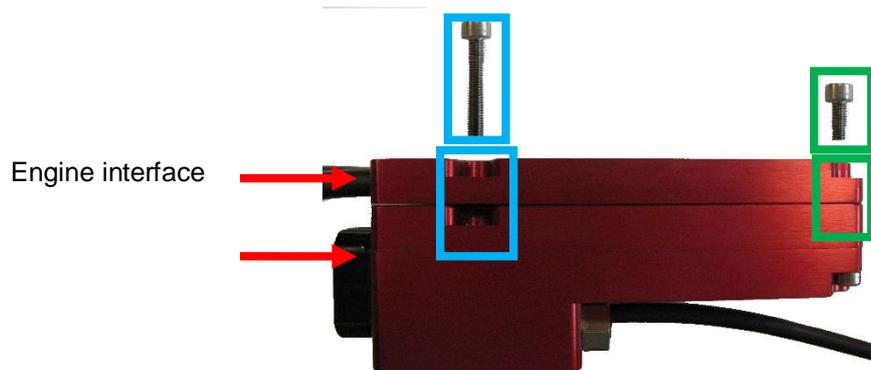


It is recommended to bolt the engine interface module to the logger: First remove all the bolts from the logger housing. To attach the controller unit to the logger as shown use the longer bolts delivered with the engine interface unit.

Turn *LG-μCAN11\_Moto2-211* as shown on the picture below. Remove the marked screws.



Place the *BC-LAF\_Moto2-213* unit onto the logger as shown below. Note the position of the mounting points and the difference in screw lengths.



Use the screws to fix the engine interface module to the logger housing, tighten the screws carefully



While fitting the two housings together, take care that the connectors are pointing to the same side!

### 2.2.2 Mounting the Moto2™ data logger to the bike

There are many ways to mount the data logger to the bike. However, please take the following topics into consideration:

- Is the data logger secured firmly or can it fall off?
- Can the data logger be removed quickly for routine bike maintenance?
- Can the data logger USB download lead be easily accessed during sessions?
- Will the mounted position expose the data logger to very high temperatures?
- Is the data logger very likely to be damaged if the bike crashes?

It is recommended to mount the data logger with adhesive Velcro® that can strongly fix it to the bike, but also be removed quickly for bike maintenance.

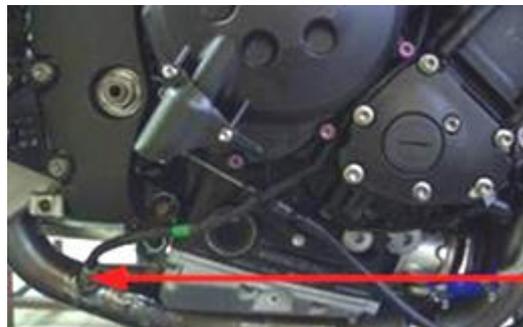


In addition to the considerations listed above, the axis positioning of the datalogger should be made so the **internal 3 axis accelerometer** is used to full effect. Each axis of the datalogger (x, y, z) contains a sensor that can measure acceleration of the **data logger** in that direction. By correctly aligning (if possible) the accelerometer axes of the data logger with the bike's (forward-back, up-down, left-right) then the acceleration of the **bike** in these directions will be measured correctly.

### 2.2.3 Mounting the LAF (lambda) sensor



The LAF (lambda) sensor should be installed onto the exhaust collector close to where the 4-2-1 pipes meet



LAF sensor

Many exhaust systems provide already a thread to attach a lambda sensor. If this is not the case: to fit the LAF (lambda) sensor, determine first the exact position of the LAF sensor.



Be sure that the swing arm, the linkages, the bodywork or other parts will not interfere with the sensor or the sensor cable at any position. If possible the sensor should **not** be mounted with its thread at the lowest position to avoid damage by condensing water!

The best way to fix the LAF sensor at the selected point to the exhaust collector is to fit a screw collar (or nut) and weld it onto the exhaust collector. This screw collar must guarantee an exact, stable and sealed fitting for the LAF sensor. The LAF sensor tip should reach about 15 mm into the gas flow to give correct values.

Once you have fixed the screw collar, a hole has to be drilled into the exhaust collector (diameter should be inner screw collar diameter) and then drilling should be cleared using a tap with the same thread as the sensor. You can also first drill the hole and then weld the collar in-line onto the pipe maintaining the correct angle for the sensor.



If you are not very experienced with welding exhaust parts we strongly recommend to pass this part of the assembly process to a workshop with competence (and good experience) for that type of modification. Failing to weld the adapter correctly may crack the sensor in the moment of fixing. It has to enter softly.

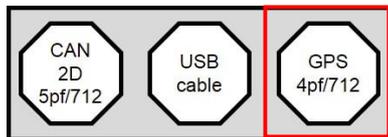
Incorrect modifications can also damage the exhaust - so be careful!

### 2.2.4 Mounting the GPS mouse

For optimal signal quality the GPS mouse must have a **free visibility towards the GPS satellites**. Therefore it should not be covered by any bike parts or by the rider. Mounting the antenna on the rear tail of the bike would be a useful solution for example. To get better signals use aluminum foil as a ground plane underneath the GPS mouse.



Double-sided “scratch tape” or Velcro® works very well to fix the GPS module. It keeps the GPS receiver fixed on its place but can be removed easily as well. The GPS mouse also has an integrated magnet on the lower surface. This allows simple and fast mounting of the receiver on all magnetic surfaces (e.g. body parts of the used vehicle).



Rear view of the 2D kit logger  
(schematic overview)



2D GPS mouse



Using the GPS mouse does not require any further action. After connection it is ready for use and will be powered by the kit logger.



The power supply of the complete system **must be switched on** before testing the GPS mouse. Incoming GPS data can be displayed online in the 2D software *WinIt*.

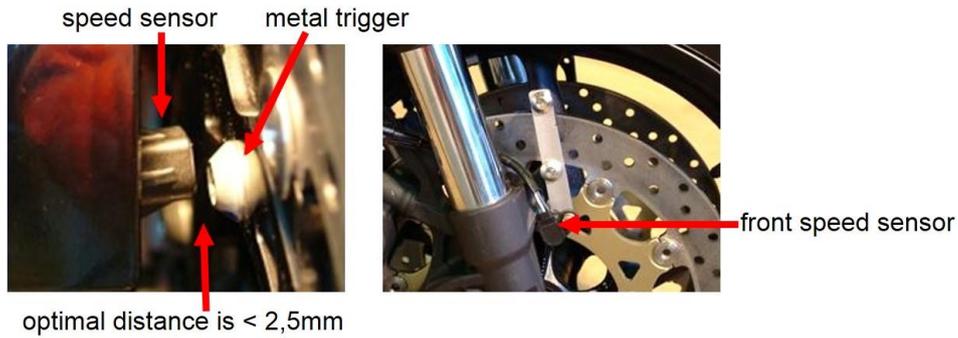
### 2.2.5 Mounting the front speed sensor

Unlike its name suggests, a speed sensor does not measure speed, at least not directly. The speed sensor generates an impulse each time one or more “triggers” pass in front of the sensor. The function of the wheel speed sensor is based on the induction of a voltage inside the sensor as a metallic component (head of a bolt/screw) passes close to the sensor.

The logger counts the number of occurrences per unit of time, then uses the number to determine the rotation speed of the wheel which, in multiplication with the wheel circumference, gives the speed of the vehicle. It is recommended to fit the speed sensor in close proximity to the brake disc mounting bolts (the bolts fixing the brake disc carrier to the wheel). The top of each bolt must pass the sensor very closely, but make sure it cannot contact the sensor! Each bolt passing the sensor will generate an electrical impulse inside the sensor that the data logger will use to generate wheel speed.



You can fix the front speed sensor with a support fixed to the right front fender using the same screws of the front fender!



You should pay attention to the mounting position of the sensor. The speed sensor **must not be centered** above the head of the bolt (figure 2). The “circular path” of the metal trigger should not “meet” the centerline of the sensor. Shift the speed sensor approx. 3 mm up or down to prevent it from sending two pulses at each bolt (figure 1). Where possible use flat-headed screws as this will improve the performance of the sensor.

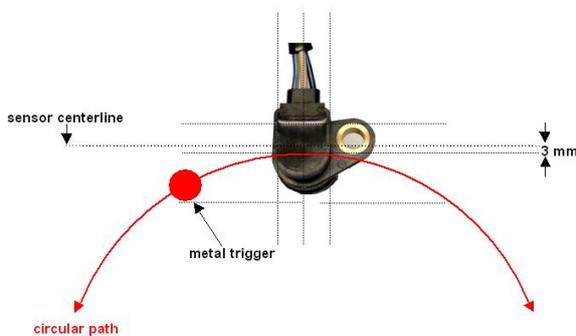


Figure 1: optimal mounting

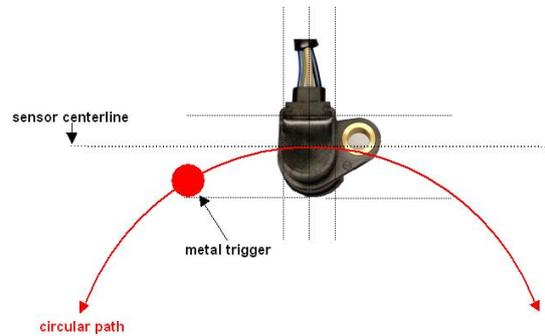


Figure 2: “bad” mounting

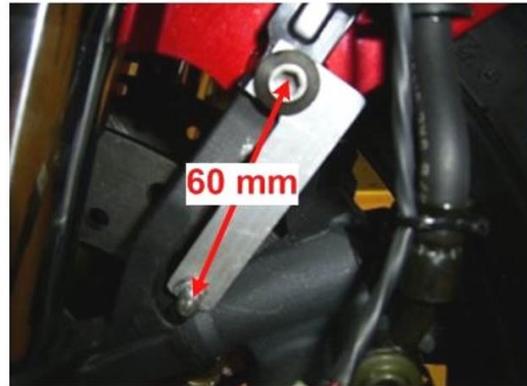
### 2.2.6 Mounting the front suspension sensor



The potentiometer has to be fitted parallel to the tubes of the fork to give correct values. The maximum measure range of the used sensor must be equal or even longer than your maximum suspension travel (150 mm range is normal for front fork measurement).



special mounting kit to fix the upper part of the front suspension sensor



special mounting kit to fix the lower part of the front suspension sensor



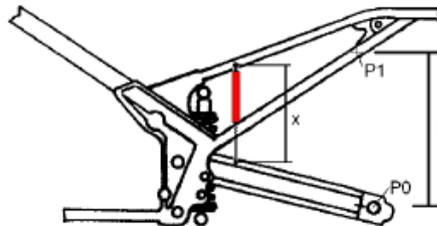
Please check **very** carefully that neither the sensor nor the mountings will limit the steering angle. Also be sure that no cables of brake lines can get caught in the sensor or get damaged by it.

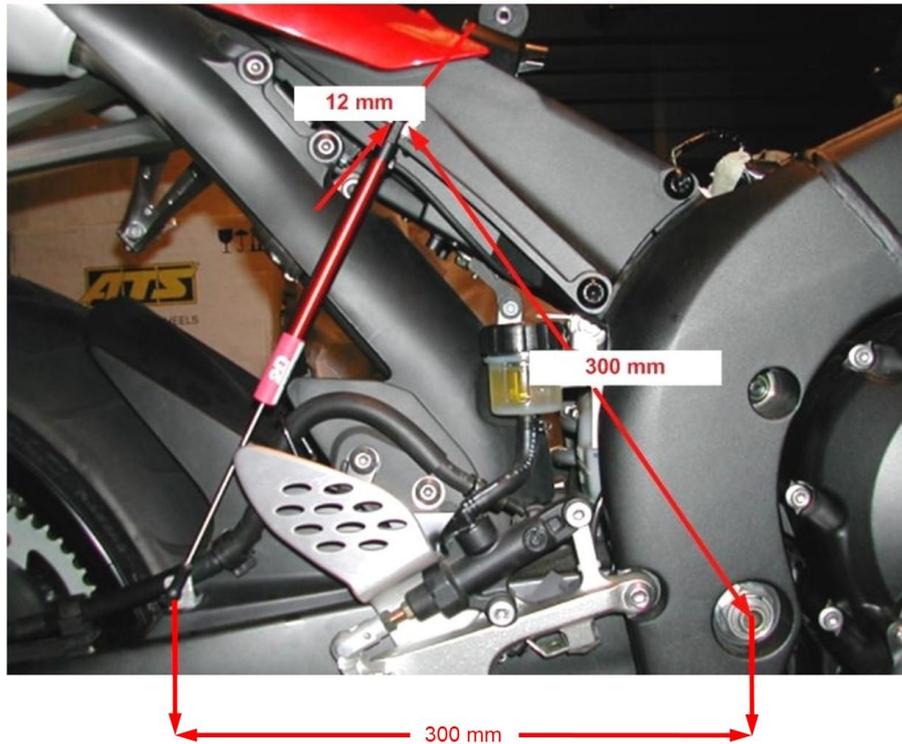
### 2.2.7 Mounting the rear suspension sensor



To measure the suspension travel of the swinging arm it is in some cases impossible to mount the sensor directly parallel to the (standard) shock absorber. In this case the sensor must be mounted somewhere between the swinging arm pivot and the rear wheel axle. This causes big differences of usable sensor measure length depending on the location of the mounting. If positioned close to the swinging arm pivot, a relatively short sensor can successfully measure rear suspension position (50 mm or 75 mm will work in most cases)

To avoid water entering the housing, please mount the slider bar in the downward position!





View of the rear suspension sensor (linear potentiometer 75 mm), e.g. mounted on a Yamaha R1



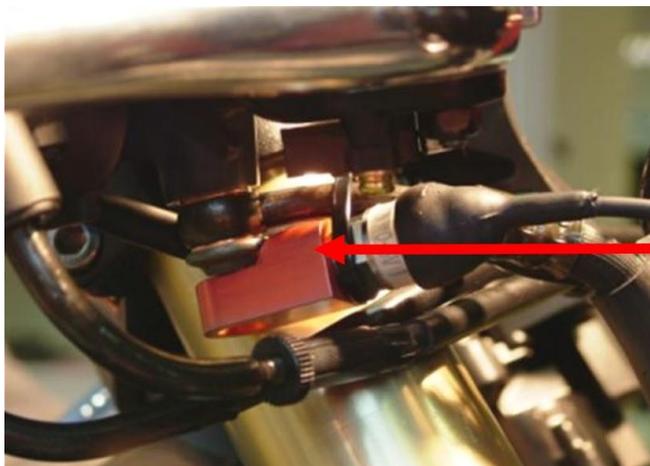
Check **very** carefully that neither the sensor nor the supports make contact with any moving parts to prevent the normal operation of the bike suspension. Check that both maximum and minimum travel are possible with the sensor fitted!

### 2.2.8 Mounting the brake pressure sensor

You may wish to use 2D brake pressure sensors for both the front and rear brake systems.



The picture below shows a possibility to fix the pressure sensor to the brake line system. For this a special mounting kit is included in the sensor add-on kit, which fixes the pressure sensor with a double hollow screw to the main brake cylinder.



2D adapter for brake pressure sensor



Be very careful to ensure that all brake connections are returned to full tightness, as failing to do so will compromise the operation of the brake system.

## 2.3 First steps with the software

### 2.3.1 Installation of the software



Before starting the installation of the 2D Moto2™ software please assure that you have **administrator rights** on your computer. Otherwise the USB driver files cannot be installed correctly

1. Insert the 2D CD.  
The installation process will start automatically.  
If this is not the case, start the "AUTORUN.EXE" on the CD manually.
2. Start installation by selecting the **<Install>** button at the bottom left.
3. A set-up wizard guides you through the installation.  
Follow the instructions and confirm your actions (in most cases with **<Apply>**).



All required files including demo data will be installed automatically. These can be found on the same drive that your operating system is installed on. If your operating system is installed on drive "C:" then your demo files will be stored at "C:\Racedata\Demodata\".

### 2.3.2 How to install the 2D USB drivers

The process of the 2D USB driver installation depends on the operating system you are using and therefore differs accordingly. We support the following operating systems: Windows XP, Windows Vista and Windows 7 or newer.



**Please note:** The necessary USB driver files will be copied to your PC during the software installation.

Therefore the software installation has to be completed before installing the drivers!

#### **USB driver installation for Windows operating systems XP/ Vista/ 7 or newer.**

##### **Preparation:**



**Before connecting the logger with the USB cable to the computer, connect it to external power supply of 12 V.**

After you connect the logger with your PC the system detects the new hardware.

The "Found new Hardware Wizard" will guide you through the installation. Confirm the dialogs as shown on next page to finalize the USB driver installation.

## Installation of the “2D virtual COM port device”

### Windows XP – 32 bit



**Dialog 1:** Confirm with button <Next>



**Dialog 2:** Confirm with button <Next>



**Dialog 4:** Finalize with button <Finish>



**Dialog 3:** Confirm always with button <Continue Anyway>

### Windows XP – 64 bit

In dialog 2 of the sequence shown above select the option “Install from a list or a specific location (Advanced)”. In the following select the CD drive with the 2D CD inserted. The installer will the automatically locate the driver on the CD.

### Windows 7 or newer

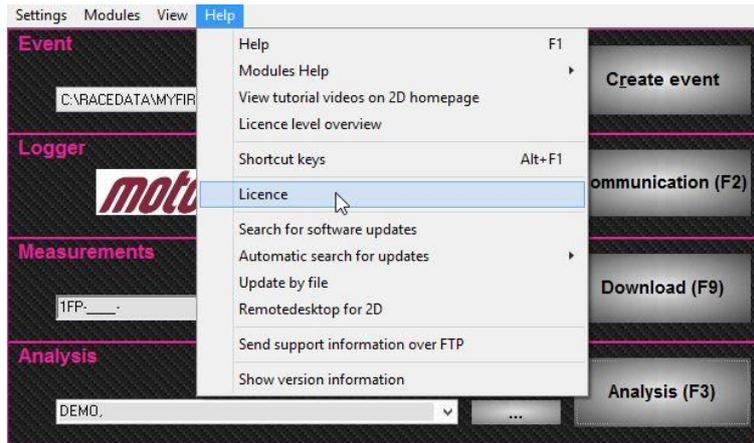
If you have a newer operating system, and the computer has a working connection to the Internet follow the instructions of the “Found new hardware Wizard” and allow it that it can look for the drivers in Windows Update. The installation will then happen automatically.

If the computer is not connected to the Internet follow the same procedure as described for Windows XP – 64 bit.

### 2.3.3 Licensing of the 2D software

To be able to use all the features of your software, you have to license it.

1. Start *WinARace*
2. Choose “Licence” in the *Help* menu



3. Please fill in a license name - beginning with **Moto2\_** - (e.g. Team name or your name) and your contact details.
4. Under license level you have to select the license level you have ordered with your kit system.



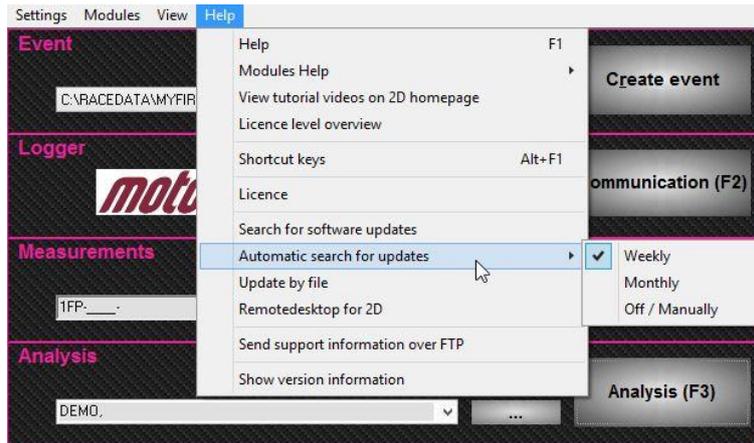
5. Send the form via e-mail to 2D by clicking on the **<E-Mail>** button.
6. You will get your license key and serial number back from 2D by email!
7. Fill the information (*Licence key, Serial number*) in the respective fields.
8. Finish the licensing process by clicking the button **<Store licence>**

After successfully licensing the software you can use all features available to your license level.

### 2.3.4 How to update the software via Internet

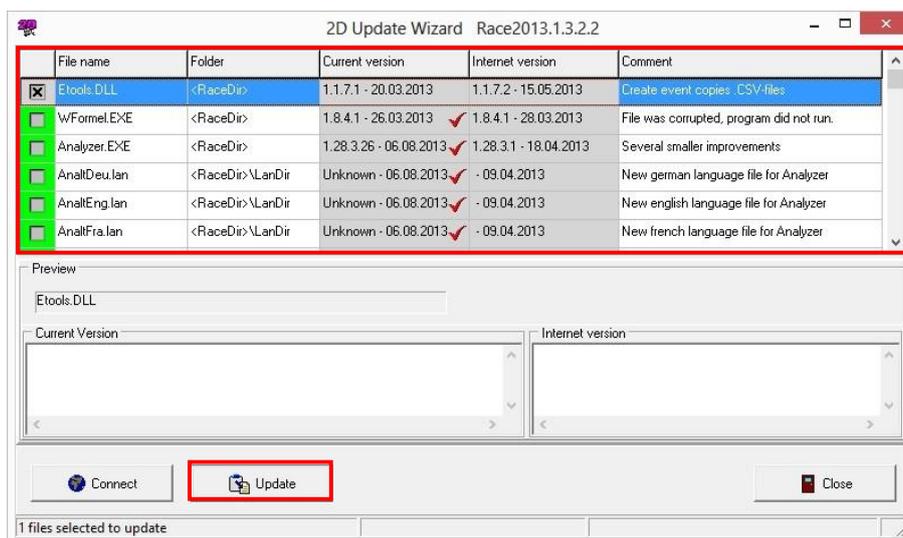
From time to time 2D publishes improvements for your installed software. In order to always have the latest versions you should search periodically for updates.

1. Start *WinARace* and select menu item **<Help>**
2. A) for manually search: select **<Search for software updates>**  
 B) for automatically search: select **<Automatic Search for updates>** and choose an update interval



The main menu items are also available via right mouse click. That's very practical, if the main menu is hidden.

- The "2D Internet Update Wizard" offers a list of possible files for update. Check all boxes from the list you want to update.



- Confirm your selection with the button <Update>

## 2.4 WinIt for Moto2™

From the WinARace panel you can establish communication with your Moto2™ data recording system by clicking the button <Communication (F2)> or by pressing <F2> on your keyboard.



After connection has been established between the Moto2™ data recorder and your PC, the WinIt simplified kit user interface will appear as shown below.



The *WinIt* simplified kit user interface performs as a “control panel” that lets you view the current values of standard measured data channels for the Moto2™ kit system. The currently active devices of your Moto2™ data recording system can be viewed in the system tree as shown above.

From this panel you can begin a download of your bike’s measurement data by either pressing the key <F9> on your keyboard or by selecting the button <Download (F9)> from the screen as shown above.

Alternatively you can erase all data from the memory of the logger by either pressing the key <F3> on your keyboard or by selecting the button <Empty (F3)> from the screen as shown above.



It is highly recommended that you **DO NOT** use this option unless you are certain that all data inside the logger is invalid and not useful!



The front end can be used to quickly apply channel settings and to calibrate or zero a sensor where appropriate.

### 2.4.1 Configuration of the measured channels

Using the *WinIt* front end, the following settings can be performed on the listed channels:

- **V\_Front** – input of tire circumference and number of pulses per wheel rotation
- **P\_Oil** – set present value of oil pressure to zero
- **Susp\_F** – set zero position of the front suspension
- **Throttle** – perform full calibration AND/OR apply zero position for the throttle
- **Susp\_R** – perform full calibration AND/OR apply zero position to the rear suspension
- **ACC\_X/Y/Z** – perform full calibration AND/OR apply zero value for the accelerometer
- **Gyro** – make present value of the gyro equal to zero
- **Volt\_1** – set measured voltage value equal to zero, used for **Acc\_Fax** or option sensors
- **Volt\_2** – set measured voltage value equal to zero used for **Acc\_Rax** or option sensors
- **Brake\_F/R** – set present value of front/rear brake pressure to zero
- **P\_Fuel** – set present value of fuel pressure to zero



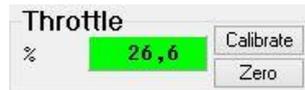
The processes required for setting the calibrations/zeros are well defined by the two examples shown in the following.

### 2.4.2 Basic calibration steps (2 common examples)

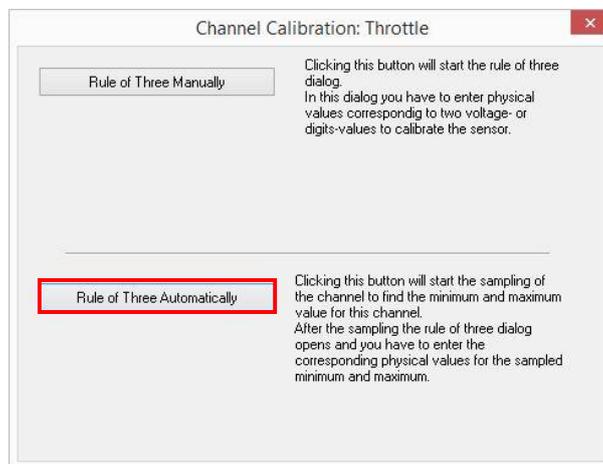
#### Example 1: Calibration of the throttle

To correctly calibrate your throttle sensor carefully follow these instructions step by step:

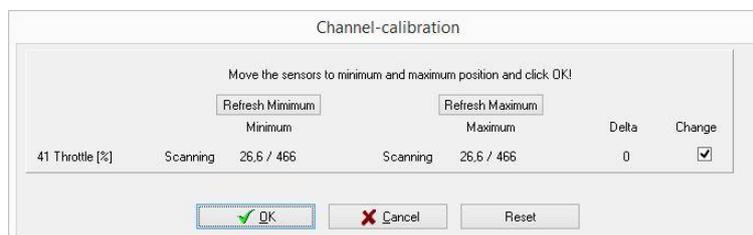
- Within the zone “Throttle” from the *WinIt* front end panel, click the button <Calibrate>



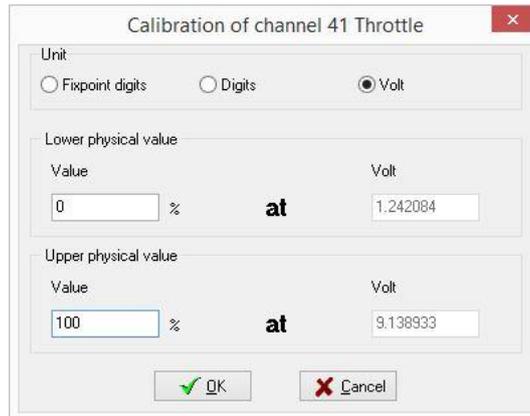
- You will be prompted with the screen as shown below, for the throttle you must click <Rule of Three Automatically>



- Move the throttle to the fully open position, press <Refresh Maximum> and hold this throttle position until the 2.5 second time count is complete
- Release the throttle to the closed position, press <Refresh Minimum> and maintain this condition until 2.5 second time count is complete



- Next you must enter the physical values attained by the sensor when the “Refresh Minimum” and “Refresh Maximum” operations were made. For this example the “Lower physical value” is 0 (zero) and the “Upper physical value” is 100.



