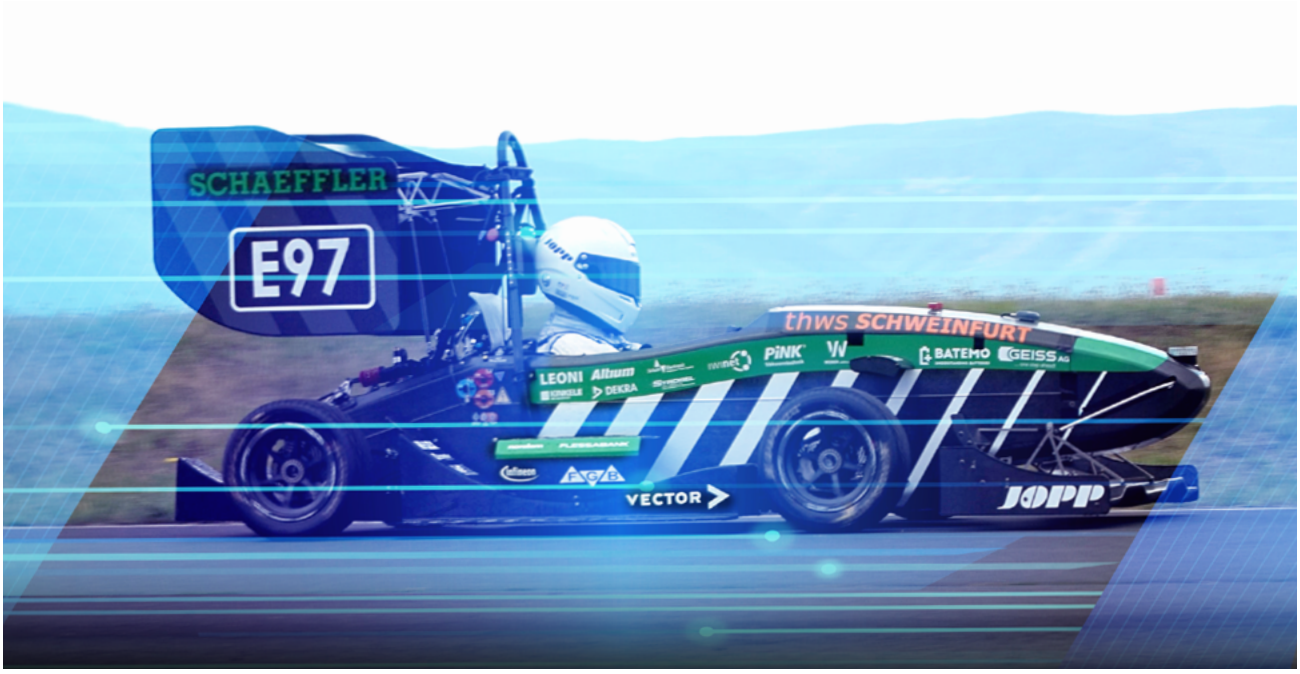
LAUTERBACH
DEVELOPMENT TOOLS




LEADING EDGE ON THE RACE TRACK

# Pole Position in Formula Student with TRACE32® Debug and Trace Tools

*Since 2006, motorsport enthusiasts from the University of Applied Sciences Wuerzburg-Schweinfurt have been successfully developing a new racing car for the Formula Student every year. We have been supporting them in electronics and software development with our TRACE32® debug and trace tools. In this case study, we take a closer look at some of the race car's electronics applications and the practical use of TRACE32®.*

*By Frank Riemenschneider, Senior Marketing Engineer, Lauterbach GmbH*

The Formula Student project at the University of Applied Sciences Wuerzburg-Schweinfurt was launched in 2006 as the brainchild of individual motorsport enthusiasts. Motivated students from various departments formed a team called Mainfranken Racing (MFR). Mainfranken Racing e.V. is a registered, non-profit association based at the University of Applied Sciences Wuerzburg-Schweinfurt [1]. In addition to building an efficient vehicle, the aim of the association is to compete in several international competitions. The individual disciplines are described on the Formula Student website [2].

To achieve this, it is essential to develop a racing car each season that is as light as possible, but at the same time powerful and competitive. To achieve this, the students plan and manufacture the various components over a period of months.

Over the past 18 years, they have implemented many changes and innovations. For example, they raced with a combustion engine for the first 11 seasons and then decided to switch to the pioneering electric version. With the third e-vehicle came the switch to a monocoque made of carbon fiber in order to achieve optimum weight savings in the chassis. A monocoque is a special construction that forms the chassis and replaces the frame. This allows greater stability to be achieved with less mass in the vehicle. However, the production process is very complex and takes several months.

CASE STUDY

The current, fourth vehicle with an electric motor, the MF16, also includes an autonomous system for corresponding competitions.

The team celebrated great successes in the 2024/2025 season: at the beginning of August 2024, the team achieved 3rd place in the driverless disciplines with the autonomous system in the Czech Republic. In France, the team stood on the podium in 2nd place in the overall standings.

## Lauterbach is a Long-standing Partner for Development Tools  Tools

Training and supporting students of engineering and other STEM (Science, technology, engineering, and mathematics) subjects has always been a matter close to Lauterbach's heart. We always offer interested working students the opportunity to work on specific projects in order to get to know the real working world in a market-leading high-tech company. More than once this has resulted in a new employee after graduation. Supporting the MFR team with our tools was therefore an absolute no-brainer.

For central vehicle functions, three of which are explained in detail later in this article (dashboard, battery management system, traction control & slip control), the team opted for microcontrollers from Infineon's XMC family [3]. Infineon is the world leader in automotive semiconductors.

Lauterbach's debug and trace tools under the TRACE32® brand are also global market leaders and are used by all well-known automotive manufacturers, their suppliers and chip manufacturers. To ensure the simplest and smoothest possible use with the XMC chips, we have consequently recommended our µTrace® [4] to MPR, which allows debugging of the Cortex-M CPUs with the highest bandwidth available on the market. In addition to its leading debugging capabilities in the Cortex-M market, µTrace® can record real-time information such as system traces and parallel ETM flow traces, enabling code coverage and code profiling, for example.

## Commissioning of Lauterbach's µTrace Debugger with Evaluation Board for Infineon XMC4700

To test the trace and debug interface of an XMC4700 microcontroller for the first time, the team designed its own evaluation board (Fig. 1a). The purpose of this board was to be able to put the TRACE32® µTrace® debugger into operation as quickly as possible with the given hardware. Figure 1b shows the block diagram of the evaluation board.

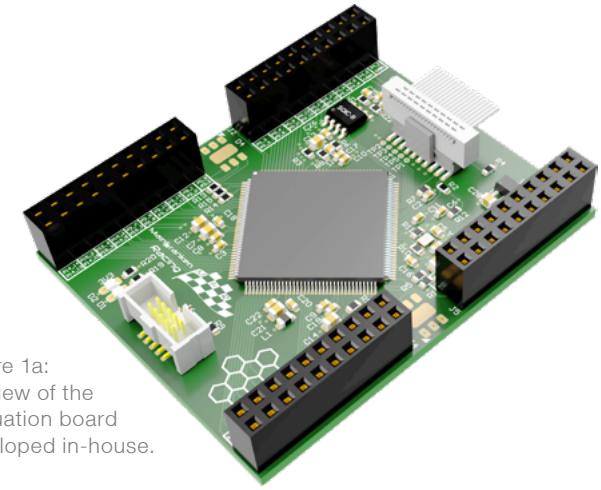


Figure 1a:
3D view of the evaluation board developed in-house.