- Click <OK> to complete the calibration of the throttle



If your data logger contains recorded data you will be shown a warning to explain that applying the new calibration setting will erase the memory of your data logger. Only continue if you are certain the data inside the logger is not useful!

Otherwise you can cancel the calibration, download your data and then repeat the calibration process for your sensor.



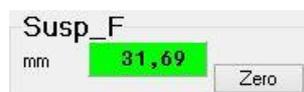
The calibration of the rear suspension sensor should be performed using the same method. However in this case you must carefully determine the lower and upper physical values of the sensor.

This can be performed with a ruler, but great care must be taken to not induce significant errors during calibration.

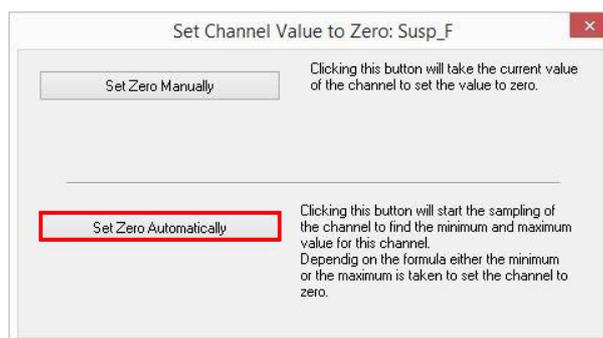
### Example 2: Setting the front suspension channel to zero

To correctly zero the measurement of your bike's front suspension carefully follow these instructions step by step:

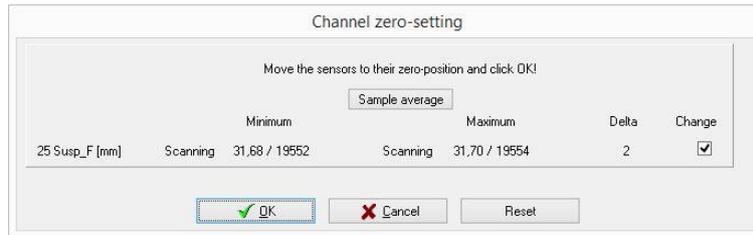
- First make sure your bike is on a pit stand that will allow the front suspension to rest at its bottom position with the forks fully extended
- Within the zone "Susp\_F" from the *WinIt* front end panel, click the button <Zero>



- You will be prompted with the screen as shown below. To correctly zero the front suspension you must click the button <Set Zero Automatically>



- Next you must click the button <Sample average> from the screen below



After the 2.5 second average value is determined, click <OK>



The new zero position of your bike's front suspension will then be modified inside the setting of your Moto2™ data logger.



If your data logger contains recorded data you will be shown a warning to explain that applying the new calibration setting will erase the memory of your data logger. Only continue if you are certain the data inside the logger is not useful!

### 2.4.3 GPS module configuration

Before you can correctly use the GPS module, you must first ensure it is correctly mounted to the bike.



It is important that the mounting instructions for the GPS module (section 2.2.4) are followed exactly. If the GPS module is incorrectly fitted to the bike, the quality of GPS channels may be reduced significantly!

Secondly you must make sure the data logger's "operation mode" is set according to the type of GPS module you are using. Normally the operation mode is set to "Autodetect". But by dropping an older setting to your logger, it may be that you have to choose a mode.

The data logger operation mode is changed by:

- selecting your Moto2™ data logger from the system tree
- selecting the tab "Operation modes"
- choosing the correct option from the dropdown menu

The GPS modules look very similar, they only differ in the color of the status LED. By connecting the GPS mouse to your data logger and looking at the color of the status light, you can identify if you are using a 10 Hz or a 12.5 Hz mouse and make the following settings:

- 10 Hz GPS mouse has red status LED ⇒ select operation mode "NMEA".
- 12.5 Hz GPS mouse has green status LED ⇒ select operation mode "3D\_1g".

If you have any questions about this feel free to contact 2D for assistance.

### 2.4.4 TMS configuration

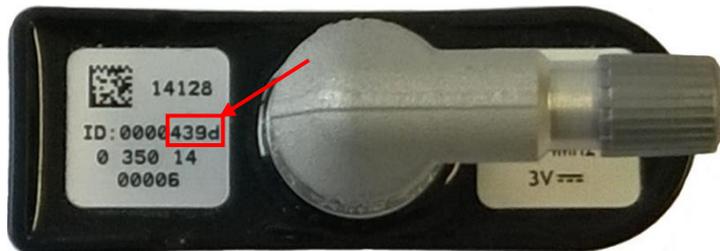
The task of the receiver unit is to receive information from your tires and send that information to the logger. To be able to identify the correct tires it has to have the IDs of the Wheel Unit Sensors (WUS).

To recognize the Wheel Unit Sensors (WUS) of your system you have to enter its ID into the TMS-code table. You can find the ID of the printed label on the sensor (last 4 characters).

“old” sensor



“new” sensor

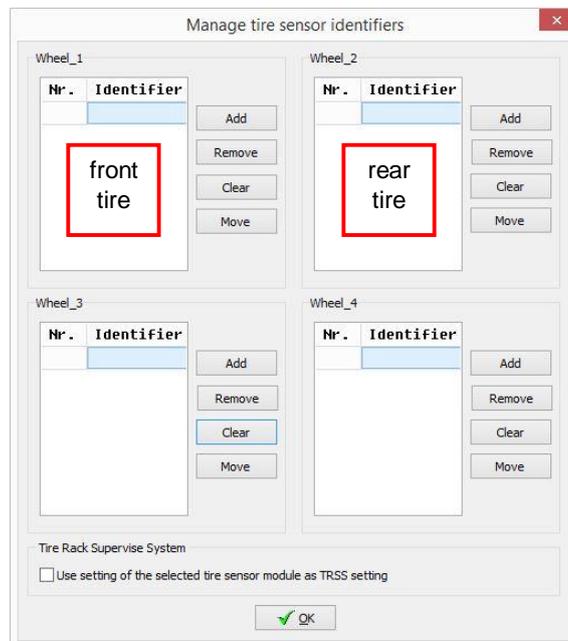


Connect your powered system to your PC and start *WinIt*.

To enter the WUS IDs click on the wheel icon. You can find it in the *WinIt* toolbar if you select your receiver unit in the system tree:



You can sort your WUS IDs in two columns. Wheel\_1 is meant for all front tires and Wheel\_2 is meant for all rear tires:



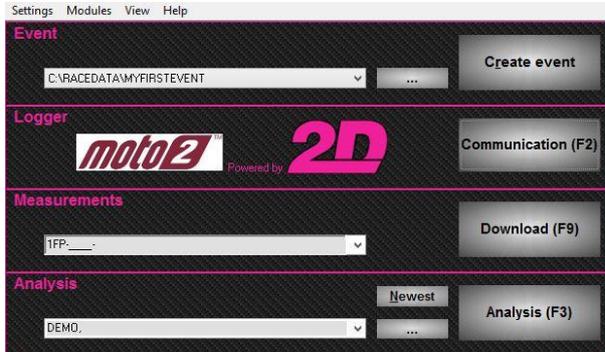
Enter your WUS IDs as printed (last four characters as marked in the picture above) in the corresponding column and save this table.

Confirm your changes with <OK> and <Apply>.

### 3 Working with the software

#### 3.1 General information on the data structure

WinARace, the front-end program started on the desktop, shows the following three levels:



- a.) Event/measurement administration
- b.) Logger communication
- c.) Measurement naming & download
- d.) Measurement selection & analysis

- To begin downloading data, select the button **<Download (F9)>** or press **<F9>** on your keyboard.
- At the time of download, the program *WinIt* generates a new folder inside the current **event directory**. The new folder contains all data files coming from the data logger during download. All measurement files have got the extension **\*.MES**
- The name given to the measurement is a combination of the current logger name and master name.
- Your measured data should be stored inside your computer according to the following directory structure:

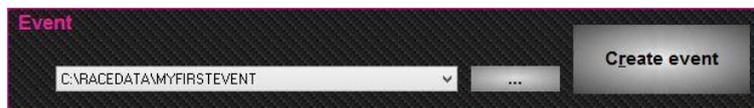


##### 3.1.1 Create a new event directory



The first and most important step **before** downloading the first measurement is to create an “event”. Always start with this step, so that you always know where your data has been saved!

From the diagram below, the current directory is shown on the left hand side. This defines where data will be saved or from where measurements are read.

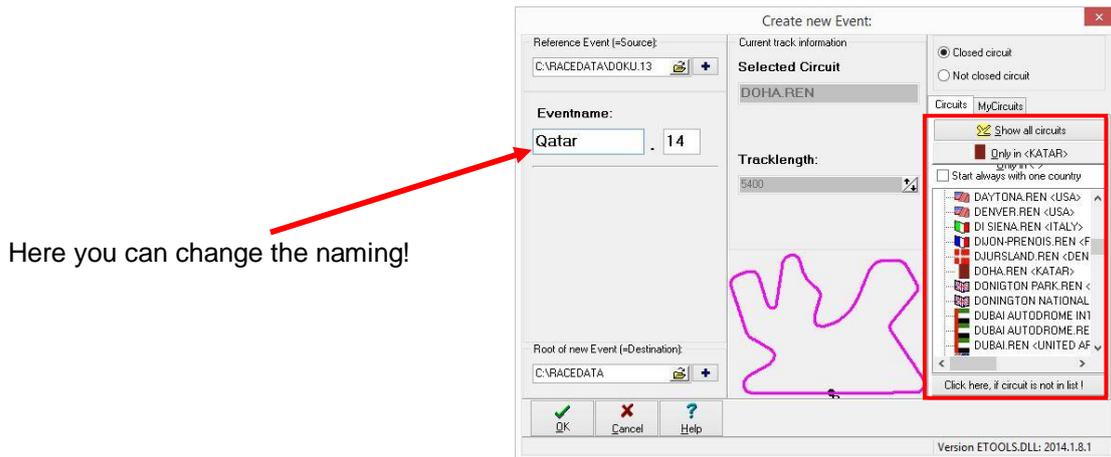


On the right side there are two options:

1. You can switch to the “event module” by clicking on the button **< ... >**.
2. **<Create Event>** will set up a new directory with the option to select the track used at that event.



To create new event directories use the button **<Create Event>** in the front-end tool *WinARace*. The following window will appear:



- Select a circuit from the list on the right hand side. The name of the selected track combined with the current date will become the default name of the event.
- To change the name to your preferences edit the field “Eventname”.
- Confirm your selection with the button <OK>.

**There are two different circuit sub-directories:**



1. The sub-directory “\Circuits“: Includes a list of **pre-defined tracks** delivered with the 2D race software.
2. The sub-directory “\MyCircuits“: Contains track maps which have been **created by the user** via the analysis tool *2D Analyzer*.



If a track is not listed, select **<Click here if circuit is not in list>** and a basic circuit will be selected (*Base.ren*). Rename the *Eventname* according to the track that the data are from.

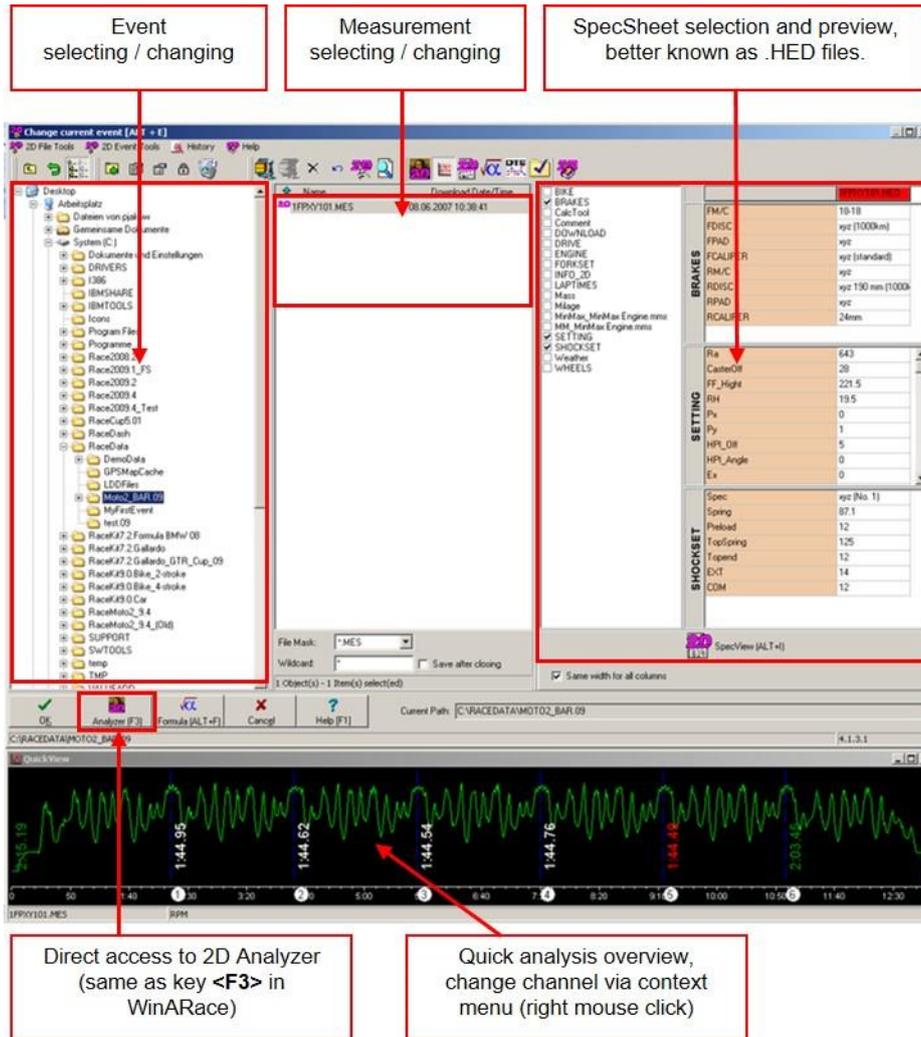
### 3.1.2 Change event directory



You have the possibility to change the current event by selecting the button < ... >. The figure on next page shows you the start-window of the “event module”

Inside the change event screen you can perform many administrative tasks including:

- Select a previous event from your computer directory
- Review *SpecSheet* info (lap times/bike settings/rider comments)
- Rename a measurement (this is the only recommended way to rename!)
- Delete a measurement



### 3.1.3 Communication with the logger - the program WinIt



In this dialogue box the logger and computer are linked to each other. Communication with the logger is established to download the measured data and set up the logger.

1. Activate main power switch of your bike, providing full power supply to data logger and sensors.
2. Connect the USB connector of the kit logger with the computer. (Make sure red LED is on.)
3. Start the communication program *WinIt*. (click on <Communication> or hit the hot key <F2>)



The software recognizes an attached Moto2™ system logger automatically and will display the following window.



It's VERY important to first activate the bike's power supply and then connect the data recording system to your computer by USB. Otherwise your USB port might get damaged.



This window allows the user to modify the circumference of the front wheel and also the number of pulses for one wheel rotation. Additionally you can calibrate the suspension and other analog sensors.



See section 3.3 for information on using *WinIt* to configure your Moto2™ kit system before making a measurement on race track.

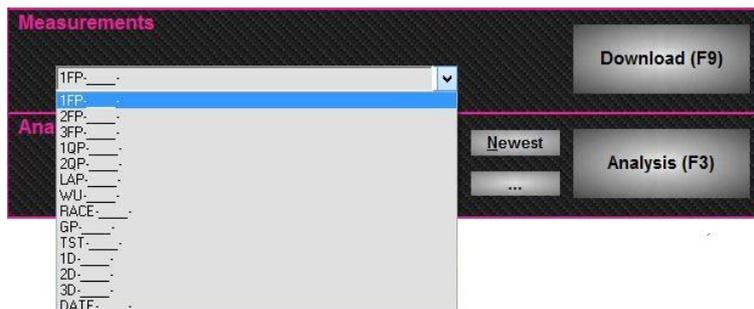
### 3.2 How to download data

#### 3.2.1 How to prepare the download



**Before** each run (or download) you should check the name that will be given to the measurement. Take care choosing the name to assure which measurement relates to which rider and session. Then it is a lot easier for you to work with the measurements later.

1. Switch to the *WinARace* start window.
2. On the left side of the logger-panel select an abbreviation from the drop-down list. Here the type of measurement you are making is defined by selecting a "master name".



The measurement name is a combination of the current master name (type of test) and the logger name. The master name is specified from *WinARace* (icon at your desktop) and the actual logger name you specified in the communication tool *WinIt*.

Selectable master names	
1FP-____-	(1. Free practice)
2FP-____-	(2. Free practice)
3FP-____-	(3. Free practice)
1QP-____-	(1. Qualifying)
2QP-____-	(2. Qualifying)
LAP-____-	(Lap)
WU-____-	(Warm Up)
RACE-____-	(Race)
GP-____-	(Race)
TST-____-	(Test)
1D-____-	(Test Day 1)
2D-____-	(Test Day 2)
3D-____-	(Test Day 3)
Date-____-	(current date information)

“DATE-\_\_\_\_-” is a **special master name**. The software replaces the part “Date” of the master name by the month and day on which the measurement was downloaded. The first two characters are replaced by the month and the next two by the day (MMDD).



Your data logger device must have a defined name. If a logger is connected for the first time with the communication tool, it prompts the user to enter a valid logger name. **The logger name should be defined according to the requirements of IRTA.**



IRTA demand the last 4 characters of your logger name are “RRNN”, where “RR” is **rider initials** e.g. AA, and “NN” is **the rider number** e.g. 01.



In a new measurement, the underscores of the master names are replaced by the name of the device (letters from the back) and a correlative number is added.

**Examples:**

- If the master name “TST-\_\_\_\_-” is selected and the logger has the name “AW13” (rider initials AW and starting number 13), the first measurement name will be “TST-AW13-01.MES”.
- If the master name “DATE-\_\_\_\_-” is selected, the name of the logger is “DA77” and the measurement is downloaded on the 23<sup>rd</sup> February, the resulting measurement name for the first download is “0223-DA77-01.MES”.

**3.2.2 How to change the logger name?**

You can change the logger name in *WinIt* by direct selection + input or via the context menu (using right mouse click on selected logger node).



Click here to change your logger name!



The logger name is limited to 8 characters.

### 3.2.3 Why to change the logger name?

It was previously explained that the 2D measurement name is the combination of current master name **and** current logger name. Each underscore character (“\_”) used in the master name will be replaced by a corresponding character from the logger name (beginning from the end).

#### Examples:

If your logger name is “**Moto2**” and you use the master name “TST-\_\_\_\_-” (4x underscore characters), your first downloaded measurement will be named “TST-oto2-01”

If your logger name is “**SL22**” and you use the master name “1QP-\_\_\_\_-” (4x underscore characters), your third downloaded measurement will be named “1QP-SL22-03”

### 3.2.4 How to start a download?

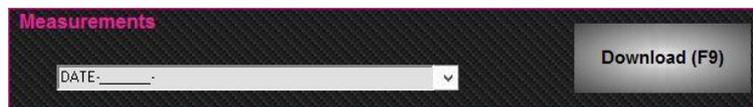
“Download” means to transfer the measurement data from a connected data logger to the PC. Make sure a new event has been created and that the current event is selected.



If the wrong event is selected in *WinARace*, your data will go to the wrong location!

There are two ways to start a download:

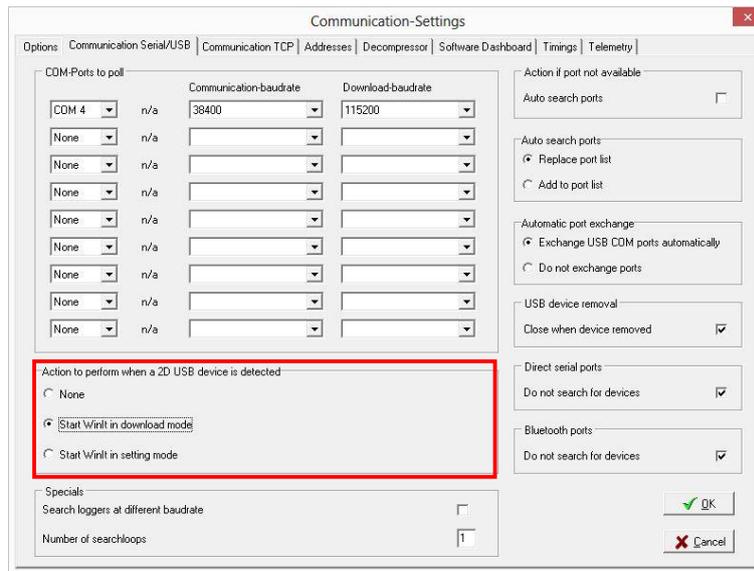
1. The first way is to start the download operation manually. Start the tool *WinARace*, connect the USB cable to the logger. Select the button **<Download (F9)>** to start the download operation



2. The second way is to perform the download automatically every time a 2D USB device is detected. This option can be set in *WinIt* (⇒ **<Options>** - **<Settings>**, “*Communication-Settings*” tab **<Communication Serial/USB>**). This is the standard pre-setting after the first software installation.



To always start the download automatically **do not uncheck** this option!



### 3.3 Data administration with the 2D program SpecView

#### 3.3.1 What is SpecView?

SpecView is a 2D program that enables you to keep accurate records of the settings used on your Moto2™ bike. When measurements are made by the data logger and transferred to your PC during a data download, the bike setting data is saved to the same location as your data download.



SpecView can be used to view the bike setting information for several measurements at once, allowing you to more easily determine the most effective settings for your bike.

#### 3.3.2 What are SpecSheet files?

SpecSheet files contain the bike setting information that is viewed (or modified) by the 2D program SpecView. SpecSheet files have the file extension “.HED”, acknowledging them to be “header” files that are used for administration of your bike data.

There are two types of SpecSheet files:

1. Permanent info files
2. SpecSheet measurement files

The **permanent info files** are a special type of header file that allows you to define the **current** bike setting and component data **before data download**. The data of the permanent info file is contained within several predefined groups e.g. rear suspension, front suspension, engine, gearbox, tires, etc.

The **SpecSheet measurement files** contain the bike setting information saved inside the permanent info file **at the time of data download**. A single SpecSheet measurement file is stored inside the measurement folder at the time of data download. Also included inside the SpecSheet measurement file are specific information about your downloaded data, e.g. lap times, logger info, weather info, user defined comments, etc.



When the measurement is being downloaded, Winlt searches for the contents of the **permanent info file**, from which it can accumulate further data for writing into the new **SpecSheet measurement file**.

### 3.3.3 What do the SpecSheet files do?

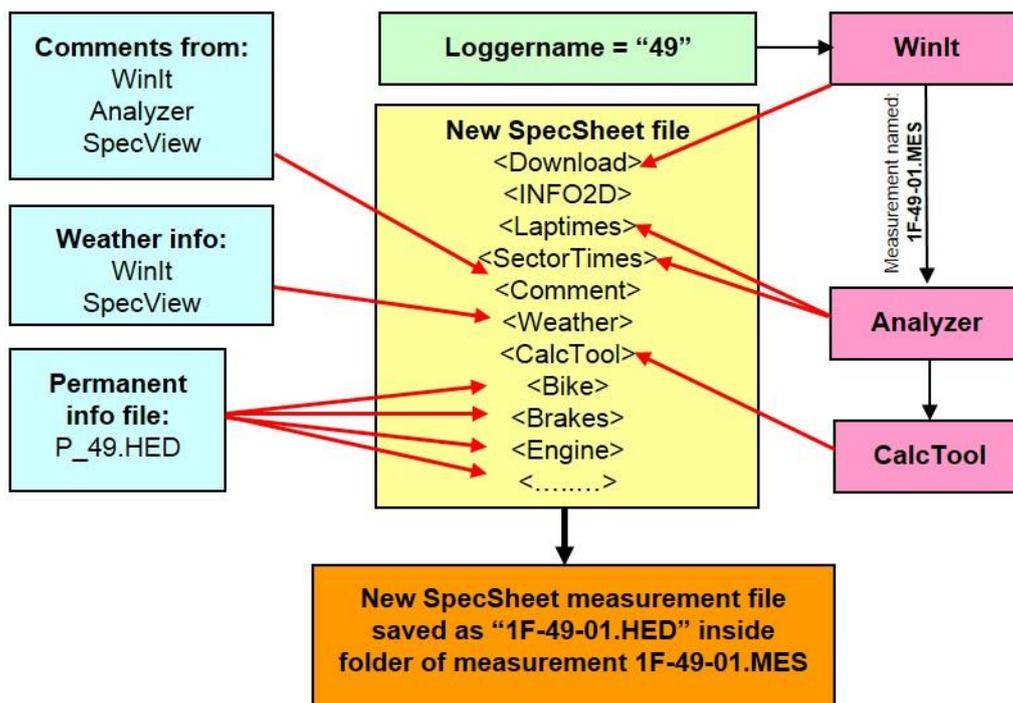
After downloading bike data, the following six default groups are **automatically generated** in the **SpecSheet** measurement file:

- **DOWNLOAD** – by *WinIt*, basic data about session (elapsed distance) and data download
- **LAPTIMES** – by *2D Analyzer*, all lap times for the measurement
- **WEATHER** – by *WinIt*, if weather information is input by engineer
- **COMMENT** – by *SpecView/ WinIt/ 2D Analyzer*, from comments inserted to ‘Quick Info’
- **CALCTOOL** – the CAL files used to make additional data channels for you to analyze
- **INFO2D** – information regarding your computer, 2D software used and your user license level



During a race/test session, the team can fill the **permanent info file** with settings **currently used by the bike**. When downloading bike data, this information will be written to the **SpecSheet** measurement file. The bike setting info will then accompany the automatic groups DOWNLOAD/LAPTIMES/WEATHER/etc.

The processes involved with the generation of *SpecSheet* files are illustrated below:



### 3.3.4 Naming of SpecSheet files

#### SpecSheet measurement files

The *SpecSheet* measurement files are always located within the folder of each downloaded measurement.

*WinIt* automatically gives the *SpecSheet* measurement file a name identical to the measurement file that is downloaded, however instead of the “.MES” extension used by the measurement files, the *SpecSheet* measurement files have the extension “.HED”, for “header”.



For example a downloaded measurement will be contained within its **own folder** named “TST-NT18-01.MES”. Inside this folder will be a *SpecSheet* measurement file with the name “TST-NT18-01.HED”.

## Permanent info files

The permanent info file must be **named by the user according to the data logger name to which it is associated**. Data logger naming was previously explained and in accordance with IRTA requirements your logger name must include rider name initials and the bike's race number, e.g. AA01.



If your data logger name is "AA01", your permanent info file must be named "*P\_AA01.HED*"

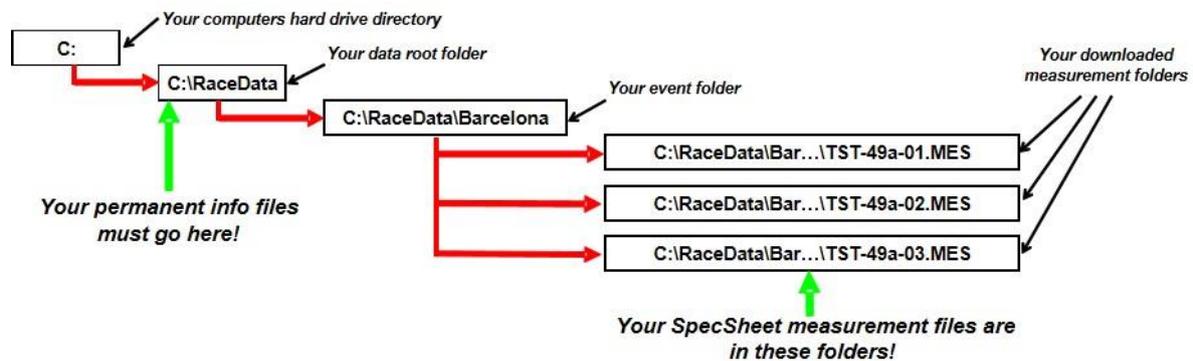
If your data logger name is "MK36", your permanent info file must be named "*P\_MK36.HED*"

This is logical because the data you enter to the permanent info file must always be valid for the bike on which the data logger is connected!

### 3.3.5 Correctly using your permanent info file

As explained previously the naming of the permanent info file is very important! It is also very important that the permanent info files are located in the correct directory of your computer!

The diagram below helps to explain the correct file locations you should use!



You should make sure that you are using a permanent info file that contains all data fields you need to keep accurate and meaningful notes of your bike's settings. 2D supplies with the Moto2™ kit a permanent info file named "*P\_Moto2.HED*". This should include all the data fields you require for your bike and if you are new to the 2D system it is recommended that you use this permanent info file.

If you are an experienced user of 2D software and feel confident to change your permanent info file, this can be most efficiently performed by opening the file with the Windows application "NOTEPAD".



You must rename "*P\_Moto2.HED*": replace "**Moto2**" with your bike's actual data logger name!

Your Moto2™ data logger is delivered as standard with the logger name "**Moto2**". To match the logger name, the permanent info file supplied with your Moto2™ software installation is named "*P\_Moto2.HED*".

When you change the name of your logger (to match your bike number for example), *WinIt* should prompt you to advise that the **permanent info file** must also attain a new name.



The renaming of your permanent info file should occur automatically when you change the logger name. However it is still advised that you **double-check that the permanent info file has the correct naming AND location** before you make the download of a measurement!

As a test/practice/qualifying session progresses and multiple downloads are made, **it is crucial that the data technician continually updates (and saves) the permanent info file when bike settings/conditions change.**

Maintaining good records of settings used during each session provides an organized and efficient system for quickly viewing bike setup information.

### 3.3.6 SpecView and using SpecSheet files

More information about *SpecView* is provided here, including:

- Using the mileage function
- Updating entries of the permanent info file
- Printing your *SpecSheet* data



For more comprehensive explanation of the 2D program *SpecView*, more documentation can be attained on [www.2d-datarecording.com](http://www.2d-datarecording.com):

<support> - <download> - <manuals> - <SpecView>

#### 3.3.6.1 The mileage function

*SpecView* includes a function that enables the number of kilometers completed by the bike to be recorded in the *SpecSheet* file for the team's records.

The mileage function is activated by entering the notation “(nkm)” into a particular field of the permanent info file, e.g. brake disc. When preparing the permanent info file, “n” should be replaced by the number of kilometers this component has so far been used up to the beginning of the events first session. The “*mileage notation*” does not have to be the only entry in that field, as beside the mileage function the specification of the component can also be entered e.g. “**Spec1 (0km)**”.

After the first session of the event, your measurement is downloaded from the data logger using the program *WinIt*. After the download is complete *WinIt* performs some special tasks to make sure the distance travelled by the bike is added to your records.



At the time of data download many operations occur that help correctly administrate your bike setting information and also the distance covered by your bike during the measurement!

After downloading a measurement, *WinIt* performs the following actions:

- generates a new *SpecSheet* file to contain documentation data for that measurement
- writes to the *SpecSheet* file the group <Download>, containing the distance travelled
- locates the permanent info file defined for that data logger
- locates in the permanent info file all the fields containing the mileage function “(n km)”
- adds the distance travelled in that session to the each “n” value of the permanent info file
- saves the permanent info file so the modified mileage values will exist for the next download
- transfers all group names, field names, entry values and mileage data from the permanent info file to the new *SpecSheet* file
- saves the new *SpecSheet* measurement file to the measurement directory with the file extension **.HED**

**Example:**

Looking at the group “Brakes” from the permanent info file supplied by 2D. The column “pre event” shows the values that are entered in the permanent info file before the first download is made. The fields *FDISC*, *FPAD*, *RDISC* and *RPAD* all have the mileage function included and each have the start mileage value “(0 km)”, meaning the fitted components are unused.

	pre event	download 1: 15 km session	download 2: 7 km session
<b>[BRAKES]</b>			
FM/C	XYZ	XYZ	XYZ
FDISC	XYZ (0 km)	XYZ (15 km)	XYZ (22 km)
FPAD	XYZ (0 km)	XYZ (15 km)	XYZ (22 km)
FCALIPER	XYZ (standard)	XYZ (standard)	XYZ (standard)
RM/C	XYZ	XYZ	XYZ
RDISC	XYZ 190 mm (0 km)	XYZ 190 mm (15 km)	XYZ 190 mm (22 km)
RPAD	XYZ (0 km)	XYZ (15 km)	XYZ (22 km)
RCALIPER	24 mm	24 mm	24 mm



If the bike completes a total of **15 km** in the first measurement, all the mileage values of the permanent info file will therefore be updated and have a value of “(15 km)” after the download is completed. This is shown in the above table under the column “**download1: 15 km session**”.

Next the bike leaves the pit box (with no modifications made to the brakes) and completes a further 7km on the track. The measurement download is completed and the permanent info data is as shown in column “**download2: 7 km session**”.

This will continue to occur as data is downloaded for the bike, until a component is replaced.

If new front brake pads are fitted, you must modify the mileage value for the permanent info file entry “*FPAD*” and set this to “(0 km)” **before** the next measurement download is completed! If the fitted brake pads are not 100% new, the team must estimate what value of mileage to make for that component inside the *SpecSheet*.



The mileage for any component can be reset by clicking the icon  in the *WinARace* toolbar.

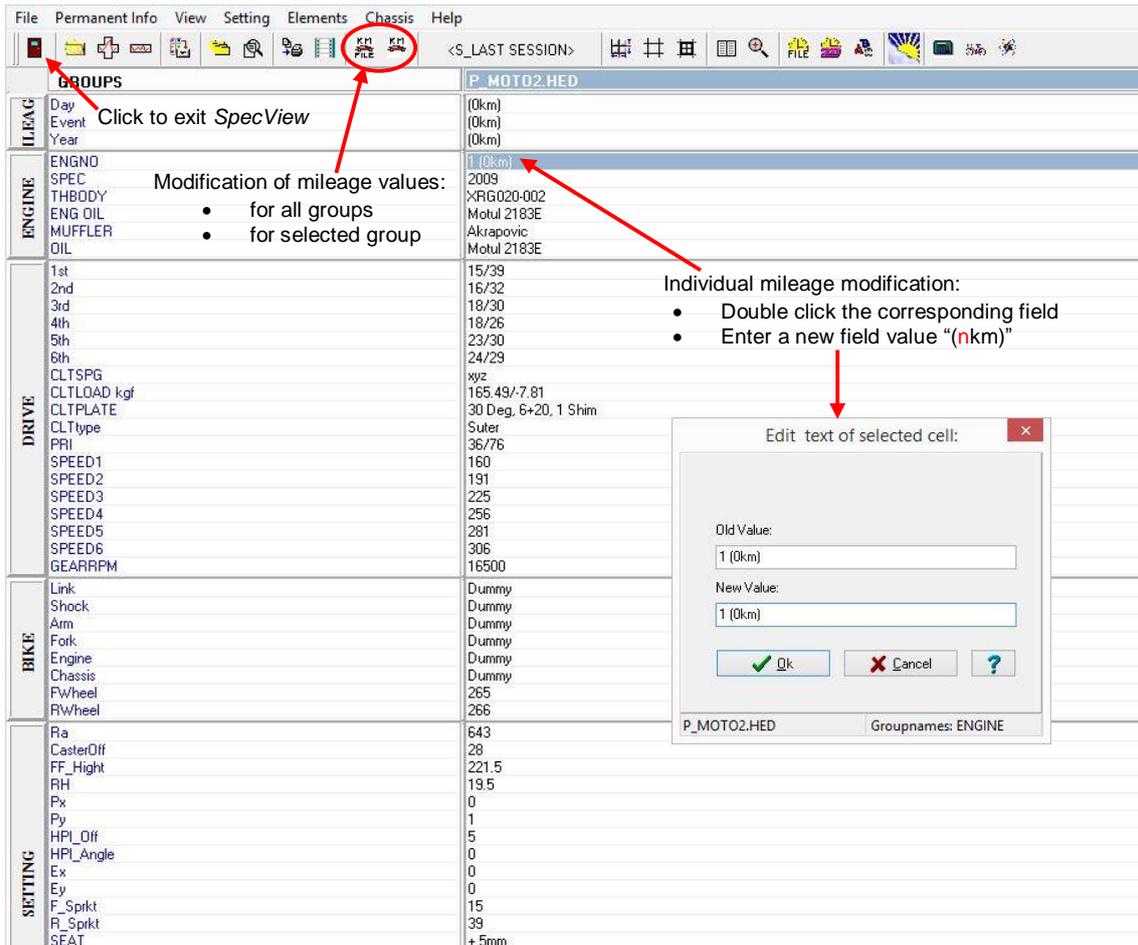
Pressing this <P> *SpecSheet* icon automatically opens all the permanent info files within the selected event folder using *SpecView*.

Using *SpecView* the mileage values can be amended for each field where necessary.



It is critically important that the correct permanent info file is updated, i.e. update only the file that operates with the bike to which the component modifications occur!

Be sure to save your changes to the permanent info file **before the next download** as any updated mileage or entry values will not be applied to the next *SpecSheet* measurement file if they are not already saved to the permanent info file.



Click to exit *SpecView*

Modification of mileage values:

- for all groups
- for selected group

Individual mileage modification:

- Double click the corresponding field
- Enter a new field value "(n)km"

GROUPS	P_MOTO2.HED
ILEAG	(0km)
Day	(0km)
Event	(0km)
Year	(0km)
ENGINE	1 (0km)
ENGNO	2009
SPEC	XRG020-002
THBODY	Motul 2183E
ENG OIL	Akrapovic
MUFFLER	Motul 2183E
OIL	
DRIVE	
1st	15/39
2nd	16/32
3rd	18/30
4th	18/26
5th	23/30
6th	24/29
CLTSPG	xyz
CLTLOAD kgf	165.49/-7.81
CLTPLATE	30 Deg, 6+20, 1 Shim
CLTtype	Sutler
PRI	36/76
SPEED1	160
SPEED2	191
SPEED3	225
SPEED4	256
SPEED5	281
SPEED6	306
GEARRPM	16500
BIKE	
Link	Dummy
Shock	Dummy
Arm	Dummy
Fork	Dummy
Engine	Dummy
Chassis	Dummy
FWheel	265
RWheel	266
SETTING	
Ra	643
CasterOff	28
FF_Hight	221.5
RH	19.5
Px	0
Py	1
HPI_Off	5
HPI_Angle	0
Ex	0
Ey	0
F_Sprkt	15
R_Sprkt	39
SEAT	+ 5mm



Mileage modifications can be made automatically to **all groups** or for only the **selected group**.

In the case of a single component being changed, the mileage must be modified by double-clicking the field that must be changed, and entering the new mileage value that should exist in that field (as shown above). Once the correct changes have been made to the permanent info file, click the icon labelled above to exit *SpecView*.



When exiting *SpecView* you will be prompted to save or not save the file modifications made. If you are sure that the changes made are 100% correct, click yes. This will update the permanent info file and all mileage values are correct for the next measurement data to be downloaded.

When your team arrives at a new event and prepares the motorbike, the component mileages after the previous event are known as they are stored in the permanent header files from that event (for each bike). This provides the starting point for the new permanent info files to be prepared for the beginning of the next event.

When considering the mileage function and its benefits, the reasons for needing a different permanent info file for each bike (and the data loggers fitted to them) becomes obvious! With two different permanent info files, the mileage values used by one bike are totally independent of the other bike.



By carefully maintaining the mileage values entered to the permanent info files, the number of kilometers accumulated by many bike components can be accurately measured across the entire season.

The following mileage functions are supplied inside the 2D permanent info file:

- FDISC – front brake disc – group “BRAKES”
- FPAD – front brake pad – group “BRAKES”
- RDISC – rear brake disc – group “BRAKES”
- RPAD – rear brake pad – group “BRAKES”
- ENGNO – engine mileage – group “ENGINE”
- Day – mileage for that day – group “Mileage”
- Event – mileage for that event – group “Mileage”
- Year – mileage for that year – group “Mileage”
- FTIRE – front tire – group “WHEELS”
- RTIRE – rear tire – group “WHEELS”



If more mileage functions are needed, they can be assigned to other fields of the permanent info file. Be careful as including too many mileage functions makes the job of keeping each field correctly updated more complicated!

### 3.3.6.2 Default permanent info file for Moto2™

It is very important that the layout and content of the permanent info file is correctly matched to the settings actually used on the bike. To help teams get started more quickly, 2D has prepared a permanent info file that should provide a good **starting point** for using the Moto2™ kit system.

The data content of the file is listed under the columns “Heading” and “Value” in the table below:

Heading	Value	Explanation	Additionally used for...
<b>[BIKE]</b>			
FRAME	Dummy	Title for group containing bike component IDs Team ID for frame that is used	
SWGARM	Dummy	Team ID for swingarm that is used	
Link	Dummy	Team ID for shock link that is used	
Shock	Dummy	Team ID for shock that is used	
Arm	Dummy	Team ID for shock arm that is used	
Fork	Dummy	Team ID for fork that is used	
Engine	Dummy	Engine ID that is used	
Chassis	Dummy	Team ID for frame that is used	
Fwheel	265	Team ID for front wheel that is used	
Rwheel	266	Team ID for rear wheel that is used	
<b>[BRAKES]</b>			
FM/C	18-18	Front brake master cylinder spec/ setting	
FDISC	XYZ (0km)	Front disc spec + (mileage function)	To track bike mileage
FPAD	XYZ (0km)	Front brake pad spec/ manufacturer + (mileage function)	To track bike mileage
FCALIPER	XYZ (standard)	Front brake caliper spec	
RM/C	XYZ	Rear brake master cylinder spec/ setting	
RDISC	XYZ 190mm (0km)	Rear disc spec + (mileage function)	To track bike mileage
RPAD	XYZ (0km)	Rear brake pad spec/ manufacturer + (mileage function)	To track bike mileage
RCALIPER	24mm	Rear brake caliper spec	

<b>[DRIVE]</b>		Title for group containing transmission info	
1st	15/39	1 <sup>st</sup> gearbox ratio (fixed)	Gearing calculations in <i>CalcTool</i>
2nd	16/32	2 <sup>nd</sup> gearbox ratio (fixed)	Gearing calculations in <i>CalcTool</i>
3rd	18/30	3 <sup>rd</sup> gearbox ratio (fixed)	Gearing calculations in <i>CalcTool</i>
4th	18/26	4 <sup>th</sup> gearbox ratio (fixed)	Gearing calculations in <i>CalcTool</i>
5th	23/30	5 <sup>th</sup> gearbox ratio (fixed)	Gearing calculations in <i>CalcTool</i>
6th	24/29	6 <sup>th</sup> gearbox ratio (fixed)	Gearing calculations in <i>CalcTool</i>
CLTSPG	XYZ	Clutch spring spec	
CLTLOAD kgf	165.49/-7.81	Clutch loading settings	
CLTPLATE	30Deg, 6+20, 1Shim	Clutch plate settings	
CLTtype	Suter	Clutch spec/ manufacturer	
PRI	36/76	Engine primary drive ratio (fixed)	Gearing calculations in <i>CalcTool</i>
<b>[ENGINE]</b>		Title for group containing engine info	
ENGNO	1 (0km)	Official HRC engine number + (mileage function)	To track bike mileage
SPEC	2009	Engine spec	
THBODY	XRG020-002	Throttle body spec	
ENG OIL	Motul 2183E	Engine oil spec	
Muffler	Akrapovic	Exhaust muffler spec	
OIL	Motul 2183E	Gearbox oil spec	
<b>[FORKSET]</b>		Title for group containing forkset setting info	
SPEC	XYZ (no.9)	Forkset spec	
Spring	9.25	Fork spring stiffness (N/mm)	Calculations in chassis program
Preload	3	Fork spring preload displacement (mm)	Calculations in chassis program
TopSpring	1	Top fork spring stiffness (N/mm)	Calculations in chassis program
Topend	135	Top fork spring preload displacement (mm)	Calculations in chassis program
OilLevel	205	Forkset oil level (mm)	Calculations in chassis program
EXT	14	Forkset extension damping setting	Calculations in chassis program
COM	14	Forkset compression damping setting	Calculations in chassis program
<b>[MASS]</b>		Title for group containing bike/ rider masses	
Bike	150	Total bike mass	Power calculations in <i>CalcTool</i>
Rider	75	Total rider mass	Power calculations in <i>CalcTool</i>
FWheel	17	Front wheel mass (with disc)	Power calculations in <i>CalcTool</i>
RWheel	17	Rear wheel mass (with disc)	Power calculations in <i>CalcTool</i>
<b>[Mileage]</b>		Title for group containing bike mileage functions	
Day	(0km)	Distance travelled so far on day (km)	To record bike mileage
Event	(0km)	Distance travelled so far in event (km)	To record bike mileage
Year	(0km)	Distance travelled so far in year (km)	To record bike mileage
<b>[SETTING]</b>		Title for group containing bike setting and geometry info	
Ra	643	Swingarm length	Calculations in chassis program
CasterOff	28	Steering offset distance (mm)	Calculations in chassis program
FF_Hight	221.5	Front ride height setting (mm)	Calculations in chassis program
RH	19.5	Rear ride height setting (mm)	Calculations in chassis program
Px	0	Swingarm pivot x-axis position (along bikes length) (mm)	Calculations in chassis program
Py	1	Swingarm pivot y-axis position (along bikes height) (mm)	Calculations in chassis program
HPI_Off	5	Steering bearing offset (mm)	Calculations in chassis program
HPI_Angle	0	Steering bearing angle adjustment (deg)	Calculations in chassis program
Ex	0	Engine position x-axis (along bikes length) (mm)	Calculations in chassis program
Ey	0	Engine position y-axis (along bikes height) (mm)	Calculations in chassis program
F_Sprkt	15	Front drive sprocket (teeth)	To calculate $V_{rear}$ in <i>CalcTool</i>
R_Sprkt	39	Rear drive sprocket (teeth)	To calculate $V_{rear}$ in <i>CalcTool</i>
SEAT	5	Seat position/ padding (mm)	
Handlebar	XYZ	Handlebar position setting	
EngPos	0 (STD)	Engine position spec	
ForceDim	N	Nominated force dimension	Calculations in chassis program
Footrest Posn	0	Footrest position setting	
ST-DAMP	11	Steering damper setting	

<b>[SHOCKSET]</b>		Title for group containing shockset setting info	
Spec	XYZ (no.1)	Shockset spec	
Spring	87.1	Shock spring stiffness (N/mm)	Calculations in chassis program
Preload	12	Shock spring preload displacement (mm)	Calculations in chassis program
TopSpring	125	Shock topspring stiffness (N/mm)	Calculations in chassis program
Topend	12	Shock topspring preload displacement (mm)	Calculations in chassis program
EXT	14	Shock extension damping setting	Calculations in chassis program
COM	12	Shock compression damping setting	Calculations in chassis program
<b>[WHEELS]</b>		Title for group containing tire and wheel info	
FTYRE	DF (0km)	Front tire spec + (mileage function)	To track tire mileage
RTYRE	DR (0km)	Rear tire spec + (mileage function)	To track tire mileage
FCIRC	1925	Front tire circumference (mm)	
RCIRC	2110	Rear tire circumference (mm)	To calculate V_rear in CalcTool
F-SIZE	125/600-16.5	Front tire dimensions	
R-SIZE	190/650-16.5	Rear tire dimensions	
FRIM	3.75x16.5	Front rim dimensions	
RRIM	6.25x16.5	Rear rim dimensions	
Fpress	1.9	Front tire pressure (bar)	
Rpress	1.8	Rear tire pressure (bar)	

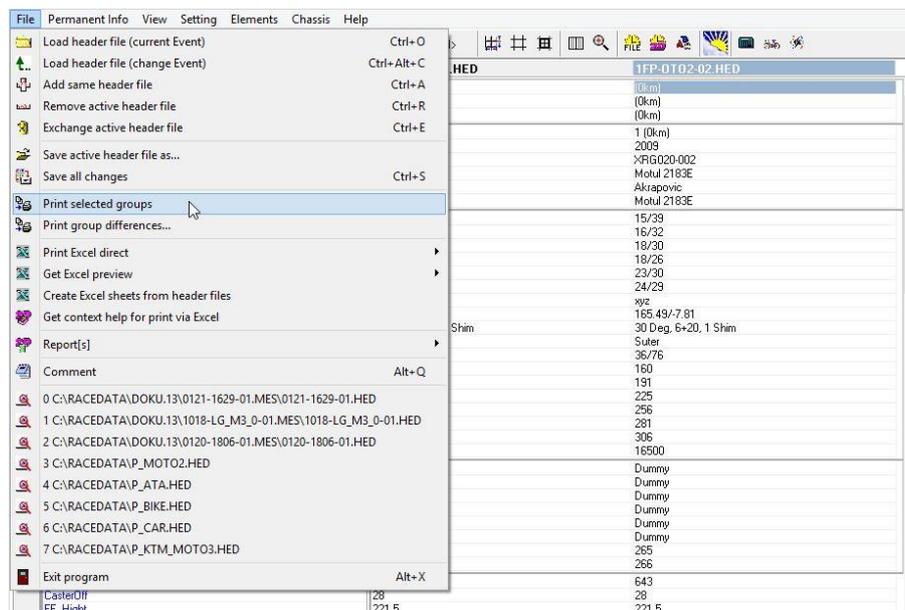
### 3.3.6.3 Printing SpecSheet data

If you are viewing one or more *SpecSheet* files using *SpecView*, it is possible to print the data out for easier reading, or to have a “hardcopy”.

Two options exist for printing the data:

- Go to: *File* ⇒ *Print selected groups*, user selects groups to print, all data will be printed
- Go to: *File* ⇒ *Print group differences*, user selects groups to print, only data changed from first measurement will be printed

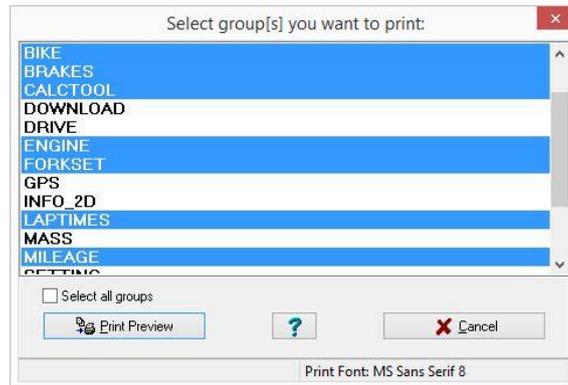
Alternatively the *SpecView* hotkey <F3> can be used, prompting you to choose from either of the two options explained above. The “*Print group differences*” option is very useful if you are attempting to observe the changes made to your motorcycle specification during the course of an entire session or even a race weekend. The menu you select from is shown in the image below.



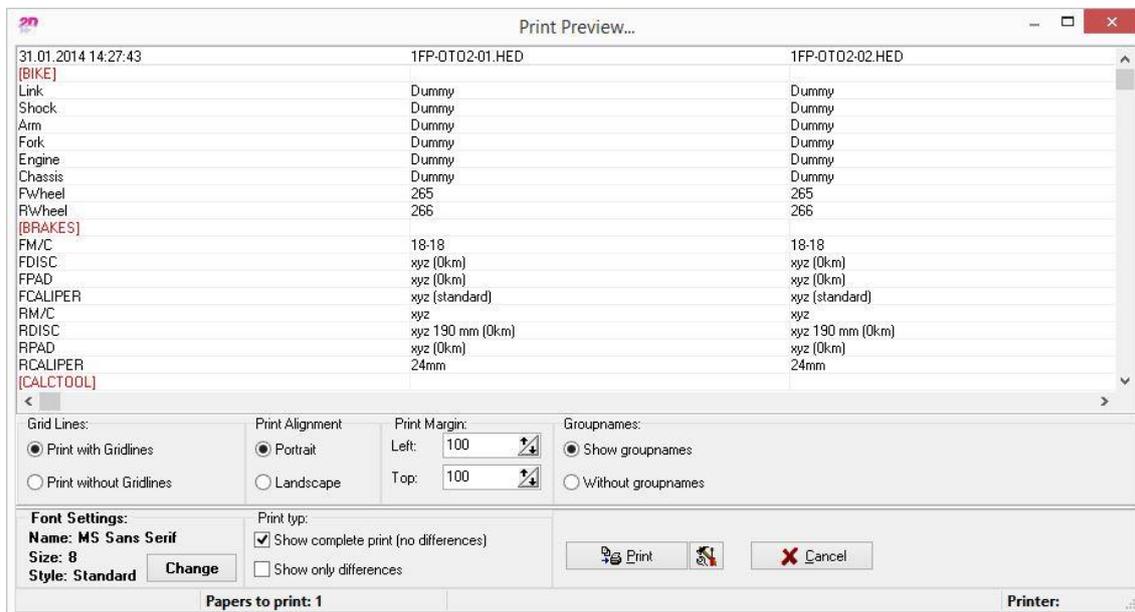
After you have selected the type of printing you want to perform, you can then define the groups that will be printed by *SpecView*. This makes sure that only the data you want is printed.

All groups can be automatically selected by checking the box “*Select all groups*”. Alternatively you can hold the <CTRL> key of your PC and individually select/deselect the groups you want to print.

Once you are happy with the selection click <Print Preview>.



The print preview will give you a chance to view the data before it is printed, here various settings can be adjusted. Many options exist for you to customize the layout of the printing. For example, printing with/without gridlines, print as portrait/landscape, font settings, print margin, and print group names/don't print group names.



### 3.3.6.4 Exporting SpecSheet data

Instead of actually printing the *SpecSheet* data onto paper, it can be exported to a predefined *Excel* spreadsheet and saved to the hard drive of your PC/Server for secure storage. From that location the information can be viewed, emailed or printed for personal records. With the Moto2™ software, 2D supplies some *Excel* templates that can be used by the team to export their *SpecSheet* data.

A template has been designed especially for Moto2™ and is named “S\_MOTO2.XLS”.



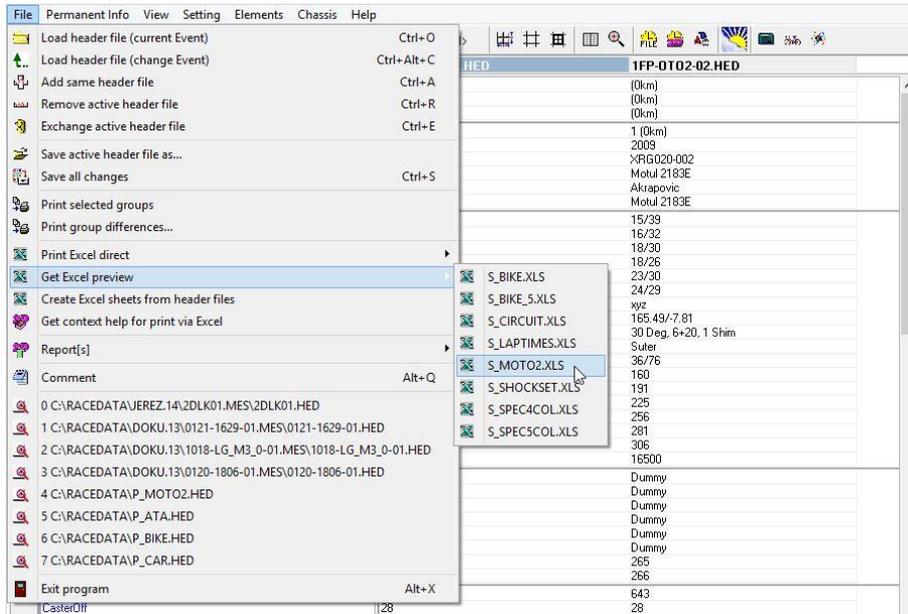
All templates must be stored in the directory, <C:\ProgramData\RaceMoto2\_14.0\Excel Sheets\SpecView>, you reach this folder by selecting *Settings* ⇒ *Folders – Protocol* in the *WinARace* menu. There you select the button <Application data>. In the opened folder you select <Excel Sheets>, <SpecView>.

The template “S\_MOTO2.XLS” provides a good starting point for the team. It is designed to function correctly with *SpecSheet* data files that are generated using the permanent info file supplied by 2D (as previously introduced). The function of the *SpecSheet* export and the interactions between the

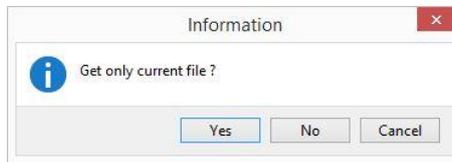
*SpecSheet* and the *Excel* template will be described briefly using the templates provided by 2D. This should provide more understanding and enable you to construct your own *Excel* templates!

Using *SpecView* the *SpecSheet* data can be exported using one of the following options:

- Select *File, Print Excel direct, S\_MOTO2.XLS* ⇒ saves to *Excel* template directly
- Select *File, Get Excel preview, S\_MOTO2.XLS* ⇒ opens *Excel* template to preview data



Whatever you select, the following prompt will be displayed:



This is asking if you want to export only the selected *SpecSheet* file to the *Excel* template you have selected. The currently selected file is shown in *SpecView* with its column title highlighted in blue.

If you are happy to export only the selected *SpecSheet* file, click <Yes>. If you want all files currently open in *SpecView* to be exported, click <No>. Otherwise click <Cancel>.

Below the “S\_MOTO2.XLS” *SpecSheet* export template is shown before and after the export is completed.

2D Debus & Diebold Meßsysteme GmbH		Alte Karlsruher Str. 8 76227 Karlsruhe	
Tel.: +49(0) 721 944850		Fax: +49(0) 721 9448529	
<b>Session</b>			
EVENT:	0	Comment	
MEASUREMENT:	0	0	
DATE:	00/01/1900 00:00	0	
<b>Weather</b>			
Ambient Conditions	0	Bike	
Air Temp (deg C)	0	Frame	0
Air Press (mbar)	0	Swing Arm	0
Track Temp (deg C)	0	Link	0
Wind Speed (m/s)	0	Shock	0
Wind Direction	0	Arm	0
<b>Setting</b>			
Ra	0	Fork	0
Caster Off	0	Engine	0
FF Height	0	Chassis	0
RH	0	Front Wheel	0
Px	0	Rear Wheel	0
Py	0	<b>Forkset</b>	
HPI Off	0	Spec	0
HPI Angle	0	Spring	0
Ex	0	Preload	0
Ey	0	Top Spring	0
Front Sprocket	0	Top End	0
Rear Sprocket	0	Oil Level	0
Seat	0	EXT	0
Handlebar	0	COM	0
Engine Position	0	<b>Shockset</b>	
Footrest Pos	0	Spec	0
Steering Damper	0	Spring	0
<b>Engine</b>			
Engine No. +KM	0	Preload	0
Spec	0	Top Spring	0
Throttle Body	0	Top End	0
Eng Oil	0	EXT	0
Muffler	0	COM	0
Oil	0	<b>Brakes</b>	
<b>Clutch</b>			
Spring	0	Front Master Cyl	0
Load	0	Front Disc +KM	0
Plate	0	Front Pad +KM	0
Type	0	Front Caliper	0
<b>Wheels</b>			
Front Tyre +KM	0	Rear Master Cyl	0
Rear Tyre +KM	0	Rear Disc +KM	0
Front Tyre Circ	0	Rear Pad +KM	0
Rear Tyre Circ	0	Rear Caliper	0
Front Tyre Size	0		
Rear Tyre Size	0		
Front Rim	0		
Rear Rim	0		
Front Tyre Pressure	0		
Rear Tyre Pressure	0		

Empty Excel file

2D Debus & Diebold Meßsysteme GmbH		Alte Karlsruher Str. 8 76227 Karlsruhe	
Tel.: +49(0) 721 944850		Fax: +49(0) 721 9448529	
<b>Session</b>			
EVENT:	VALENCIA	Comment	
MEASUREMENT:	2D-TO2-08 HED	0	
DATE:	10.11.2009 15:59:45	0	
<b>Weather</b>			
Ambient Conditions	DRY	Bike	
Air Temp (deg C)	23/-/50%	Frame	Dummy
Air Press (mbar)	1002	Swing Arm	Dummy
Track Temp (deg C)	35	Link	Dummy
Wind Speed (m/s)	0	Shock	Dummy
Wind Direction	Head	Arm	Dummy
<b>Setting</b>			
Ra	0	Fork	Dummy
Caster Off	643	Engine	Dummy
FF Height	28	Chassis	Dummy
RH	221.5	Front Wheel	265
Px	19.5	Rear Wheel	266
Py	0	<b>Forkset</b>	
HPI Off	1	Spec	0
HPI Angle	5	Spring	xyz (No. 9)
Ex	0	Preload	9.25
Ey	0	Top Spring	3
Front Sprocket	0	Top End	1
Rear Sprocket	15	Oil Level	135
Seat	39	EXT	205
Handlebar	+ 5mm	COM	14
Engine Position	xyz	<b>Shockset</b>	
Footrest Pos	N	Spec	0
Steering Damper	0	Spring	xyz (No. 1)
<b>Engine</b>			
Engine No. +KM	1 (0km)	Preload	87.1
Spec	2009	Top Spring	12
Throttle Body	KRG020-002	Top End	125
Eng Oil	0	EXT	12
Muffler	Motul 2183E	COM	14
Oil	Acrapovite	<b>Brakes</b>	
<b>Clutch</b>			
Spring	xyz	Front Master Cyl	18-18
Load	165.49/-7.81	Front Disc +KM	xyz (0km)
Plate	30 Deg, 6+20, 1 Shim	Front Pad +KM	xyz (0km)
Type	Suter	Front Caliper	xyz (standard)
<b>Wheels</b>			
Front Tyre +KM	DF (0km)	Rear Master Cyl	xyz
Rear Tyre +KM	DR (0km)	Rear Disc +KM	xyz 190 mm (0km)
Front Tyre Circ	1925	Rear Pad +KM	xyz (0km)
Rear Tyre Circ	2110	Rear Caliper	24mm
Front Tyre Size	125/600-16.5		
Rear Tyre Size	190/650-16.5		
Front Rim	3.75 X 16.5		
Rear Rim	6.25 X 16.5		
Front Tyre Pressure	1.9		
Rear Tyre Pressure	1.8		

Exported SpecSheet data

The left one shows the template before it is used to contain export data. It can be opened from its directory <C:\ProgramData\RaceMoto2\_14.0\Excel Sheets\SpecView><sup>2</sup> and modified according to the team's requirements.

The image on the right shows the *SpecSheet* template after data is properly copied to it. Now present on the sheet is measurement information, bike settings from *SpecSheet* data and an image of the track map from the measurement. If "Get Excel preview" was selected, the data-filled template is automatically opened to be viewed or modified before the final file name and save directory are defined by the user in the usual way for the *Excel* program.



Make sure you do not overwrite the original template file!

### 3.3.6.5 Further explanation of the SpecSheet export

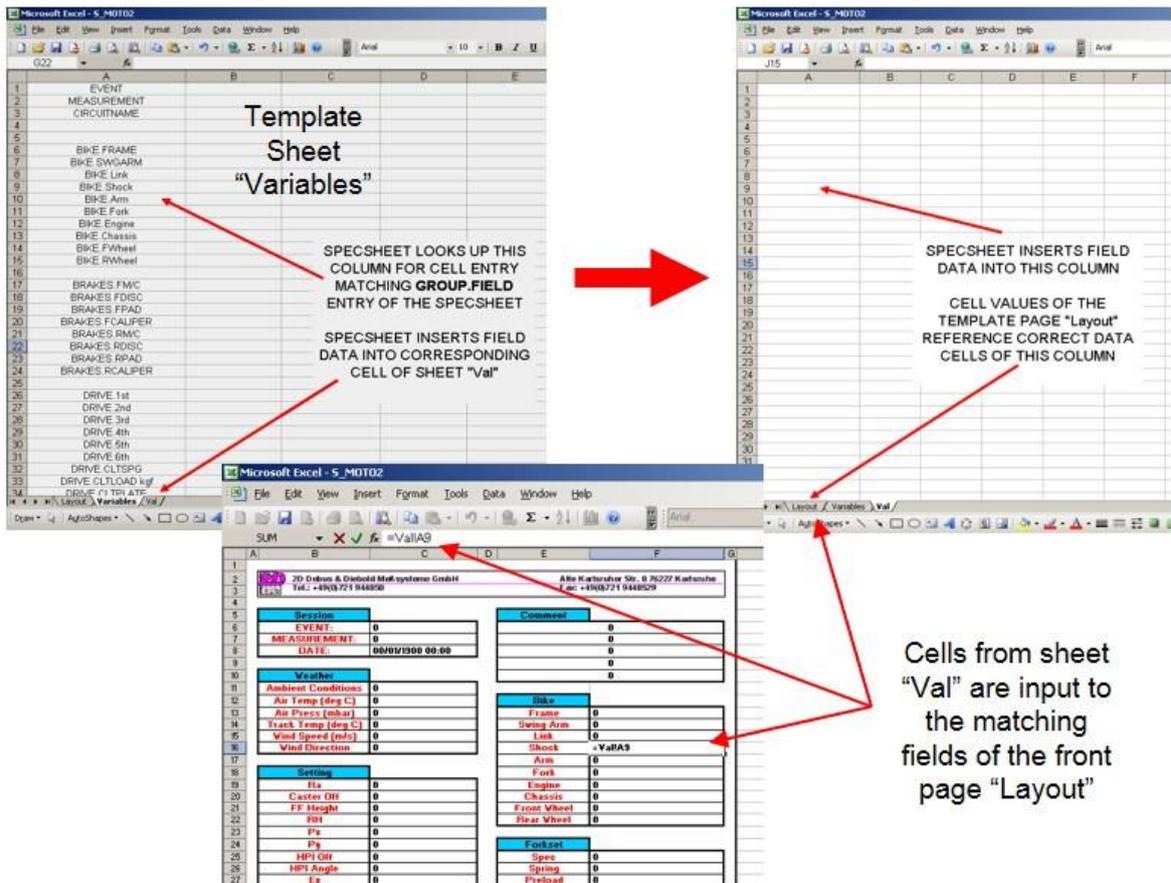
Here the basic operation of the *SpecSheet*-to-*Excel* export will be explained. When the *SpecSheet* data is exported to the *Excel* template, the following steps will be performed:

- *SpecView* begins with the first *SpecSheet* file (on left side of *SpecView* window)
- *SpecView* looks up the *SpecSheet* template that has been selected for the export
- *SpecView* searches column A of the template's "Variables" tab, seeking cell entries that match the group and field entries from the first *SpecSheet* file opened in *SpecView*
- If in row 1 a valid cell entry matches an entry from the *SpecSheet* file, the input values from that entry are pasted into cell A1 of the next tab "Val"
- *SpecView* returns to the "Variables" tab continues searching all rows in column A, repeating the above when a matching entry is found

<sup>2</sup> You reach this folder by selecting *Settings* ⇒ *Folders – Protocol* in the *WinARace* menu. There you select the button <Application data>. In the opened folder you select <Excel Sheets>, <SpecView>.

- Once all rows have been completed, *SpecView* will move to the next open *SpecSheet* file and begin the same process, only this time using column B. This continues for the remaining *SpecSheet* files

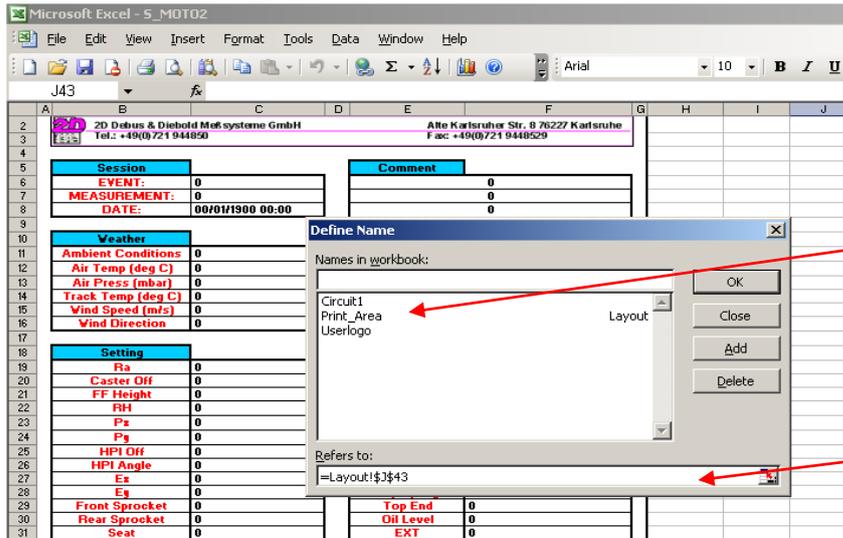
By correctly defining the cell values of tab “*Variables*”, the data from the open *SpecSheet* files can be quickly transferred to the *SpecSheet* export template. The entered values must strictly follow the convention, <groupname>.<field>. Therefore to get data from the group “*Bike*” and field “*Chassis*”, the entry “**Bike.Chassis**” must be entered to the desired row of the template, and into tab “*Variables*”. If this is carefully performed for each *SpecSheet* field, it is possible to import an entire *SpecSheet* file to the “*Val*” tab of the *Excel* template.



It is then possible to define the appearance of the template; this is located in the tab “*Layout*”. Here the printed layout of the document is defined and the information imported to the tab “*Val*” can be assembled here for a good-looking data sheet that can be printed.

A well-organized “*Layout*” tab can be defined by correctly assigning the group and field names, then entering links so that the values on the front end are equal to the correct values present in the tab “*Val*”. For example to make any cell in the front-end have an identical value to cell A9 of the tab “*Val*”, select the cell in the front end and type “=Val!A9”. By carefully administrating the links between tabs “*Layout*” and “*Val*”, a highly customized *SpecSheet* export template can be defined.

Finally the user can define some important areas of the export template. This makes it possible to include your team’s logo, the race circuit map and to permanently assign the desired print area. To do this select from *Excel*: <Insert, Name, Define>.



Names assigned by 2D template

Assign page area for selected name

It is important that when making a new sheet, the names defined for that sheet match the ones used in the template supplied by 2D. For the circuit image to be inserted it must be named "Circuit1" and also have a valid and practical page area assigned. If a user logo is defined in the program *WinIt*, this can also be included in the templates "Layout" page, but only if the settings are correctly defined!

### 3.4 Summary of measured channels

The stages for generation of data channels inside the Moto2™ data logger were previously shown by a flow diagram. Here the exact meaning and function of each data channel will be explained.

#### Susp\_F

The relative position of the front suspension. This measurement is made by a 150 mm 2D linear potentiometer mounted to the front fork of the bike. The zero position must be set before you operate the bike.

#### Susp\_R

The relative position of the rear suspension. This measurement is made by a 75 mm 2D linear potentiometer mounted either directly onto the damper (shock absorber) or to the swing-arm at an appropriate position. This channel requires a full calibration (min/max/zero defined) before you operate the bike.

#### Brake\_F

The pressure contained inside the hydraulic circuit of the front brake system. This measurement is made by a 100 bar 2D pressure sensor and has units of bar. The zero position must be set before you operate the bike.

#### Brake\_R

The pressure contained inside the hydraulic circuit of the rear brake system. This measurement is made by a 100 bar 2D pressure sensor and has units of bar. The zero position must be set before you operate the bike.

#### Volt\_1 and Volt\_2 (Acc\_Fax and Acc\_Rax)

Two additional 5 V analog input channels. While it is possible to connect various different 0-5 V sensors to these channels, the Moto2™ kit software has been configured to accept the connection of +/- 10 G 2D accelerometers into Volt\_1 and Volt\_2. At the time of download, the Moto2™ kit software takes the voltage values of Volt\_1 and Volt\_2 and creates the new channels named "Acc\_Fax" and "Acc\_Rax".

Users of the **standard Moto2™ kit software** can still view the measured **voltage values** for the sensors connected to Volt\_1 and Volt\_2; this is done in the normal way using the program *2D Analyzer*. These channels enable the connection of various different types of 2D sensor e.g. steering angle, air pressure,

etc. For the highest accuracy of measurement, the connected sensor should deliver a voltage output ranging from 0 V to 5 V across the intended physical measurement range.

**By making your own additional CAL file (defined using Full/Net/Team software license)** the correct physical values for the connected sensor can be correctly displayed in *2D Analyzer* (post download calibration).

Examples of CAL files for Volt\_1 and Volt\_2 are explained in section 4.4.1.3.

### Gyro

The banking rate of the bike as it is leaning into the corner. This measurement is made by a +/- 200 deg/s 2D gyroscope and has units of deg/s.

### P\_Oil

The pressure inside the engine's oil lubrication system. This measurement is made by a 10 bar 2D pressure sensor that is specially calibrated to operate with high accuracy at elevated temperatures (from 20°C to 120°C). This channel requires the zero pressure to be set before you operate the bike.

### Vext

A measurement of the power supply received by the data logger. The units of this channel are Volts and no external sensor is required for the measurement as it is made inside the data logger.

### ACC\_X

The acceleration of the data logger along its lateral axis (side to side). This measurement is expressed in units of G and made inside the data logger by an internally mounted +/- 5 G accelerometer. This channel requires the zero value to be set before you operate the bike.

### ACC\_Y

The acceleration of the data logger along its longitudinal axis (along its length). This measurement is expressed in units of G and made inside the data logger by an internally mounted +/- 5 G accelerometer. This channel requires the zero value to be set before you operate the bike.

### ACC\_Z

The acceleration of the data logger along its vertical axis (up and down). This measurement is expressed in units of G and made inside the data logger by an internally mounted +/- 5 G accelerometer. This channel requires the zero value to be set before you operate the bike.

### Temp

The internal temperature of the data logger. This measurement is expressed in units of degrees Celsius and made inside the data logger by an internally mounted and linearized thermo resistor.

### V\_Front

The rotation speed of the front wheel. This measurement is made by a digital sensor fitted to the front of the bike and generates electrical "pulses" each time the wheel is rotated. The number of pulses for each wheel rotation depends on the number of activating devices fitted to the wheel. Input of the tire perimeter (in mm) and the number of pulses for each wheel rotation enable the speed of the ground under the wheel to be calculated. This channel has units of km/h.



You must correctly input the perimeter of your front tire and input the pulses per wheel rotation before an accurate *V\_Front* measurement will be made.

### RPM

The engine's crankshaft rotation speed. Your Moto2™ engine is already supplied with a digital sensor inside. The 2D data logger measures the frequency of crankshaft rotations and converts to RPM for recording.

**RPMSprkt**

The gearbox output shaft rotation speed for the engine. Your Moto2™ engine is already supplied with a digital sensor inside. The 2D data logger measures the frequency of gearbox output shaft rotations and converts to RPM for recording.

**LAF**

The Air/Fuel Ratio (AFR) of the engines exhaust gases. The measurement is made by the Bosch LAF sensor as supplied by 2D. The measurement has no dimension as it is simply a ratio of air to fuel.

**LAF\_Temp**

The temperature of the LAF sensor. The measurement is made by a thermocouple built inside the Bosch LAF sensor as supplied by 2D.

**LAF\_Heat**

The switched-on power of the heating element inside the LAF sensor. The *LAF\_Heating* is controlled by the 2D engine interface module according to the measured channel "*LAF\_Temp*" and regulates a stable temperature for the LAF probe.

**Throttle**

The throttle opening of the engine intake manifold. The sensor used for measurement is a standard HRC sensor you should receive with your Honda engine. This channel is measured in % and requires a full calibration (min/max/zero defined) before you operate the bike.

**T\_Water**

The temperature of the engine's coolant fluid (water). The sensor used for measurement is a standard HRC sensor you should receive with your Honda engine. This measurement is made with units of degrees Celsius.

**IMAP**

Air pressure inside the bike's air box. The sensor used for measurement is a standard HRC sensor you should receive with your Honda engine. The units of this measurement are mbar. This channel makes a measurement of absolute air pressure and should not need any calibration or zero.

**T\_Air**

Air temperature inside the air box of the bike. The sensor used for measurement is a standard HRC sensor you should receive with your Honda engine. The units of this measurement are degrees Celsius.

**ECUErr**

Error signal from the HRC kit ECU. Measurement of this channel allows you to check the correct function of the ECU. This measurement is made by direct connection with the HRC ECU. The measurement has units of volts.

**P\_Fuel**

The fuel pressure contained inside the engine fuel rail. This measurement is made by a 10 bar 2D pressure sensor and has units of bar. This channel should not require any zero or calibration steps and will provide a measurement of absolute fuel pressure for you to analyze.

### 3.5 2D program Analyzer

The program 2D Analyzer is used to make analysis of your bike's measurement data



#### Overview window

The Overview window is located at the bottom of the screen. Selected channel traces and lap time information are shown inside this area.

The rectangle in the Overview window marks the position of the selection that can be seen in the Moving window. You can move the rectangle by pressing the left mouse button and moving the mouse. If you press the <Ctrl> key simultaneously you can change the size of the rectangle

#### Moving window

The Moving window is the main analyzing window where the channel traces of the selected part of the measurement are shown. The displayed section of the measurement can be moved by pressing the left mouse button and moving the mouse left/right.

To zoom in or out, hold the left mouse button pressed and move the cursor up or down. With a double click on the Moving window you change into the Measuring mode (see measuring mode).

#### Circuit window

The Circuit window is located in the bottom-right corner and displays the track map of your current measurement. The currently active (shown on screen) section of data is highlighted inside the Circuit window.

#### Measuring mode

This enables a very accurate analysis of the data currently selected on the screen. A measuring cross becomes active to show what data is selected. Another window opens (value window) to display the exact physical values of the current data on the screen.

You can enter the measuring mode by:

- Double-click in the Moving window
- Press the <Enter> key of your PC
- Press the <Space> key of your PC

You can start all functions from the main menu, context menus (right mouse click). The toolbar is divided into the following functions (from left to right)



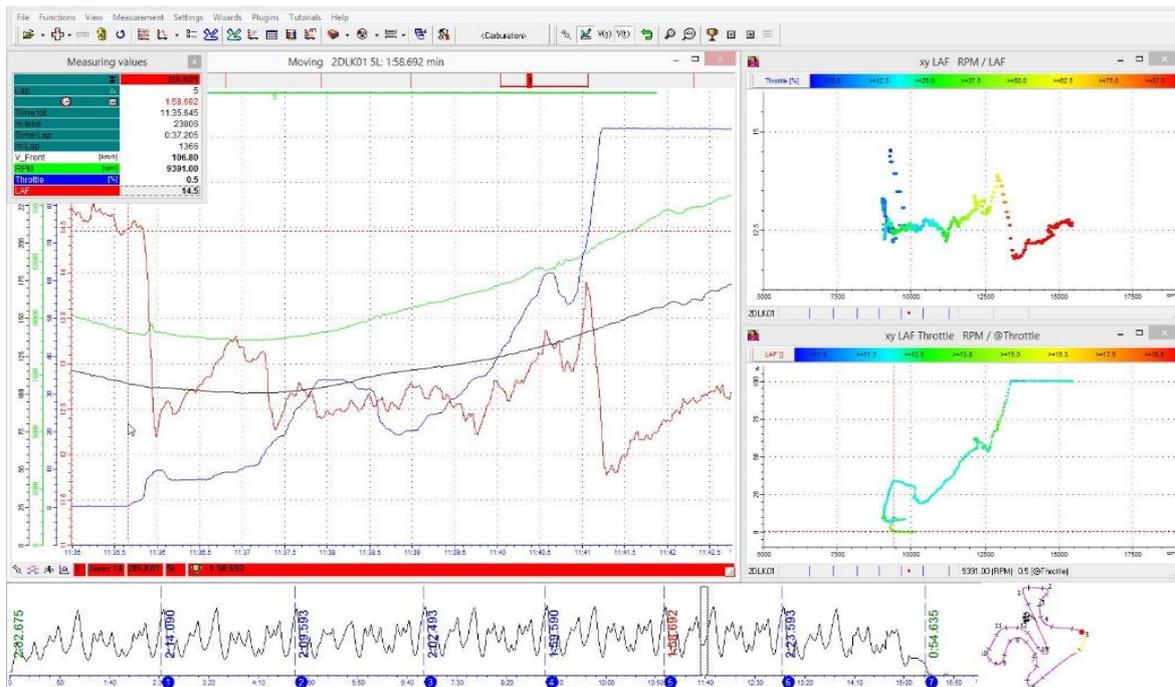
- **File functions** are functions to load and exchange 2D measurement files
- **Basic analysis** functions include Moving and Overview windows and lap times
- **Advanced analysis** functions include Track mode, XY-Plot and Min/max table
- **Visibility of channels** in the Moving window can be switched separately or by groups or via areas
- **Another function** accessible via toolbar is the selection of a screen template.
  - The **measurement functions** include commands to display the measurement(s) in different ways e.g. plot vs. time/distance

### 3.5.1 Fixed analysis layouts for Moto2™ kit license

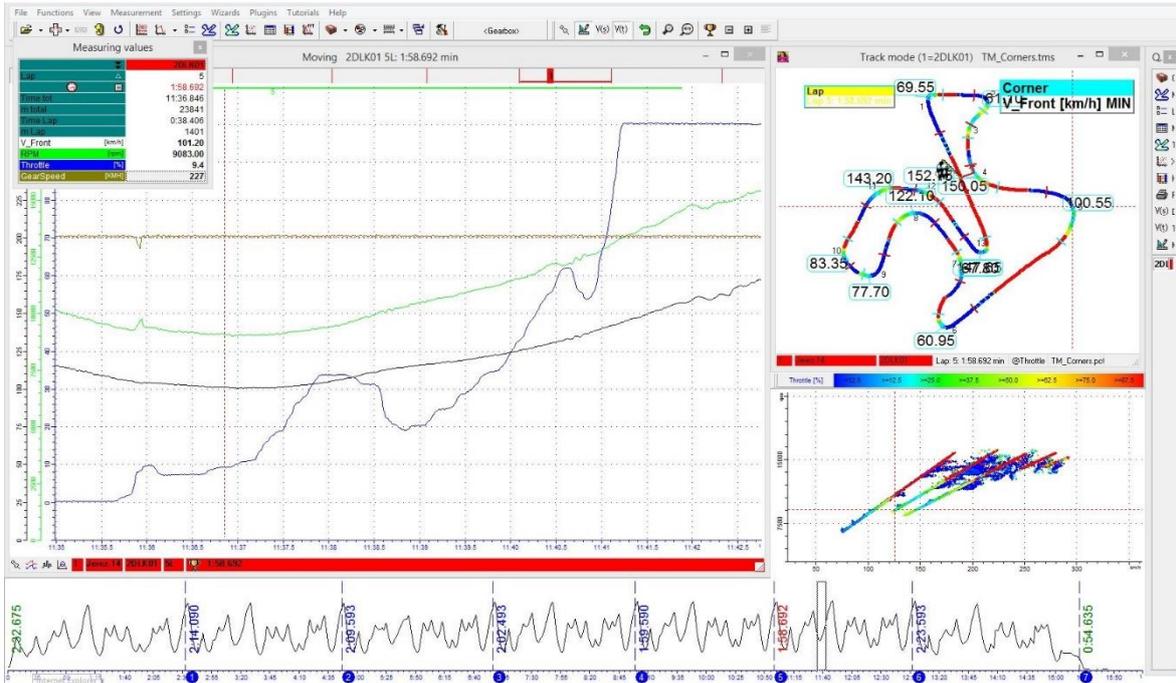
The Moto2™ kit license analysis software is delivered with your Moto2™ data recording system. This provides a predefined (fixed setting) analysis tool that you can use to view your bike's recorded data.

Your kit analysis software has three predefined "layouts":

- **Carburation** – for checking your fuel mapping
- **Gearbox** – to check your bike gearing and corner speeds
- **Overview** – full screen view of data-trace v time/distance



Carburation



Gearbox

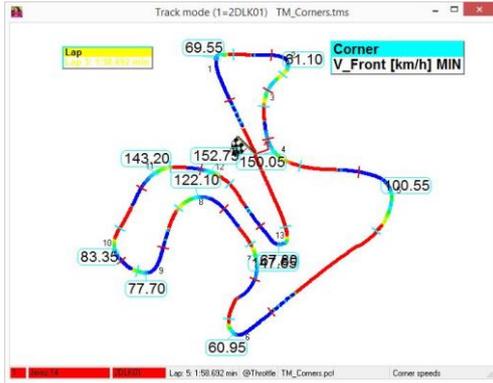


Overview

### 3.5.2 Analysis tools for Moto2™ kit license

The Moto2™ kit software is delivered with several predefined tools, for example:

- **Track mode** – a track map layout demonstrating corner speeds on the circuit
- **Min/Max tables** – giving a fast summary of min/max channel values
- **XY plots** – to compare two channels against each other in a plot
- **Histogram** – statistic info to help explain bike operation/performance



Track mode

Min/max table (1=2DLK01) Engine.mms

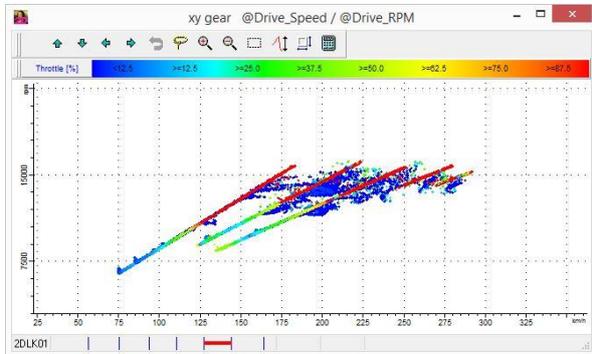
Lap	Time	Meters	RPM		T_Water	
			MIN	MAX	MIN	MAX
0	2:32.675 min	4460	59.00	15644.00	48.0	57.2
1	2:14.090 min	4461	5478.00	15949.00	56.4	62.4
2	2:09.593 min	4480	5934.00	16130.00	60.0	65.8
3	2:02.493 min	4473	6354.00	16229.00	64.6	68.6
4	1:59.590 min	4471	6436.00	16214.00	66.2	70.4
5	1:58.632 min	4491	6466.00	16175.00	67.7	71.9
6	2:23.593 min	4304	705.00	15774.00	64.7	72.5
7	54.635 sec	43	0.00	705.00	70.3	74.1
MIN			0.00	705.00	48.0	57.2
MAX			6466.00	16229.00	70.3	74.1
AVG			3928.88	14102.50	62.2	67.9

Min/Max table for engine

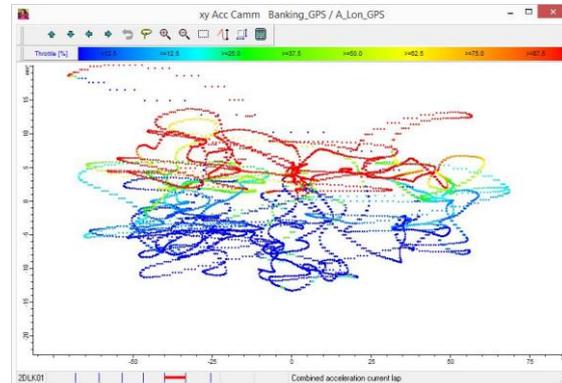
Min/max table (1=2DLK01) mm Bike.mms

Lap	Time	Meters	V_Front	V_GPS	RPM	Throttle	Swing_F	Swing_R	Yaw	Swing_F			Swing_R			Yaw
										MIN	MAX	AVG	MIN	MAX	AVG	
0	2:32.675 min	4460	222.40	221.2	15644.00	9171.66	0.2	100.6	26.5	0.25	97.84	39.96	3.53	28.98	15.82	13.47
1	2:14.090 min	4461	228.80	229.6	15949.00	10452.78	0.1	100.7	32.0	0.13	112.06	45.21	1.57	26.79	15.40	13.53
2	2:09.593 min	4480	230.90	236.0	16130.00	10768.94	0.0	100.6	30.7	0.26	113.68	45.97	2.00	27.99	16.08	13.54
3	2:02.493 min	4473	234.35	240.4	16229.00	11479.28	0.0	100.6	37.5	0.30	121.14	48.48	0.10	29.76	15.77	13.56
4	1:59.590 min	4471	237.10	237.3	16214.00	11652.20	-0.0	100.6	39.1	0.25	126.37	48.58	0.10	29.38	16.00	13.55
5	1:58.632 min	4491	236.95	235.8	16175.00	11718.33	-0.0	100.5	40.0	0.19	126.36	49.02	0.17	30.39	16.01	13.55
6	2:23.593 min	4304	236.65	236.8	15774.00	9032.89	-0.1	100.5	24.2	0.03	125.50	45.03	0.17	38.31	15.77	13.51
7	54.635 sec	43	36.80	41.8	705.00	33.23	-0.2	0.3	-0.0	0.16	77.23	23.55	4.20	12.27	5.94	13.09
MIN			36.80	41.8	705.00	33.23	-0.2	0.3	-0.0	0.03	77.23	23.55	0.10	12.27	5.94	13.09
MAX			237.10	240.4	16229.00	11718.33	0.2	100.7	40.3	0.30	126.37	49.02	4.20	38.31	16.08	13.56
AVG			207.99	209.9	14102.50	9288.66	-0.0	88.0	28.8	0.20	112.62	43.22	1.48	27.98	14.60	13.48

Min/Max table for bike



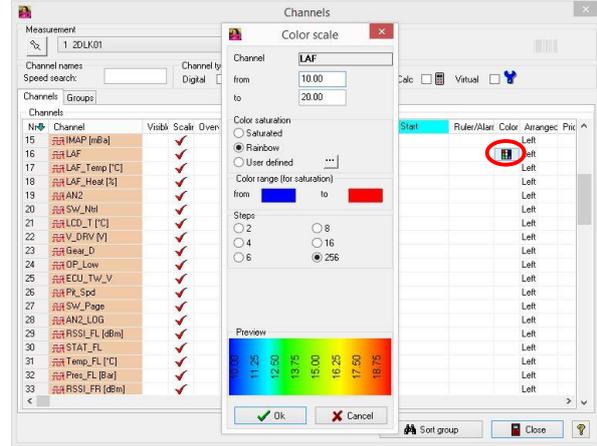
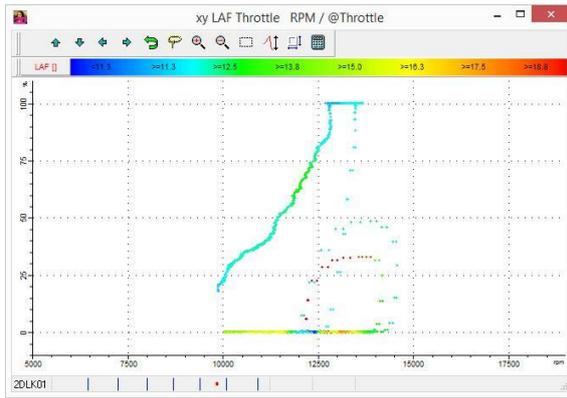
XY plot for analysis of gearbox



XY plot for bike lean angle and acceleration

**XY plot for engine LAF analysis**

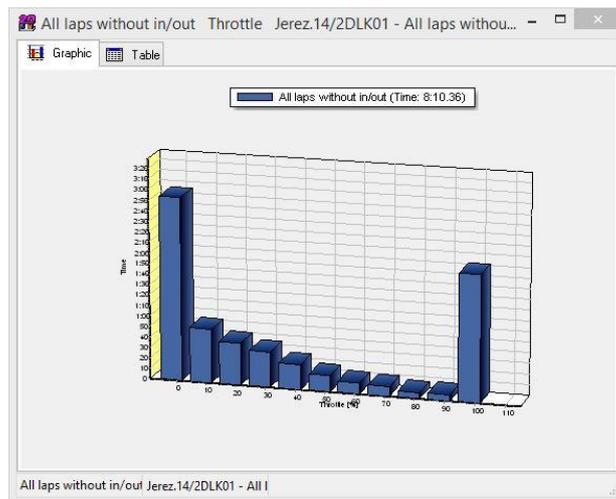
This is very useful for inspecting the fuel mapping of your bike. Only the data contained inside the moving window is displayed inside the XY plot. Therefore it is possible for you to select a small section of your bike data and use the XY plot to help make changes to your ECU fuel mapping.



To change the scaling of the LAF “color” channel:

- select the <Channels> button from the toolbar
- locate the channel “LAF” from the channel list
- selecting the “Color” field (circled above in red)
- Inside this option you can define the color range that is used for the LAF channel in XY plot

**Histogram for throttle** – giving statistical view of throttle value during your session/lap



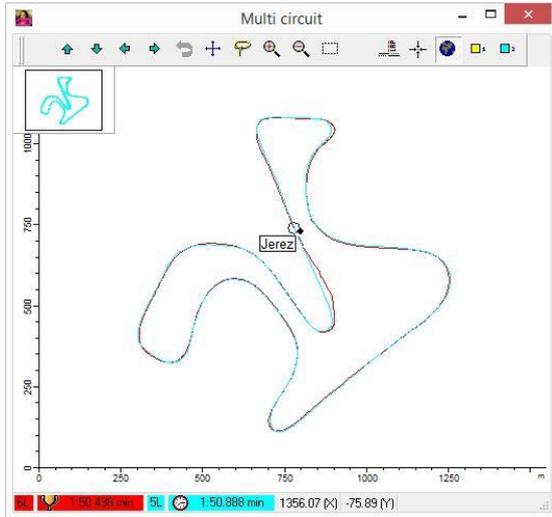
### Multi-Circuit view

This gives the option to view the driving lines taken by multiple laps of data. You can activate this feature by pressing the key <M> on your keyboard.

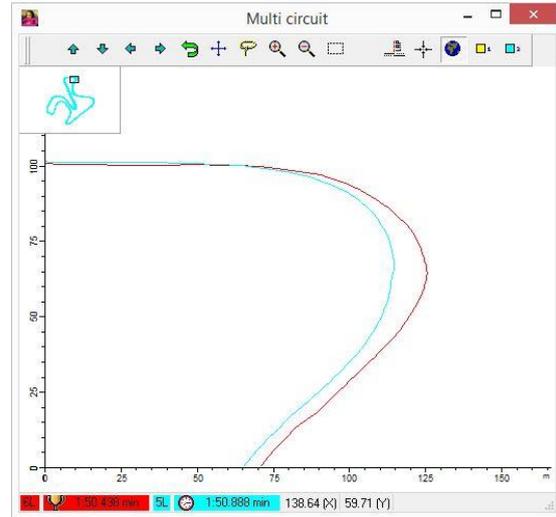


The full version of 2D Analyzer enables an automatic download of an aerial photograph of the track which can be used to better analyze the bike’s driving line.

With the kit software it is not possible to download the aerial photograph and you must use only the blank screen for line analysis.



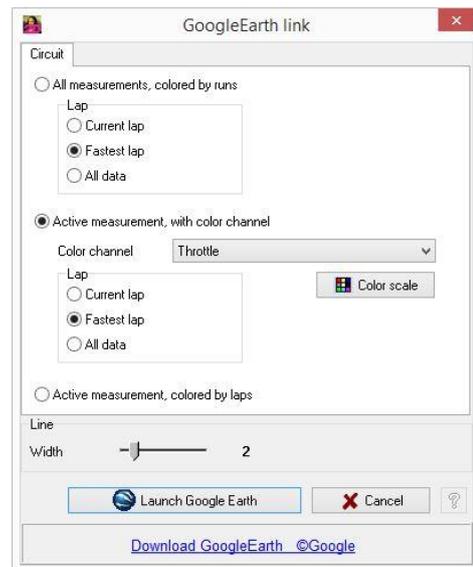
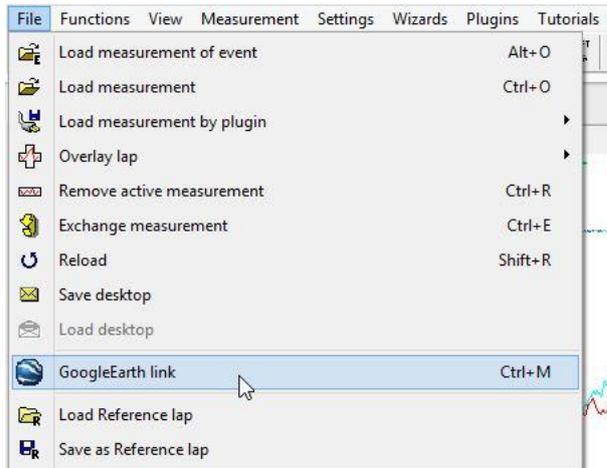
Full view of circuit



Zoom view of riding lines at turn 2

### 3.5.3 Google Earth™ Link

While download of the aerial map is not possible with kit software, you can use the *Google Earth™* link to export your bikes riding line to the actual program *Google Earth™*.





Riding line and throttle displayed for one lap of Circuito de Jerez, Jerez.



To make this feature possible you must have an active internet connection and also the most recent installation of *Google Earth™*.

### 3.6 X2 settings

The X2 transponder technology was introduced by Dorna for transmitting information about lap/section times, flag signals and other information. The transmitter receives this information when passing specific signaling loops in the racing tracks and puts them as messages onto the CAN bus.

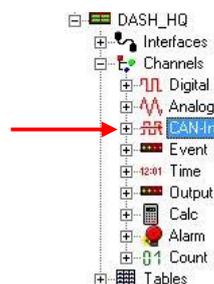
Your Moto2™ µCAN logger is pre-programmed to receive these X2 signals on CAN\_EXT (CAN-2). The logger routes these signals to the CAN\_2D (CAN-1) bus so that you can use them in your dash as well. Please refer to section 4 at the end of this document for a list of available X2 signals and their meaning.

In the following it is described how to configure the 2D products for processing the X2 signals.

#### 3.6.1 Receiving X2 transponder data

For changing the setting of your 2D display devices you connect them to your PC and start *WinIt*. Be aware that some 2D devices need to be powered by external power supply before connecting them to your PC.

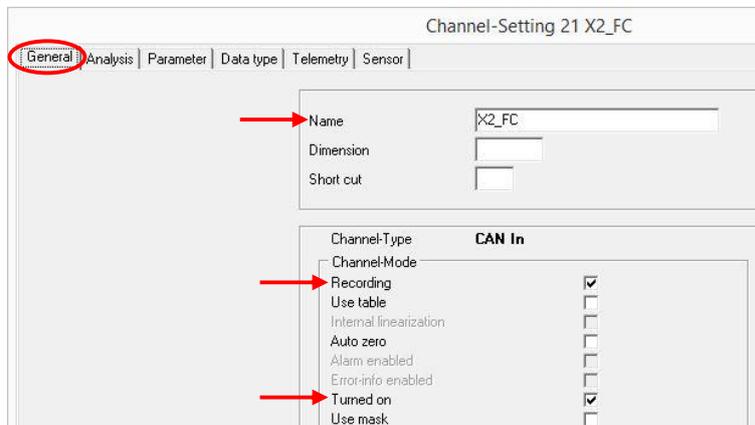
Select the device in the system tree, for example your dash or LED Bar. Go to “Channels”, “CAN-In”.



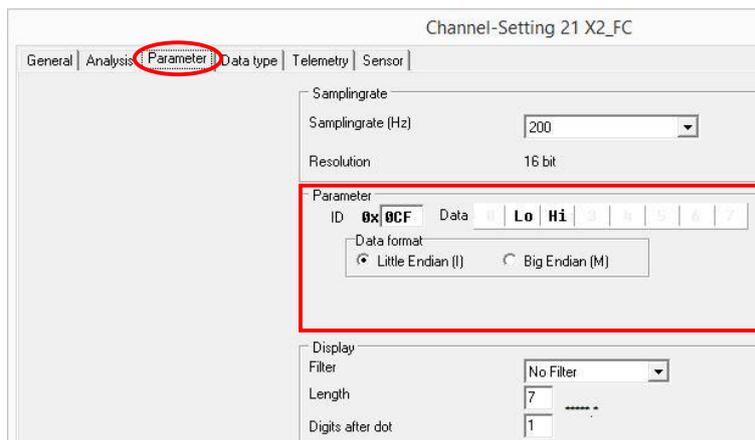
### 3.6.1.1 COMM\_FIRST\_CONTACT

The COMM\_FIRST\_CONTACT channel contains information of all track loops (pit entry, pit exit, top speed, IP1, IP2, IP3, finish line and an additional “rider’s lap” trigger with ID 30<sup>3</sup>).

To receive the track loop data you select a 16 bit channel. Turn it on and name it, if you want (tab “General”). If you want to record the signals for later analysis, tick also the “Recording” box.



Set the identifier to “0x0CF” (tab “Parameter”) and select the bytes as shown below.



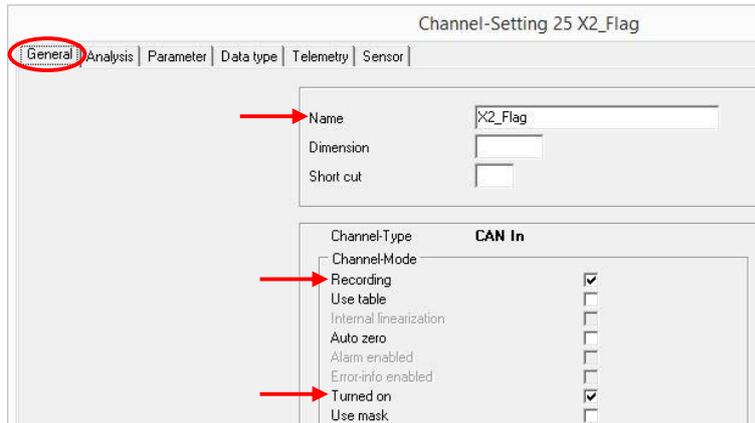
Confirm all your changes with <Apply>.

### 3.6.1.2 FLAGS\_MSG

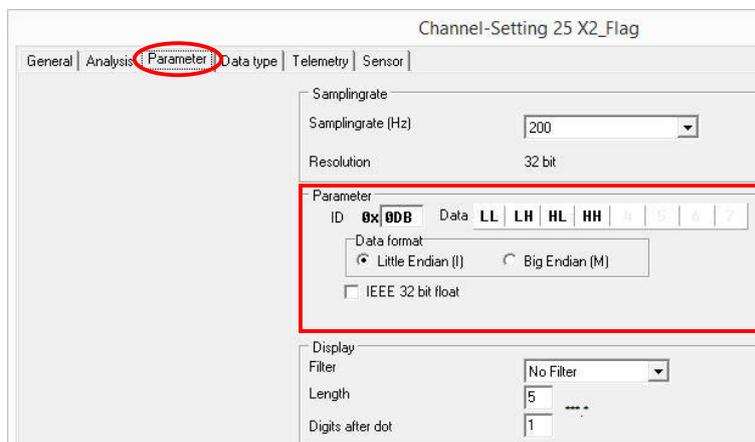
The FLAGS\_MSG channel contains the flag information. Please refer to Appendix 4.2 for a list of all flag signals

To receive the flag data you select a 32 bit channel. Turn it on and name it, if you want (tab “General”). If you want to record the signals for later analysis, tick also the “Recording” box.

<sup>3</sup> The “rider’s lap” trigger with ID 30 is only available in certain circuits.



Set the identifier to “0x0DB” (tab “Parameter”) and select the bytes as shown below.

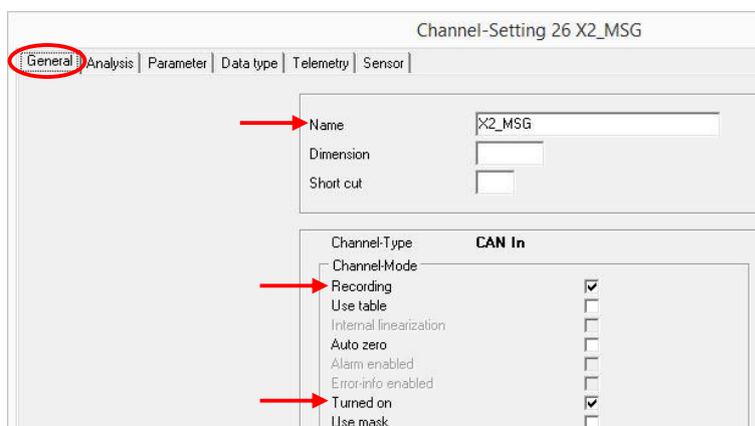


Confirm all your changes with <Apply>.

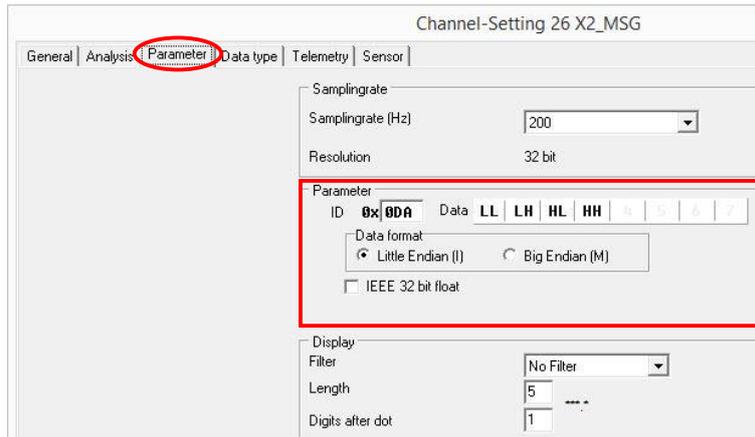
### 3.6.1.3 INFO\_MSG

The INFO\_MSG channel contains Dorna messages with up to eight characters.

To receive the messages you select a 32 bit channel. Turn it on and name it, if you want (tab “General”). If you want to record the signals for later analysis, tick also the “Recording” box.



Set the identifier to “0x0DA” (tab “Parameter”) and select the bytes as shown below



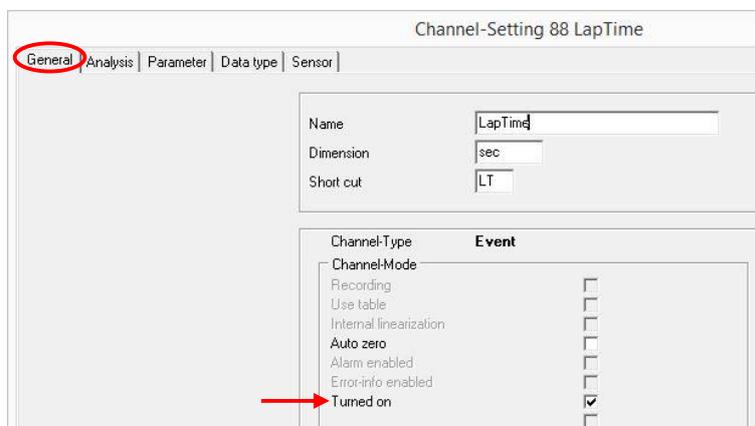
Confirm all your changes with <Apply>.

### 3.6.2 Lap times

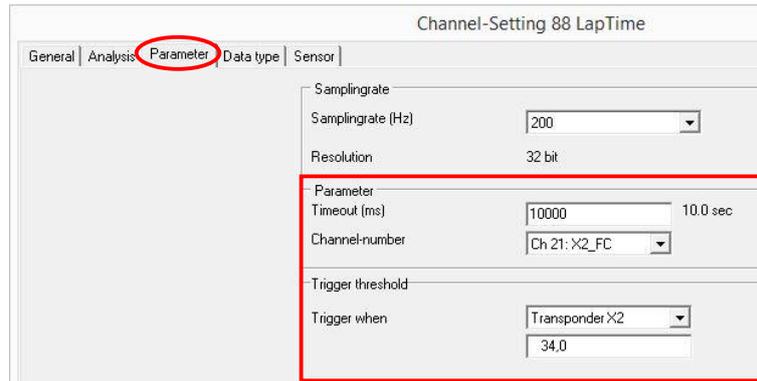
For creating lap times with the X2 transponder trigger signal you need to receive the X2 data “COMM\_FIRST\_CONTACT”. Refer to section 3.6.1.1 on how to receive this channel.

Your 2D Moto2™ µCAN logger is pre-programmed to receive the “COMM\_FIRST\_CONTACT” data on channel “T\_X2\_ID”.

To change the setting of lap times connect your device to your PC and start *WinIt*. Select your device in the system tree and go to “Channels”, “Event”. There you select the channel “LapTime” and turn it on (tab “General”).



In tab “Parameter” field “Parameter” you select the modified channel with your lap trigger information (X2\_FC in your display device or T\_X2\_ID in your logger) and you can change the lap time timeout (⇒ during this time another lap time signal will be ignored). In field “Trigger threshold” you select “Transponder X2” from the drop-down list and enter the value 34 (finish line) or 30 (additional lap trigger with ID 30<sup>3</sup>). In your Moto2™ µCAN logger the value for the finish line (ID 34) is fixed and cannot be changed.



Confirm all your changes with <Apply>.



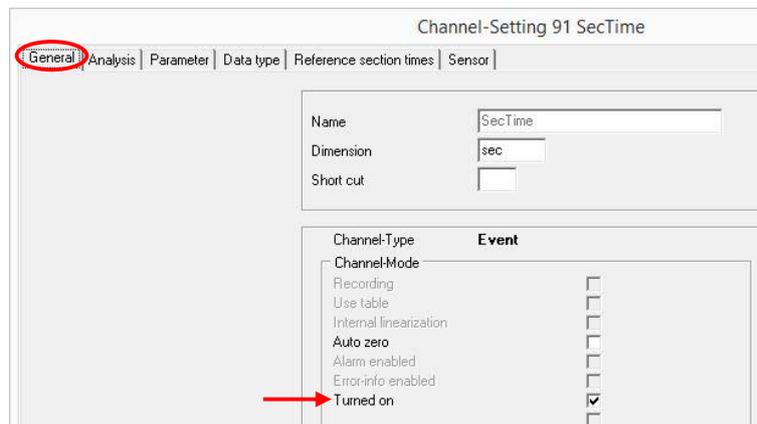
If you use ID 30 (the additional lap trigger) for generating lap times, your lap time may differ from official time keeping, as it is taken on a different position on track. For further information on track loops refer to section 4.1.

### 3.6.3 Section times

For creating section times with the X2 transponder trigger signal you need to receive the X2 data "COMM\_FIRST\_CONTACT". Refer to section 3.6.1.1 on how to receive this channel.

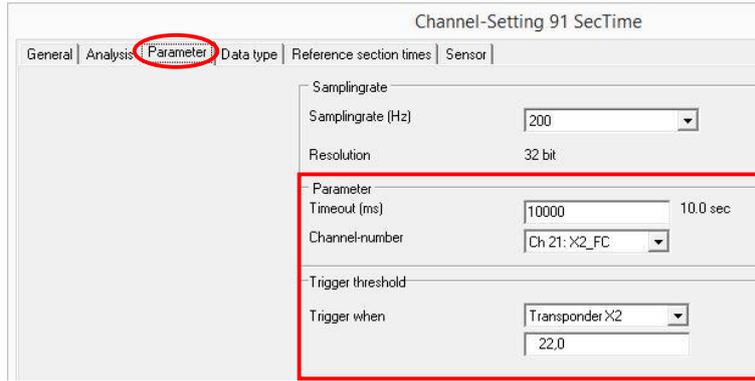
If you want to generate section times with the X2 transponder signal in your Moto2™ µCAN logger you only have to change the setting for lap times (refer to section 3.6.2). By changing the lap time setting to X2 transponder the section times are automatically generated with the X2 data.

To change the setting of section times connect your device to your PC and start *WinIt*. Select your device (for example dash or LED Bar) in the system tree and go to "Channels", "Event". There you select the channel "SecTime" and turn it on (tab "General").



In tab "Parameter" field "Parameter" you select the modified channel with your lap trigger information (X2\_FC) and you can change the section time timeout (⇒ during this time another section time signal will be ignored). In field "Trigger threshold" you select "Transponder X2" from the drop-down list and enter the value 22. (ID 22, 24 and 26 will be used as section times<sup>4</sup>.)

<sup>4</sup> For further information on track loops and their IDs refer to section 4.1.



Confirm all your changes with <Apply>.

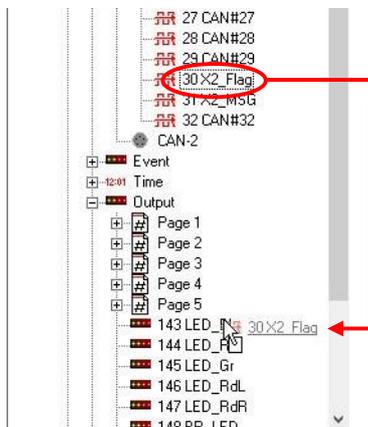
### 3.6.4 Flags

For displaying the flag information you need to receive the X2 data “FLAGS\_MSG”. Refer to section 3.6.1.2 on how to receive this channel.

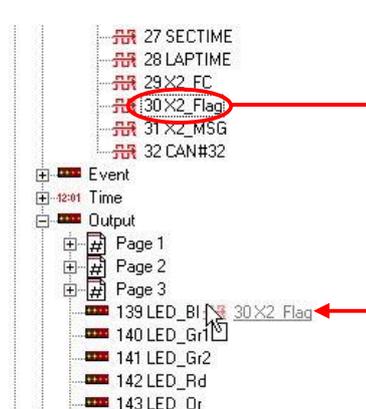
It is mandatory to indicate the flags with a light signal. On your MidiDash and BigDash you can use the blue status LED.

To activate the blue status LED you take your flag channel (X2\_Flag) and put it onto the blue LED output channel (“Channels”, “Output”, “LED\_BI”) via drag & drop. By dropping the channel the status LED is automatically turned on. Otherwise turn it on.

MidiDash:

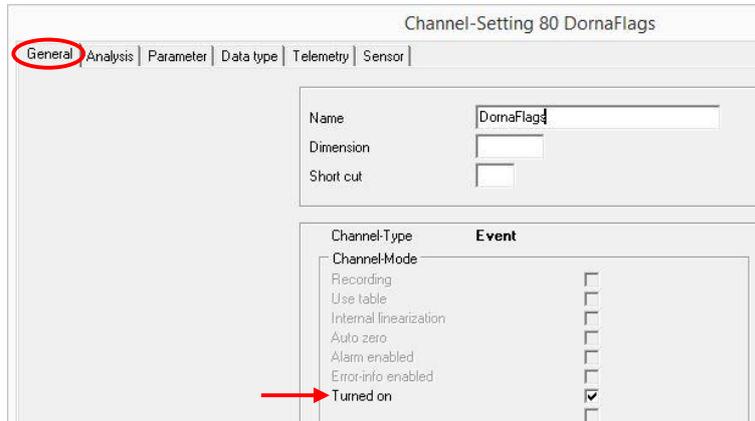


BigDash:

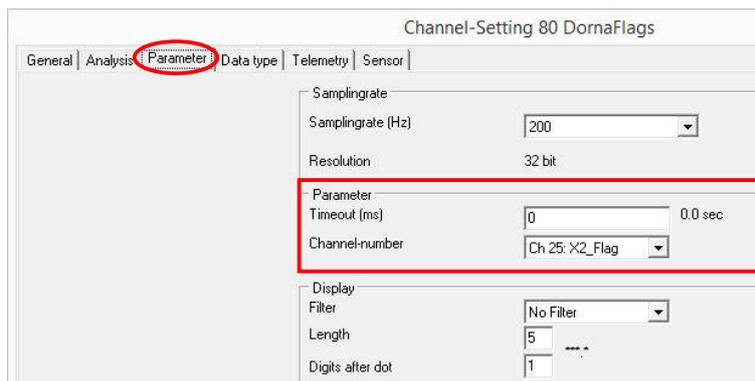


As soon as you program the blue status LED for indicating flags you can't use it for alarm indications. Otherwise the status LED will be turned off!

To display the flags on the LED Bar or the flag texts your dash you have to activate the “DornaFlags” function. Go to “Event” and turn on “DornaFlags”:



Set the channel (tab “Parameter”) to your modified 32 bit channel (X2\_Flag). You can also specify a timeout value. If the timeout is set to zero, the flag will be shown until the CAN message for this channel changes. If the timeout is greater than zero, the flag will be shown until the timeout value expires or the CAN message changes - whatever comes first. If during a timeout another flag is received this new flag is shown and the timeout is restarted.



The displaying of flags has a higher priority than normal output pages. If a flag signal is received the flag will be shown automatically. You don´t have to put this function on a page.

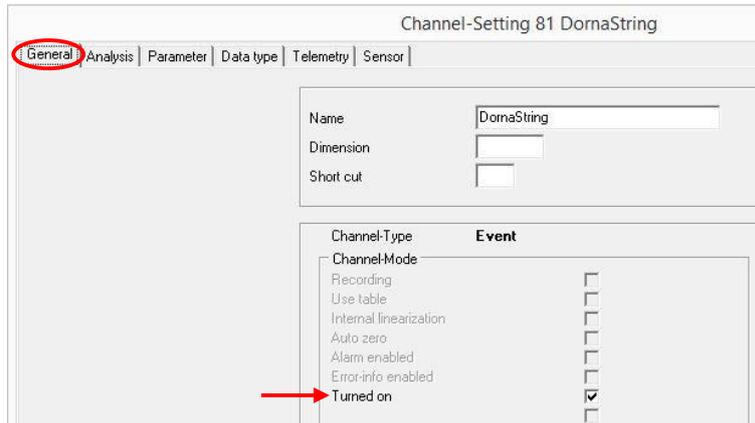
Confirm all your changes with <Apply>.

If more than one flag is received at the same time only the flag with higher priority will be shown. You can find a table with all flag indications (LED Bar or dash texts) and their priority in section 4.2.

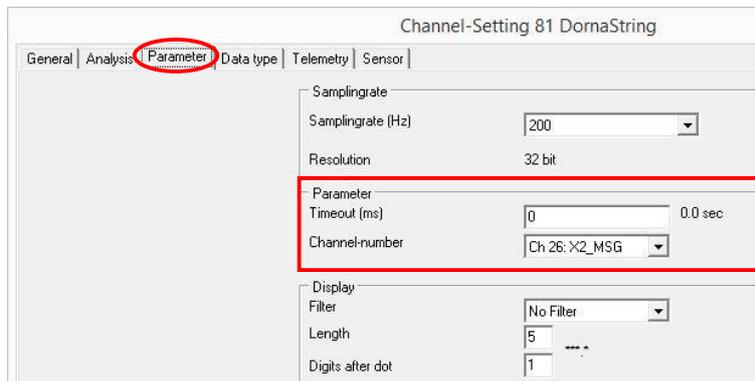
### 3.6.5 Dorna info messages

For displaying the Dorna info messages you need to receive the X2 data “INFO\_MSG”. Refer to section 3.6.1.3 on how to receive this channel.

To display the texts on your dash you have to activate the “DornaString” function. Go to “Event” and turn on “DornaString”:



Set the channel (tab “Parameter”) to your modified 32 bit channel (X2\_MSG). You can also specify a timeout value. If the timeout is set to zero, the text will be shown until the CAN message for this channel changes. If the timeout is greater than zero, the text will be shown until the timeout value expires or the CAN message changes - whatever comes first. If during a timeout another info message signal is received this new text is shown and the timeout is restarted.



The displaying of Dorna info messages has a higher priority than normal output pages. If a flag signal is received the test will be shown automatically. You don´t have to put this function on a page.

Confirm all your changes with <Apply>.

### 3.7 GPS channels

By using the 2D GPS module (mouse), 15 additional GPS channels will be measured by the 2D data recording system. The GPS data allows the riding line to be analyzed in high detail.

The GPS channels are as follows:

- **V\_Sat** – speed of bike according to changing of its GPS coordinates
- **ValidSat** – number of valid satellites signals being received
- **HHMM** – time of day in hours and minutes
- **Course** – the direction the bike is currently travelling
- **Lat\_dez** – lateral GPS coordinates for bike position
- **Lon\_dez** – longitudinal GPS coordinates for bike position
- **Altitude** – estimated altitude of the bike
- **MMDD** – the current date in month and day format
- **SSHH** – the current time in seconds and hundredth of a second format
- **A\_Lat** – lateral acceleration of bike according to changing of its GPS coordinates
- **A\_Lon** – longitudinal acceleration of bike according to changing of its GPS coordinates
- **Banking** – estimated banking angle of the bike when cornering

- **YawRate** – the rate at which the bike is changing its cornering radius
- **SpAccu** – the estimated accuracy of the GPS speed measurement
- **LapGps** – to enable lap time generation when no official circuit timing is available

### 3.7.1 Generate lap times with GPS

Lap times can be generated by the channel “*LapGps*” using the 2D GPS module. *LapGps* provides an alternative lap trigger signal that can be used when it is not possible to generate lap times with the X2 transponder.



LapGps does not make lap times by itself, but simply generates a **lap trigger signal** when the GPS location of the bike is matching the GPS location of the defined lap trigger coordinates (start line).

To create lap times with the trigger signal of “*LapGps*” you change the setting of your lap time event channel. Connect your logger to your PC and start *WinIt*. Select your logger and go to “Channels”, “Event” ⇒ “LAPTIME”. There you select the tab “Parameter”. In field “Parameter” you select the channel “LapGps” and enter a timeout value. Within this timeout value other lap trigger signals will be ignored.

The GPS lap trigger signal can be generated in two ways:

1. **Automatically** - using the table “*2DTrkPos*” (already loaded inside your Moto2™ data logger)
2. **Manually** – you input the coordinates of your circuit start/finish line (read from line file)

#### 3.7.1.1 Automatic setting of LapGps

The table “*2DTrkPos*” contains the start line coordinates of many popular race tracks from around the world. The process is automatic but you must define a “*radius*” from the table-defined start line coordinates that the *LapGps* trigger signal is active. Recommended values for the radius range from 25 m-50 m.



The GPS module compares its measured coordinates with those inside the table “*2DTrkPos*” and generates a lap trigger signal when the bike passes within the defined search radius of any start/finish coordinates defined in the table.

The screenshot shows the software interface with the 'Laptrigger coordinates' dialog box open. The 'Radius (10-200 m)' field is set to 30. A red arrow points to this field with the text: "Define a search radius for locating the start line!".

Below the dialog box, a scatter plot shows the race track data points. A red arrow points to the plot with the text: "Start line coordinates for many race circuits!".

In the left-hand tree view, the '15 LapGps' item is highlighted with a red arrow. Below it, the '4 2DTrkPos' item is also highlighted with a red arrow.

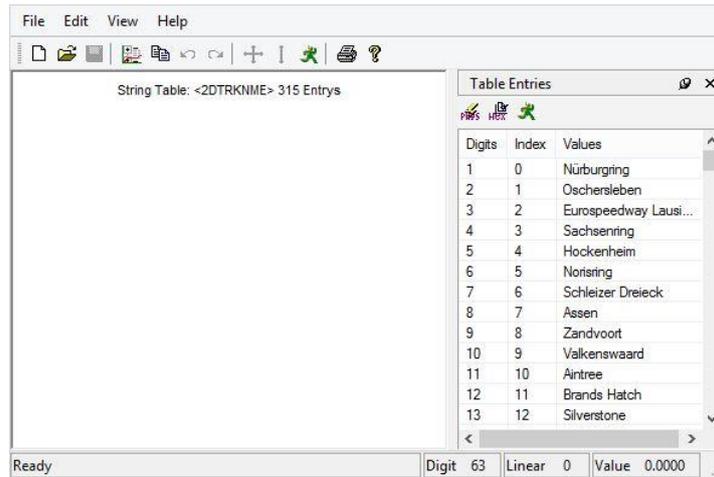


The defined search radius must be large enough to accommodate different riding lines and also GPS drift. However if the search radius is too large it may be possible to generate invalid lap times due to the lap signal occurring more than once in a lap, see figure below!



To check if your race track is inside the table "2DTrkPos", go to your system directory; <C:\ProgramData\RaceMoto2\_14.0\System\Tables><sup>5</sup> and open the file "2DTrkPos". Inside this table you can view all the race tracks currently accommodated by the automatic function for GPS lap trigger.

<sup>5</sup> You reach this folder by selecting *Settings* ⇒ *Folders – Protocol* in the *WinARace* menu. There you select the button <Application data>. In the opened folder you select <System>, <Tables>.



The screenshot shows a software window titled "String Table: <2DTRKNME> 315 Entries". On the right side, there is a "Table Entries" panel with a scrollable list. The list has three columns: "Digits", "Index", and "Values". The entries are as follows:

Digits	Index	Values
1	0	Nürburgring
2	1	Oschersleben
3	2	Eurospeedway Lausi...
4	3	Sachsenring
5	4	Hockenheim
6	5	Noisring
7	6	Schleizer Dreieck
8	7	Assen
9	8	Zandvoort
10	9	Valkenswaard
11	10	Aintree
12	11	Brands Hatch
13	12	Silverstone

At the bottom of the window, there is a status bar with fields for "Digit 63", "Linear 0", and "Value 0.0000".



If your circuit is not inside the table “2DTrkPos”, you should first check the 2D automatic updates to ensure you have the latest version of the table!

See section 2.3.4 “How to update the software via Internet” for explanation on how to perform the automatic updates!

### 3.7.1.2 Manual Setting of LapGps

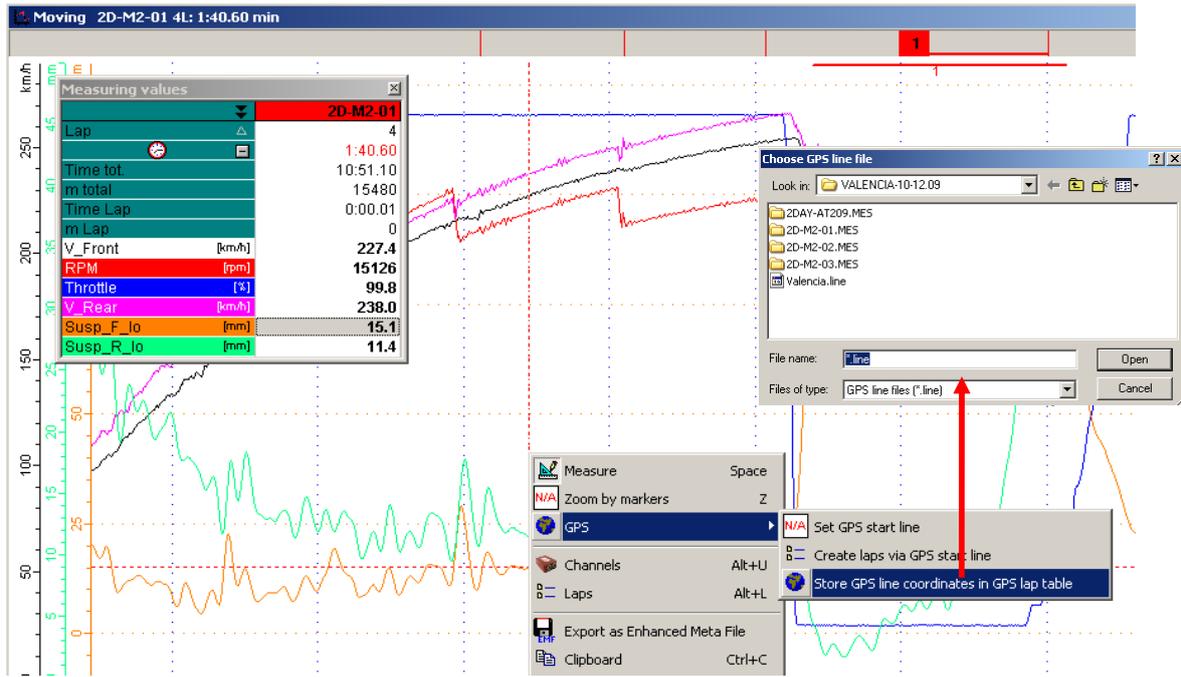
You can manually define the start line coordinates of your circuit. This is required when your circuit is not in the table “2Dtrknme” **or** you wish to modify the position on the circuit at which you are generating lap times by GPS.



If the track on which you are racing is not inside the table “2Dtrknme”, the automatic GPS lap trigger will not work!

To manually define the coordinates of your circuit start line, follow the steps below:

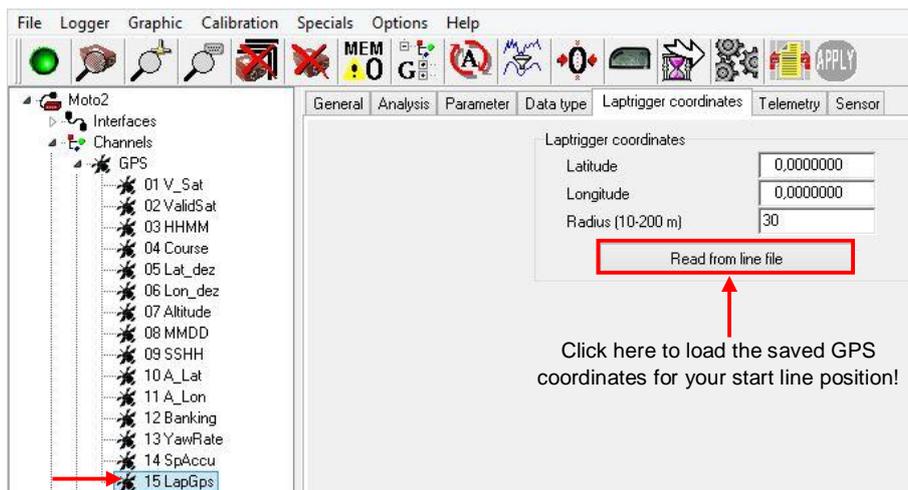
1. To obtain an accurate GPS measurement of the circuit layout and start line position.  
*Make a short installation lap to measure the circuit start line coordinates.*
2. Use the program 2D Analyzer to select your preferred start line position.  
*Open your measurement in 2D Analyzer, press <space> to enter measure-mode and position the cursor at your preferred start line position.*



- Use 2D Analyzer to save your start line position to a table. Right-click with the mouse at your preferred start line position, select <GPS> + <Store GPS line coordinates in GPS lap table>. Save the GPS coordinates to your computer at a location you will remember!
- Load your GPS coordinates to the data logger to trigger lap times by GPS. Click <Read from line file> to open the table saved from your initial measurement.



The line file is loaded easily by clicking “Read from line file” and finding the file from the location you saved it. You must also define a radius of tolerance within which the lap signal is generated by the GPS. A recommended value is from 25 m-50 m.



The Moto2™ data logger will always give priority to the values loaded by <Read from line file>. If the data logger does not measure your bike to pass within the defined radius (e.g. 30 m) of the start line, it will immediately check the loaded table “2DTrkPos”.



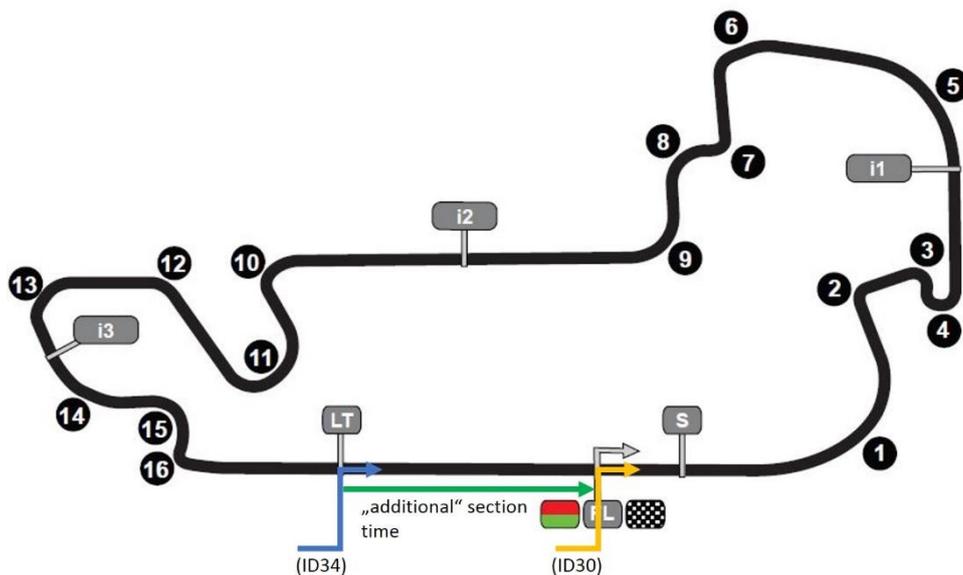
Once you have correctly defined the channel “LapGps” you must then change the settings of the event channel “LapTime” to use “LapGps” as the trigger signal. The definition of the LapTime channel will be explained next!

## 4 Appendix

### 4.1 Track loops

The X2 transponder receives signals from different track loops while passing them. Every loop has its own identifier:

ID	Description
3	Pit entry
18	Pit exit
20	Top speed (S)
22	IP1 (i1); end of section 1
24	IP2 (i2); end of section 2
26	IP3 (i3); end of section 3
30	Finish line (FL)
34	Lap time (LT)



By selecting ID 34 for your lap time, your lap times will be generated at the track loop "lap time". If you select ID 30 for your lap time, then the lap times will be generated at the track loop "finish line". Please keep in mind, that track loop "finish line" is not available at every track.

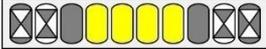
If you record the trigger signals, you can use the lap trigger with ID 30 to create an additional section time in your analysis.

## 4.2 Flag indications

The 2D displays and LED Bar are all able to indicate flags: The LED Bar uses its multi-color LEDs to indicate the flags. The MidiDash and BigDash show a text in combination with a single LED.

The following table shows an excerpt of the displayed flags and texts supported by the 2D devices. These flags are used by the Dorna since 2014.

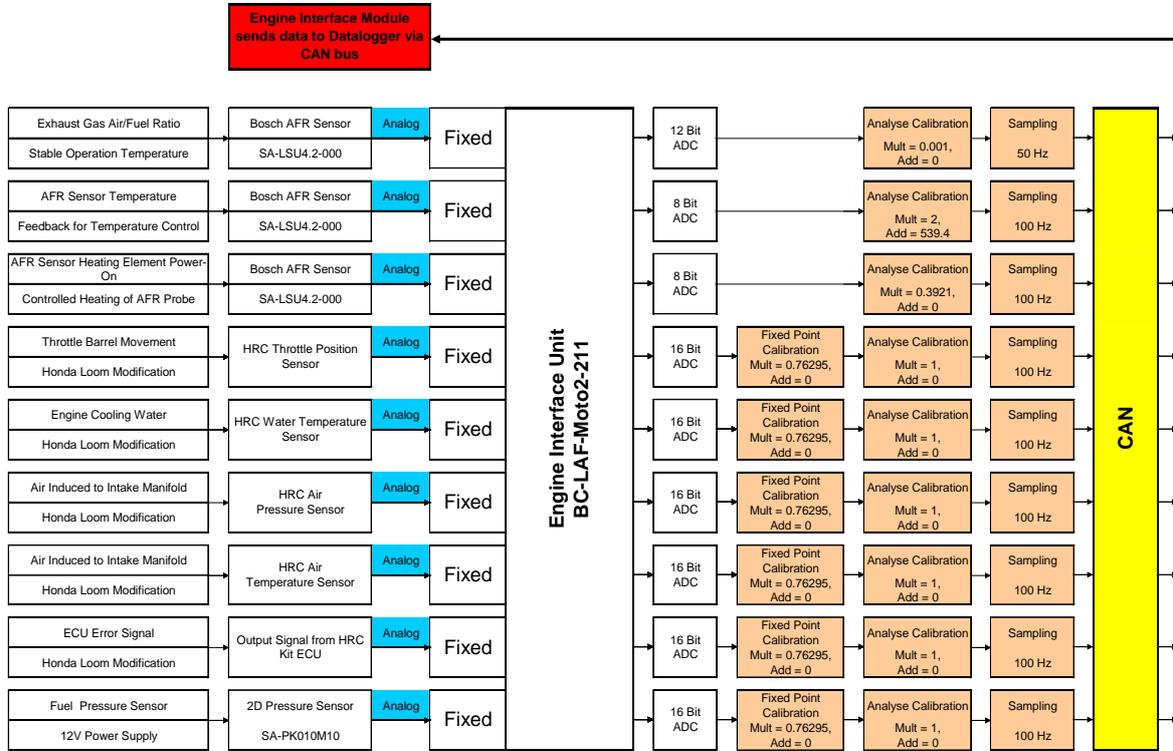
If more than one flag is received at the same time only the flag with higher priority will be shown. (Smaller number ⇔ higher priority; 1 ⇔ highest priority.)

Name	LED Bar	Dash	Priority
Red flag		RED	1
Black flag with orange disc		BLACK/ORANGE	2
Black flag		BLACK	4
Penalty: ride through		RIDE THROUGH	5
Penalty: go back 1 position		BACK 1 POS	6
Penalty: go back 2 positions		BACK 2 POS	6
Penalty: go back 3 positions		BACK 3 POS	6
Penalty: go back 4 positions		BACK 4 POS	6
 LEDs off	 flashing LEDs (bright white/off)	 flashing LEDs (red/off)	

### 4.3 Measured channels generation

#### 4.3.1 Engine interface module

The diagram below gives a detailed illustration of the engine interface module's operation.



This module simultaneously measures the analog input signals (from the sensors), converts each measurement to digital format using the Analog to Digital Converter (ADC), applies the pre-set calibration functions and finally sends the data for each channel to the data logger via the CAN bus.

While the analog measurements are made by a 16 bit ADC (digit values 0 – 65535), a pre-calibration multiplier of 0.76295 (50000/65535) converts the measured digit range to digit values 0 – 50000. The digit values 0 – 50000 are directly relating to the measurement made at the analog sensor and are sent via the CAN bus to the data logger.

The CAN bus address IDs for all Moto2™ datarecorder devices are explained in section 4.3.3.

The electrical signals input to the module by each engine sensor are immediately converted to 'Raw Digits'. Next a "pre-calibration" stage occurs; converting the 'Raw Digits' into **fixed point digits** scaled between 0 and 50000.

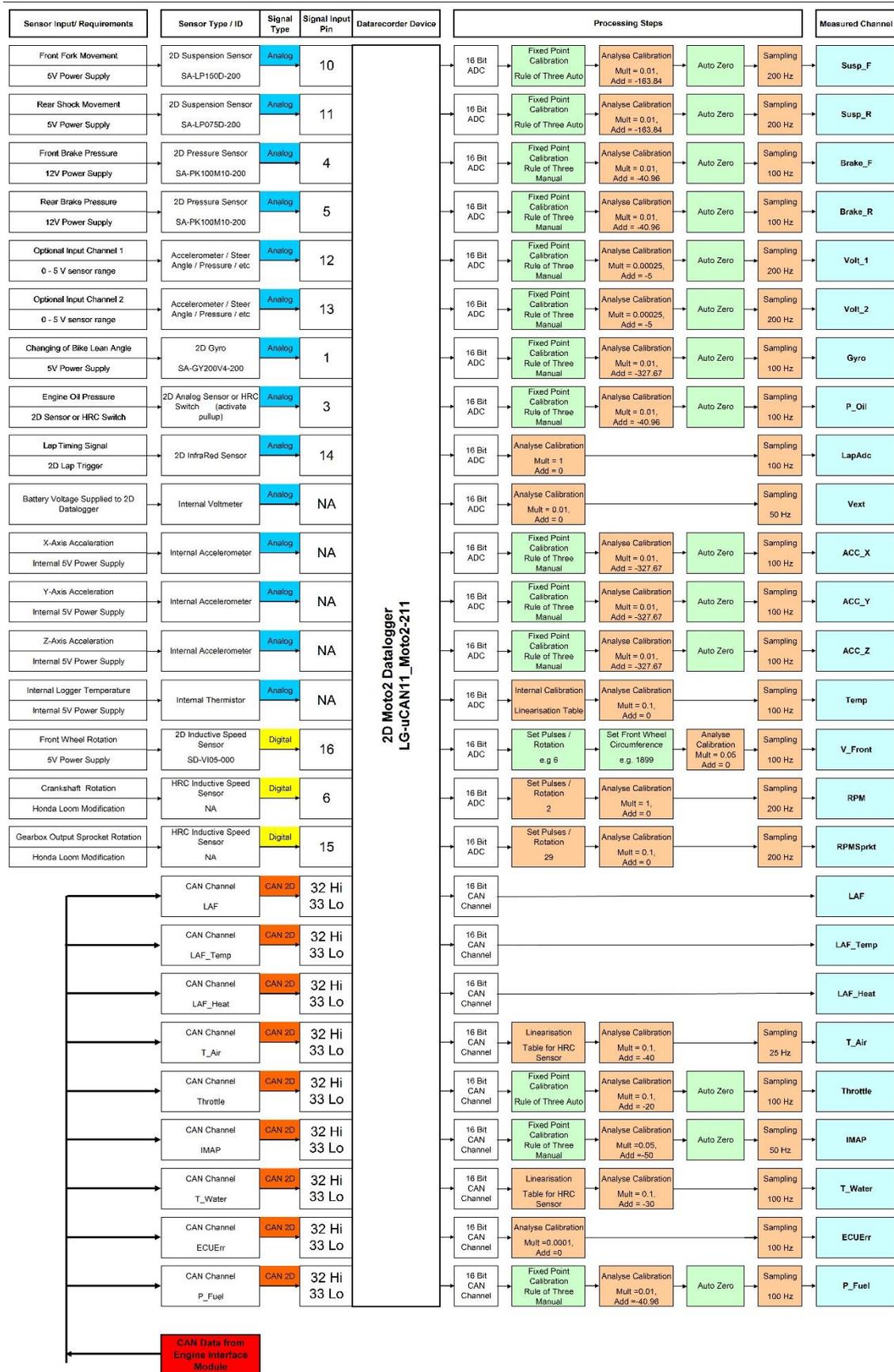
Due to this pre-calibration, the physical voltage values measured at each analog input are expressed inside the device as digital integer values (whole numbers). The digital (or digit) values are 10000 times greater than the physical voltage generated by the sensor.



E.g. if the throttle sensor makes a signal of 3.211 Volts, this pre-calibration process will generate a digital number of 32110 digits and send this on the CAN bus for that channel. With this information available on the CAN bus and the sensor properties being known, other CAN devices can understand the position of the throttle.

### 4.3.2 Moto2™ data logger

The stages for generation of each data channel are shown below:



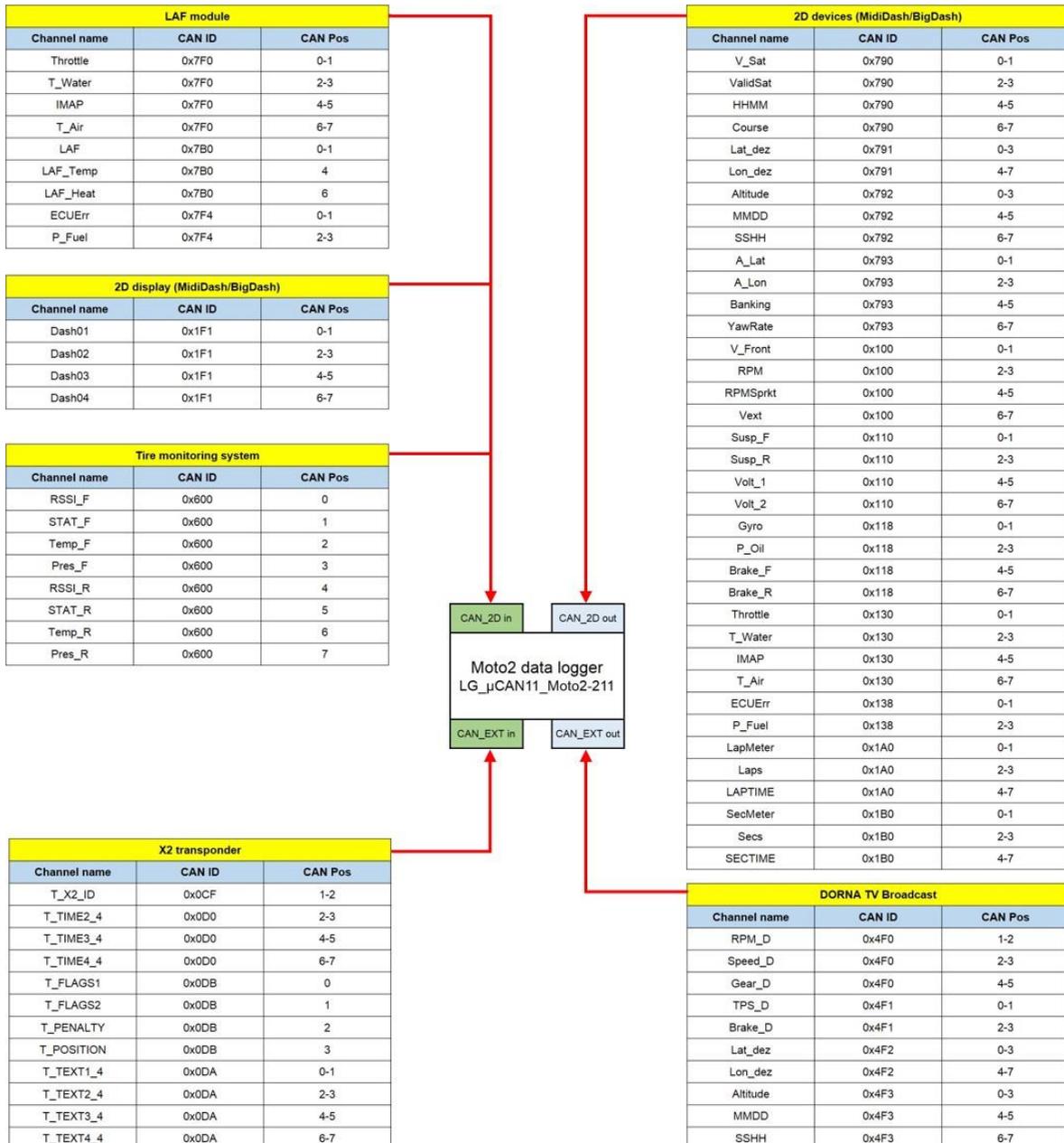
### 4.3.3 CAN IDs for Moto2™ data recording system

For a data channel to be sent via CAN bus from one 2D device to another, the CAN send ID must be correctly defined. Without the correct CAN ID, the data for a channel will go to the wrong location or simply not be sent anywhere!



The CAN ID is defined inside the “Interfaces” node of the device that is sending the data channel.

In principle the operation of a CAN bus is similar to how mail and packages are delivered around the world. Without a valid and correct delivery (send) address, the package will not reach the correct destination! To help avoid problems, 2D have carefully predefined the CAN IDs required for the Moto2™ data recording system. The correct/recommended CAN IDs for each Moto2™ device are listed below.



## 4.4 Detailed info about software for Moto2™ kit system

### 4.4.1 Calculated channels

All the data channels recorded by the Moto2™ system were previously explained. Many of the channels were generated directly by sensors (measured channels) or received from another device via CAN bus (CAN channels). Other recorded channels include Event/ Time/ Calc/ Count.

The software *2D Analyzer* enables many more data channels to be generated after the measurement has been downloaded from the bike. This allows a more extensive analysis of the bikes performance to be made.



It is important not to confuse the “*Calculation Channels*” (calculated after data download) with the “*Calc*” channels (made inside the logger/dash)!

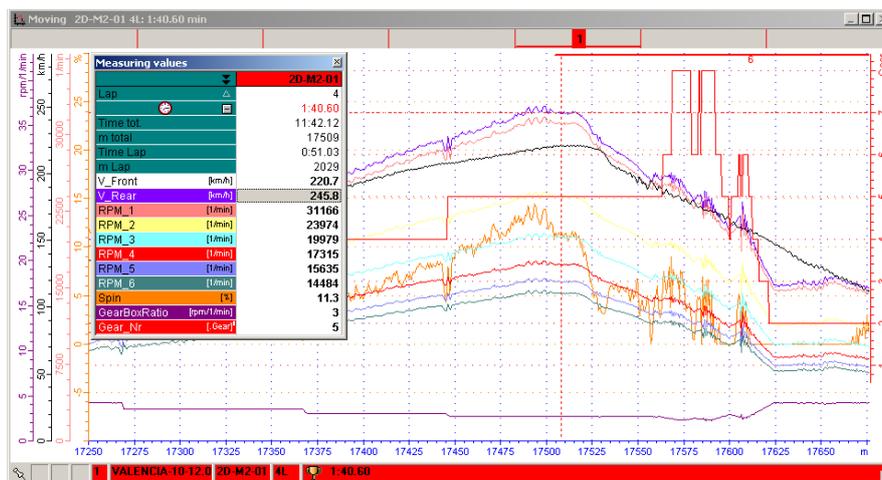


The calculated channels are defined by “*CAL files*” and provide a very powerful tool for best using your bike data. It is possible to analyze many extra channels that could not be directly measured by the datarecorder system

The operation of calculated channels can be summarized as follows:

- The 2D program *CalcTool* is used to define the calculation channels.
- *CalcTool* allows you to prepare channels according to your requirements, giving a very logical format for correctly defining them
- When the entries of the calculations are correctly defined, they are saved as a “*CAL file*” using *CalcTool*.
- The CAL file is used by *2D Analyzer* to generate all the calculated channels after the bike data is downloaded.
- More than one calculated channel can be generated by a single CAL file.
- CAL files can be created from combinations of measured data (downloaded from the bike) and entries of the *SpecSheet* measurement file.
- The calculated channels can be viewed in the program *2D Analyzer* together with the measured channels (downloaded from the data logger).

The screenshot below illustrates the analysis of calculated channels from a Moto2™ bike.



As part of the Moto2™ kit system, a predefined set of CAL files have been prepared by 2D. The default CAL files provide many useful channels for analysis, but are **coded and cannot be changed by the user**. Each of the default calculated channels will be explained in this section.



If you want to have full access for changing and creating CAL files, a full licence of the Moto2™ analysis software must be purchased



The program *CalcTool* will be explained to a basic level, however further information can be found in the 2D *CalcTool* manual. ([2d-datarecording.com](http://2d-datarecording.com))

<support> - <downloads> - <manuals> - <CalcTool>

#### 4.4.1.1 Default calculated channels for Moto2™

Two CAL files named “2D\_Moto2” and “2D\_SuspChannels” contain all the calculated channels defined by 2D for immediate use on your Moto2™ bike.

These CAL files must always be located in your computer directory <C:\ProgramData\RaceMoto2\_14.0\System\Cal><sup>6</sup>; ‘C’ is the partition of your computer where the Moto2™ software is installed. If placed in the wrong location, your CAL files may not operate correctly with the 2D programs *SpecSheet* and *Analyzer*.



The contained calculations of the CAL files are in a coded format and cannot be viewed or modified using the Moto2™ kit software.

The CalFile ‘2D\_Moto2’ defines the following calculated channels:

- **V\_Rear** – rear wheel speed
- **Spin** – rear wheel spin
- **GearRPM** – pseudo value for engine rpm in different gears
- **Distance** – distance travelled by bike during the session
- **used\_gas** – fuel used by bike at that point in session
- **BikeMass** – mass of bike during session
- **GEARRATIO** – total gear ratio of bike
- **GearBoxRatio** – total selected gearing ratio within the engine
- **Gear\_Nr** – calculation for selected gearbox ratio (values 1-6)
- **GearSpeed** – the maximum speed possible (at rev limit) for the selected gear
- **F\_Res** – total resistance force on bike (drag)
- **F\_Eng** – total driving force provided by engine
- **F\_Acc** – resultant force making the bike accelerate
- **T\_wheel** – total torque transmitted at the rear wheel
- **T\_eng** – engine torque (based on force transmitted through drive train)
- **P\_eng** – engine power (based on force transmitted through drive train)
- **T\_eng\_V** – engine torque (based on bike GPS velocity)
- **P\_eng\_V** – engine power (based on bike GPS velocity)
- **Banking\_Angle** – bike lean angle calculated from gyro
- **Acc\_Fax** – applied calibration of Acc\_Fax when connected to channel Volt\_1
- **Acc\_Rax** – applied calibration of Acc\_Rax when connected to channel Volt\_2

The CAL file ‘2D\_SuspChannelsMoto2’ defines the following calculated channels:

- **Susp\_F\_lo** – low speed movement of front suspension
- **Susp\_F\_speed** – velocity of front suspension
- **Susp\_R\_lo** – low speed movement of rear suspension
- **Susp\_R\_speed** – velocity of rear suspension
- **SuspPitch** – pitch angle of the bike

<sup>6</sup> You can reach this folder by selecting *Settings* ⇒ *Folders – Protocol* in the *WinARace* menu. There you select the button <Application data>. In the opened folder you select <System>, <Cal>.

#### 4.4.1.2 CalcTool entries for Moto2™ calculation channels

##### V\_Rear

The rear wheel speed is calculated here. Instead of using a sensor to directly measure the rear wheel speed when the bike is racing, it is determined by calculation. The entries used for the calculation are shown below.

```

3 [V_Rear]
4 C2 = /({#RPMSprkt,1000*1000/60 /Wheels.RCirc *SETTING.R_Sprkt/SETTING.F_Sprkt})
5 C2=Set(DIM='km/h')
6 Result = Word(#C2,0.1)

```

The channel is titled “V\_Rear” and will therefore be viewed in *Analyzer* with the same name. The calculation includes the rotation speed of the gearbox output shaft, measured by the data logger as channel “RPMSprkt”.

In addition the following *SpecSheet* entries for that measurement must be available and valid:

- rear tire circumference in millimeters, (Wheels.RCirc)
- rear axle sprocket teeth number, (SETTING.R\_Sprkt)
- engine sprocket teeth number, (SETTING.F\_Sprkt)

The CAL file also sets the dimension (as viewed in *Analyzer*), to “km/h” and installs to this channel a resolution of “0.1”. This means that the smallest increment of change of the V\_Rear channel when viewed in the program *Analyzer* is 0.1, e.g. 95.1 km/h → 95.2 km/h → 95.3 km/h.

##### Spin

The relative speeds of front and rear wheels are compared here as a ‘percentage difference’. This is very useful for diagnosing the amount of rear wheel spin the bike is experiencing during the session. The entries used for the calculation are shown below.

```

3 [Spin]
4 C1 = -({#V_rear,#V_front})
5 C1 = /({#C1,#V_front})
6 C1 = *({#C1,100})
7 Result = Word(#C1,-32.768,32.767)
8 Result = Set(DIM='%')

```

The speed of the rear wheel (from calculation V\_Rear) is compared to the measured front wheel speed (V\_Front). The channel is viewed in *Analyzer* with the name “Spin”. The calculation is defined as a percentage, where a value of +5% means the rear wheel is rotating (spinning) at a speed 5% higher than the front wheel. Also note that if the rear wheel is blocking (slower than front wheel) the value will become negative!

The CAL file also sets the dimension to “%” and installs to this channel a resolution of “0.01”. As a result a very small change in rear spin (or locking!) can be viewed using the Moto2™ analysis software.

##### GearRPM

If you are analyzing the gear selected by the rider at a particular corner, it may be useful to know what RPM the engine makes if the rider was to use either one gear higher or one gear lower. Six RPM channels (RPM\_1, RPM\_2...RPM\_6) are calculated to help determine the optimum gear selection for each corner of the circuit, the CAL file can be viewed in *2D Analyzer*.

The calculated *GearRPM* channels can be used to quickly check if a higher or lower gear will produce over-revving or perhaps will provide more available power from the engine (if RPM is at a better point of the power-band). The entries used for the calculation are shown below.

```

7 [GearRPM]
8 C1 = I{(#RPMsprkt,DRIVE.PRI)
9 RPM_1 = I{(#C1,DRIVE.1st)
10 RPM_2 = I{(#C1,DRIVE.2nd)
11 RPM_3 = I{(#C1,DRIVE.3rd)
12 RPM_4 = I{(#C1,DRIVE.4th)
13 RPM_5 = I{(#C1,DRIVE.5th)
14 RPM_6 = I{(#C1,DRIVE.6th)

```

This calculation requires a data input for the rotation speed of the gearbox output shaft, provided by the data logger as the measured channel “RPMsprkt”.

In addition the following *SpecSheet* entries for that measurement must be available and valid:

- engine primary drive ratio (DRIVE.PRI)
- gearbox 1st gear ratio (DRIVE.1st)
- gearbox 2nd gear ratio (DRIVE.2nd)
- gearbox 3rd gear ratio (DRIVE.3rd)
- gearbox 4th gear ratio (DRIVE.4th)
- gearbox 5th gear ratio (DRIVE.5th)
- gearbox 6th gear ratio (DRIVE.6th)

### Distance

The distance travelled by the bike is calculated here. The entries used for the calculation are shown below.

```

3 [Distance]
4 Result = I{#V_GPS)

```

‘V\_GPS’ is the only channel required for the distance calculation and a simple integration of this channel generates a travelled distance. The units for ‘V\_GPS’ are meters per second (m/s). The calculated channel “distance” has units of meters (m).

### BikeMas

The total mass of the bike during the session is calculated here. The entries used for the calculation are shown below.

```

7 [BikeMass]
8
9 c1 = If (@Acc_lon,<,10000,Mass.Bike+Mass.Rider+Mass.Fwheel+Mass.Rwheel+Mass.Gas,0)
10 c1 = Set(DIM='Kg')
11
12 used_Gas = *(#Distance,0.15*0.75*0.001) ; 0.15 Liter per kilometer,
13 ; 0.74kg per liter, 1000 m per kilometer
14 used_Gas = Set(DIM='Kg')
15 C2 = -(#c1,#used_Gas)
16 C2 = Freq{#c2,12.5)
17 BikeMass = Word{#c2,0,327.67) ; current Bike Mass (bikemass + FuelIn - Fuelused)

```

By adding all the start masses together, then subtracting the mass of fuel used to that point in the session, an estimate of that bike mass during the session is provided for analysis. Using an estimated fuel consumption of 0.15 liters per kilometer and the distance travelled by the bike so far in the session, the mass of fuel consumed by the bike is estimated.

The measured channel for longitudinal acceleration (Acc\_lon) and calculated channel ‘Distance’ are used in this calculation.

In addition the following *SpecSheet* entries for that measurement must be available and valid:

- bike mass, (Mass.Bike)
- rider mass, (Mass.Rider)
- front wheel mass, (Mass.Fwheel)

- rear wheel mass, (Mass.Rwheel)
- fuel mass, (Mass.Gas)

### used\_gas

As explained above, the mass of fuel used by the bike so far in the session is calculated. This can be viewed as an individual channel in *2D Analyzer*.

### GEARRATIO

This calculates the total gearing ratio made between the engine (crank shaft rotation speed) and the rear wheel (rotation speed of rear wheel). The entries used for the calculation are shown below.

```

3 [GEARRATIO]
4 C1 = /(@Drive_Speed,(3.6 / 60 * Wheels.RCirc /1000))
5 Result = /(@Drive_RPM,#C1)
6 Result = Set(DIM='')

```

This calculation requires input from the channels '@Drive\_RPM' and '@Drive\_Speed'. Also a valid *SpecSheet* entry must exist for rear wheel circumference (*Wheels.RCIRC*), making possible the conversion from road speed to rate of wheel rotation (*RPM*). This channel is 'dimensionless' as it is by definition a ratio!

### GearBoxRatio

This calculates the gearing ratio made only inside the engine. The entries used for the calculation are shown below.

```

8 [GearBoxRatio]
9 C1 = /(@Drive_RPM,#RPMSprkt)
10 Result = Word(#C1,0.001)

```

The calculation divides the engine rotation speed (*@Drive\_RPM*) by the rotation speed of the engines gearbox output shaft (*#RPMSprkt*). This channel is also dimensionless, the same as *GEARRATIO*.

The calculation can be viewed in *2D Analyzer* with a resolution of 0.001, giving a very clear indication of the gear ratio made by the transmission system of the engine.

### Gear\_Nr

Here the gear selected by the bike is estimated by calculation. The entries used for the calculation are shown below.

```

8 [Gear_Nr]
9 ; take GearBoxRatio, calculated in 2D_EngineTorque
10 ; it is represented with 0.001 resolution.
11 ; so 2.8 is represented as 2800 to index into the table
12 ; us this as index into the gearbox table
13
14 C1=F(#GearBoxRatio, F(AVG(50)))
15 Result=F(#C1, T(Gear2010))
16 Result=Set(DIM='.Gear')

```

Using the previously explained calculation "#GearBoxRatio" as input and Moto2™ gear table "Gear2010" as a value lookup table, the currently selected gear can be determined by the calculation. Note that this approach has limited accuracy under braking as the effect of the slipper clutch will make the channel "#GearBoxRatio" become false!

### GearSpeed

This is a calculation of the maximum speed that can be attained using the currently selected gear (as estimated by the calculation "#Gear\_Nr"). The entries used for the calculation are shown below.

```

12 [GearSpeed]
13 C1 = F(@Drive_Speed,F{med 5})
14 C2 = F(@Drive_RPM,F{med 5})
15 C4 = *(DRIVE.GearRPM,#C1)
16 C5 = /(#C4,#C2)
17 C5 = Set(DIM=KMH)
18 Result = Word(#C5,0,350)

```

This calculation requires input from the channels '@Drive\_Speed' and '@Drive\_RPM'; both of these have a median filter applied in order to remove unwanted "spikes" in their data. Also a valid *SpecSheet* entry must exist for the desired shift rpm (or maximum engine rpm), this should be in the field 'Drive.GearRPM' in your permanent info (*SpecSheet*) file.

### F\_Res

The total resistance force acting on the bike is calculated here. This is the resistance force that prevents the bike from accelerating properly, and is mainly due to aerodynamic drag. The entries used for the calculation are shown below.

```

3 [F_Res]
4 C2=*(#V_GPS,0.014)
5 C3=+(#C2,0.00022)
6 C4=*(#C3,#V_GPS)
7 C5=+(#C4,7.5)
8 C6=*(#C5,-1)
9 C6=Set(DIM='N')
10 Result=Word(#C6,-3275.7,3275.6)

```

The calculation is performed using drag coefficient data taken from a 2D test motorcycle. The only channel required for input is the measured channel 'V\_GPS'.

### F\_Eng

The total force generated by the bike to make it accelerate (or decelerate) is calculated here. The entries used for the calculation are shown below.

```

19 [F_eng]
20 C1 = Freq(#BikeMass,100)
21 F_acc = *(@ACC_lon,#C1)
22 F_acc = Set(DIM='N')
23 C1 = Freq(#F_res,100)
24 F_eng = -(#F_acc,#C1)

```

Initially the mass of the bike (from calculation "BikeMass") is multiplied by the bikes longitudinal acceleration (@ACC\_lon). The result of this gives the force making the bike perform the measured acceleration. Next the bikes total resistance force (from calculation "F\_res") is subtracted from the value of the acceleration force. However the resistance force is always negative and so this is actually calculating the total force made by the bike to both accelerate and to overcome the wind resistance drag. When braking, a negative value will be made, giving an indication of the deceleration force made by the brakes! The channel 'F\_eng' can be viewed in *Analyzer* the same as any measured channel.

### F\_Acc

As explained above, the force making the bike accelerate is determined as part of the *F\_Eng* calculation. This can also be viewed as an individual channel in *Analyzer*.

### T\_wheel

This is a calculation for the torque transmitted at the rear wheel of the bike. The calculation simply takes the channel 'F\_eng' and divides it by the radius of the rear wheel to convert the calculated driving force into a driving torque made by the wheel of the bike. The entries used for the calculation are shown below.

```

33 [T_wheel]
34 T_wheel = *(#F_eng,Wheels.RCirc/2/Pi/1000)
35 T_wheel = Set(DIM='Nm')
36 T_eng = /(#T_wheel,#GearRatio)

```

### T\_eng

As shown above, the calculation of engine torque 'T\_eng' is made in the same CAL file section as 'T\_wheel'. The calculation simply involves dividing the result of T\_wheel by the total gear ratio of the bike (from calculation "GearRatio"). This gives an indication of the torque actually made by the engine.

### P\_eng

Using the calculation above for engine torque (T\_eng), the power output of the engine is calculated. The entries used for the calculation are shown below.

```

45 [P_eng] ; Power calculated By Engine RPM
46 ; W = P * T [ Nm] = NM/s * s
47 ; Pe = (M * n) / 9554 [KW = Nm * 1/Min] 60*1000/(2*3.14)=9554
48 ; 9554 = (60 * 1000)/(2*Pi)
49 ;C1 = *(#P_eng,((60 * 1000) / (2 * Pi)))
50 C1 = /(#T_eng,((60*1000)/ (2 * Pi)))
51 Result= *(#C1,@Drive_RPM)
52 Result= Set(DIM='KW')

```

This calculation requires input from the channel '@Drive\_RPM', which is converted from rpm to radians per second, before multiplying by 'T\_eng' to generate a value for the engine power.

### P\_eng\_V

An alternative method for calculating the power transmitted by the bike is to multiply the speed of the bike (in units of m/s) by the previously explained calculation for total driving force 'F\_eng'. The entries used for the calculation are shown below.

```

39 [P_eng_V] ; Power calculated by GPS_Speed
40 ; P = F * v : [N * m/s]
41 C1 = /(#V_GPS,3.6 * 1000) ; Speed in m/s (1000 wg. KiloWatt)
42 Result = *(#C1,#F_eng)
43 Result = Set(DIM='KW')

```

For the calculation of P\_eng\_V, the speed of the bike is provided by the measured channel 'V\_GPS'.

### T\_eng\_V

Using the calculation above for bike power (P\_eng\_V), the torque output of the engine is calculated. The entries used for the calculation are shown below.

```

54 [T_eng_V]
55 ; W = P * T [ Nm] = NM/s * s
56 ; Pe = (M * n) / 9554 [KW = Nm * 1/Min]
57 ; 9554 = (60 * 1000)/(2*Pi)
58 ;C1 = *(#P_eng,((60 * 1000) / (2 * Pi)))
59 C1 = *(#P_eng_V,9554)
60 Result= /(#C1,@Drive_RPM)

```

The engine torque is calculated by multiplying 'P\_eng\_V' by a constant that accommodates the different units of '@Drive\_RPM', then dividing by the channel '@Drive\_RPM'.

### Banking\_Angle

Here the lean angle of the bike is calculated from the measured channel "Gyro". You must have fitted a 2D gyro sensor to the bike for this calculation to be functional!

The gyro sensor measures the rate of change for lean angle. Therefore the actual lean angle of the bike can be determined by applying a special integration to the value of measured channel "Gyro".

2D have defined a special “Banking” function for the estimation of bike lean angle. In keeping with the normal integration process, a “reference value” is required to define the lean angle value at the beginning of the integration process. If this is missing the calculation is not accurate! The entries used for the calculation are shown below.

```

3 [Banking_AngleNoOffset]
4 IfExists(#Gyro)
5 IfNotSpecValueExists(3ax.Bank_Offset)
6 Banking_Angle = Banking(#Gyro)
7 Banking_Angle = Set(DIM=deg)
8 C1 = *(-1, #Banking_Angle)
9 3ax.Bank_Offset = AvgValue(#C1, 100)
10
11 [Banking_AngleWithOffset]
12 IfExists(#Gyro)
13 IfSpecValueExists(3ax.Bank_Offset)
14 Banking_Angle = Banking(#Gyro, 3ax.Bank_Offset)
15 Banking_Angle = Set(DIM=deg)

```

The first stage of calculation determines the bike lean angle if no *SpecSheet* value exists for the integration “reference value” (*3ax.Bank\_Offset*). The banking function determines the bike lean angle, but with no defined “reference value” the calculated lean angle will not be correct.

The average value of this lean angle calculation is determined for the duration of the lap using the function “AvgValue”. The average value is automatically saved to the *SpecSheet* file under the entry “3ax.Bank\_Offset”, where it can be used for the next stage of banking angle calculation!

Next the same process is repeated where the 2D banking function is used, only in this case the *SpecSheet* value for average bike lean angle is used as the integration reference value! This enables a much more accurate calculation of bike lean angle and this is then viewed in *Analyzer* with units of “deg” for degrees.

### Acc\_Fax and Acc\_Rax

The function of measured channels Volt\_1 and Volt\_2 has been kept open so that teams have options for what type of sensors they employ on the bike. These channels simply measure the 0-5 V output of the connected sensor. It is therefore necessary for the voltage measurement to be converted to the physical value experienced by the sensor – this is calibration!

```

3 [Volt2Acc_1]
4 IfExists(#Volt_1)
5 Acc_Fax=#Volt_1, 4)
6 Acc_Fax=Set(DIM=G)
7 Hide(#Volt_1)
9 [Volt2Acc_2]
10 IfExists(#Volt_2)
11 Acc_Rax=#Volt_2, 4)
12 Acc_Rax=Set(DIM=G)
13 Hide(#Volt_2)

```

If the Moto2™ wheel acceleration sensors are connected to Volt\_1 and Volt\_2, the Moto2™ kit software automatically makes correct sensor calibrations with the calculations on previous page.

### Susp\_F\_lo and Susp\_F\_speed

Under the heading “Suspension\_front” the calculations for *Susp\_F\_lo* and *Susp\_F\_speed* are made.

For *Susp\_F\_lo* the measured channel for front suspension position is modified to eliminate movement frequencies higher than 3 Hz. This channel is therefore known as the low speed suspension movement. This is useful for understanding the overall ‘posture’ of the bike as it is braking, cornering and accelerating. The units of this calculation channel remain as the measured channel, in mm.

```

8 [Suspension_front]
9 IfExists(@Front_Susp)
10 C1 = *(@Front_susp, 1)
11 Susp_F_lo = F(#C1,F(L_PHANN3HZ77))
12
13 ; generate suspension movement Speed
14 C1 = *(@Front_susp, 1)
15 C1 = F(#C1,F(L_PHANN20HZ77))
16 C1 = F'(#C1)
17 C1 = Freq(#C1,100)
18 Susp_F_speed = F(#C1,F(L_PHANN20HZ77))

```

For *Susp\_F\_speed* the measured channel for front suspension position is modified to first eliminate movement frequencies higher than 20 Hz and then perform a time based derivation on the result to determine the speed of the suspension movement. The units for this channel are in mm/s. Note that the suspension position changing in the extending position (relative position gets smaller) the calculated *Susp\_F\_speed* value will be negative.

### Susp\_R\_lo and Susp\_R\_speed

Under the heading “*Suspension\_rear*” the calculations for *Susp\_R\_lo* and *Susp\_R\_speed* are made.

For *Susp\_R\_lo* the measured channel for rear suspension position is modified to eliminate movement frequencies higher than 3 Hz. This channel is therefore known as the low speed suspension movement. This is useful for understanding the overall ‘posture’ of the bike as it is braking, cornering and accelerating. The units of this calculation channel remain as the measured channel, in mm.

```

20 [Suspension_rear]
21 IfExists(@Rear_Susp)
22 C1 = *(@Rear_susp, 1)
23 Susp_R_lo = F(#C1,F(L_PHANN3HZ77))
24
25 ; generate suspension movement Speed
26 C1 = *(@Rear_susp, 1)
27 C1 = F(#C1,F(L_PHANN20HZ77))
28 C1 = F'(#C1)
29 C1 = Freq(#C1,100)
30 Susp_R_speed = F(#C1,F(L_PHANN20HZ77))
31

```

For *Susp\_R\_speed* the measured channel for rear suspension position is modified to first eliminate movement frequencies higher than 20 Hz and then perform a time based derivation on the result to determine the speed of the suspension movement. The units for this channel are in mm/s. Note that the suspension position changing in the extending position (relative position gets smaller) the calculated *Susp\_R\_speed* value will be negative.

### SuspPitch

Using the previously calculated channels *Susp\_F\_lo* and *Susp\_R\_lo*, a value for the pitching angle of the bike is calculated. The lo-speed front suspension position is used as an estimate for the front wheels vertical position.

```

1 [SuspPitch]
2 ;assuming a orthogonal model for front And rear wheel travel
3 IfExists(#Susp_F_lo)
4 IfExists(#Susp_R_lo)
5
6 C1 = /{#SUSP_F_lo,1400} ; 1400 is assumed to be wheel base may be adjusted by customer
7 C1 = Freq(#C1,25)
8 C2 = /{#SUSP_R_lo,1400} ; 1400 is assumed to be wheel base may be adjusted by customer
9 C2 = Freq(#C2,25)
10
11 C3 = *(#C2,2.1) ; Assumed Link ratio rear shock to rear axel (most bikes are around this value)
12 C7 = ArcTan(#C1)
13 C5 = ArcTan(#C3)
14 C6 = -{#C7,#C5}
15 C6 = Set(DIM='°')
16 Result = *{#C6,@RAD2DEG} ; pitch angle. 0 is assumed to be with both wheels full extended

```

Rear wheel vertical position is calculated using an estimated linkage ratio of 2.1. With values for the relative positions of both front and rear wheels, rules of geometry are used to define the relative pitching angle of the bike. Note that pitching angle will read zero when the front and rear suspension positions are at zero (their calibration reference points). The units for the *SuspPitch* calculation are degrees.



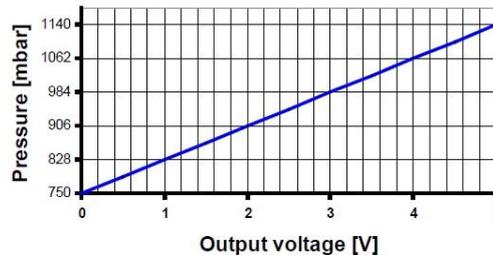
This completes the explanation of the calculated channels automatically created by the 2D Moto2™ kit system.

#### 4.4.1.3 Making your own CAL files for analog channels Volt\_1 and Volt\_2

This quick guide provides the necessary information to customize a CAL file for applying the correct physical value and units to any sensor connected to analog channels Volt\_1 or Volt\_2.

##### Example 1: Air box pressure sensor connected to Volt\_1

The first step is to view the datasheet for the sensor you are connecting. First make sure that the electrical connections of the sensor are correct, it is very important that the sensor receives the correct supply voltage (5 V or 12 V). The datasheet will also explain the calibration curve of the sensor, indicating the voltage made by the sensor across the physical measurement range as shown below.



With reference to the air box pressure sensor, the datasheet tells us that the minimum physical measurement is 750 mbar, while the maximum is 1140 mbar. Across this physical measurement range, the voltage output of the sensor increases from 0 V to 5 V in a linear way. Therefore the sensor makes 0 V at 750 mbar and 5 V at 1140 mbar.

This provides the necessary information for making the CAL file, as shown below.

```

1 ;last modified 02/25/2011 8:40:25 PM by MC
2
3 [P_Air]
4 IfExists(#Volt_1)
5 C1 = / (#Volt_1,5) ;Divide measured voltage value by voltage measurement range (5volts)
6 C2 = * (#C1,390) ;Multiply by physical measurement range of sensor (1140mb - 750mb = 390mb)
7 C3 = + (#C2,750) ;Add lower physical value, the measured value when sensor makes 0V (750mb)
8 Result = Word(#C3,500,1500) ;Set data range of new data channel (500mb to 1500mb)
9 Result = Set(DIM='mbar') ;Set units to "mbar"

```

#### Notes:

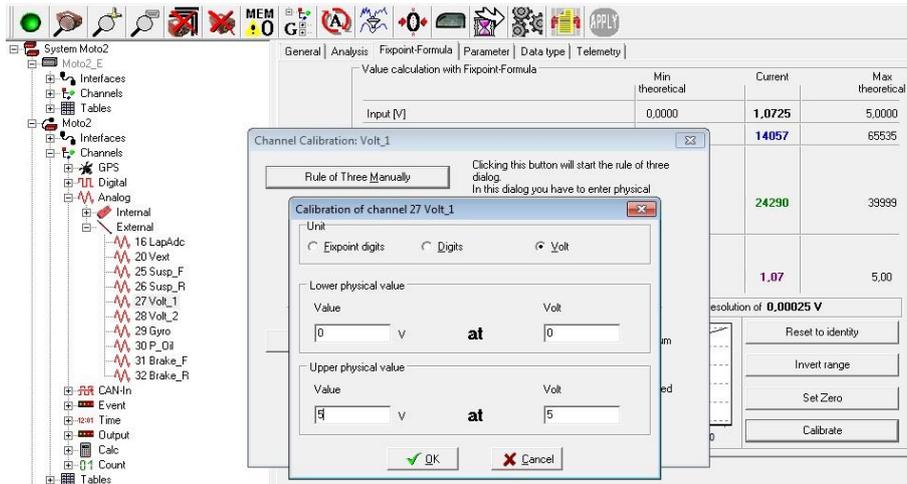
The notation *[P\_Air]* announces that the result of this set of calculations will be named “P\_Air”. The “*IfExists*” notation ensures that the calculation only occurs if the channel “Volt\_1” is recorded. Initially the value of Volt\_1 is divided by the voltage range (5 V as indicated by the datasheet). If you are connecting the sensor to Volt\_2, simply type Volt\_2 instead of Volt\_1!

Next we multiply by the physical measurement range (1140 mbar – 750 mbar = 390 mbar), then add the lower physical value (750 mbar). The result is set with a data range between 500 mbar and 1500 mbar, providing a resolution of (1500-500)/65535 = 0.01526 mbar. Finally calculation is completed by defining the dimensions “mbar”; these will be viewable inside *Analyzer*.

### Calibration steps:

Some basic steps must be taken to correctly configure the channel Volt\_1 (or Volt\_2). Connect the Moto2™ data logger to your PC and open the program *WinIt*. Inside the system tree of the data logger you must locate the channel Volt\_1, which can be found inside **Channels** → **Analog** → **External**.

- In the tab **General**, make sure the channel is recording by ticking the appropriate box.
- Open the tab **Fixpoint-Formula**
- Select **“Calibrate”**, then click **“Rule of Three Manually”**
- Input the **“Lower physical value”** as 0 V
- Input the **“Upper physical value”** as 5 V



This completes the configuration of the P\_Air CAL file.

### Example 2: Steering position sensor connected to Volt\_2

Next we will consider the example of a linear potentiometer mounted to the steering damper in order to measure the steering position of the bike. Various sensor types are possible, but we will assume a 75 mm linear stroke sensor is used<sup>7</sup>.

The first step is to fit the sensor to the bike. Make sure that the sensor can measure the full movement range of the steering without preventing the motion of the steering. With the sensor fitted to the bike, you can try to measure the actual steering angle left and right.

The steering angle may be different from one side to another, so to keep things simple for this example the convention for steering position will be:

- Straight ahead = 0%
- full left turn = -100%
- full right turn = 100%

Using this information we can generate the steering position CAL file, as shown below.

```

1 ,last modified 3/1/2011 11:34:15 AM by IRTA2D
2
3 [Steer_Pos]
4 IfExists(#Volt_2)
5 C1 = / (#Volt_2,5) ;Divide measured voltage value by voltage measurement range (5volts)
6 C2 = * (#C1,200) ;Multiply by physical measurement range (200%)
7 Result = Word(#C2,-100,100) ;Set data range of new data channel (-100% to 100%)
8 Result = Set(DIM='°') ;Set units to "°"

```

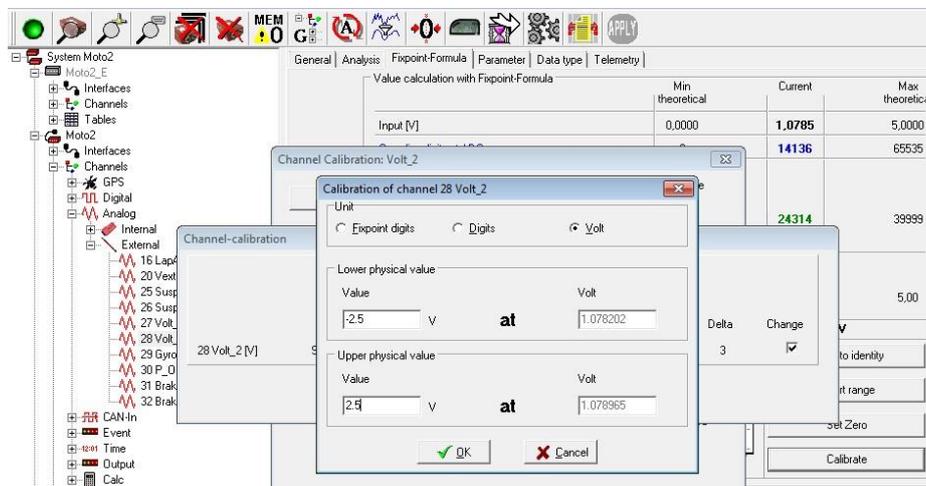
<sup>7</sup> You can find a datasheet on [2d-datarecording.com/en/produkte/hardware/sensoren/sensoren-lange/potentiometer-double-track](http://2d-datarecording.com/en/produkte/hardware/sensoren/sensoren-lange/potentiometer-double-track)

**Notes:**

The notation *[Steer\_Pos]* announces that the result of this set of calculations will be named “Steer\_Pos”. Next we multiply by the physical measurement range (100% - - 100% = 200%). The result is set with a data range between -100% and 100%, providing a resolution of 200/65535 = 0.00305%. Finally calculation is completed by defining the dimension of “%”; this will be viewable inside *Analyzer*.

**Calibration Steps:**

- Make sure the channel Volt\_2 is recording by selecting the tab **General**, and ticking the appropriate box
- Open the tab **Fixpoint-Formula**
- Select “**Calibrate**”, then click “**Rule of Three Automatically**”
- Move the steering position fully left, then click “**Refresh Minimum**”
- Move the steering position fully right, then click “**Refresh Maximum**”
- Click “**OK**” to proceed to the screen shown below



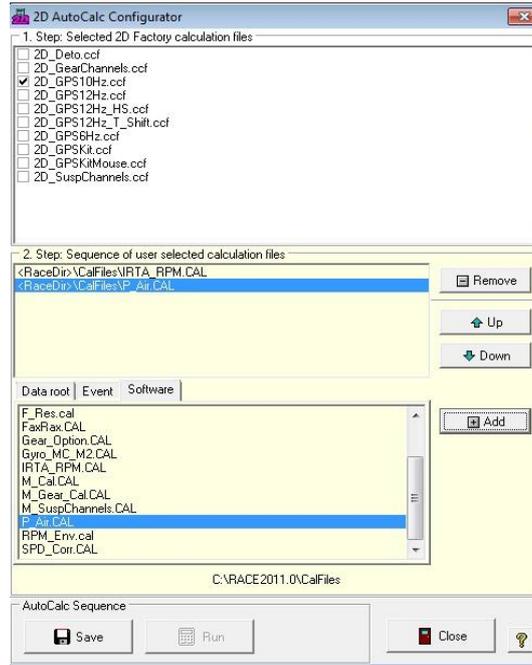
- Input the “**Lower physical value**” as -2.5 V.
- Input the “**Upper physical value**” as 2.5 V.

This completes the configuration of the CAL file for steering position.

**4.4.1.4 Adding a CAL file to the AutoCalc configurator list**

Once the CAL file is defined, you must make sure it is saved and located inside the directory of your other system CAL files. The directory is: **C:\Users\Public\Documents\RaceMoto2\_14.0\CalFiles**<sup>8</sup>. Once the defined CAL file is in the directory, you must ensure that it is actually calculated at the time of download by using the *AutoCalc* configurator, as shown below.

<sup>8</sup> You can reach this folder by selecting *Settings* ⇒ *Folders – Protocol* in the *WinARace* menu. There you select the button **<User data>**. In the opened folder you select **<CalFiles>**.



To view the *AutoCalc* configurator, open *WinARace*, select the button **<Modules>** and click **“AutoCalc Configurator”**. Select the tab **“Software”** and inside this list you should find and select the CAL file **“P\_Air.CAL”**. Once selected, click **“Add”** to place the CAL file **“P\_Air.CAL”** on the list of calculations that will be made at the time of data download. Before exiting, click **“Save”** to complete this stage of the setup.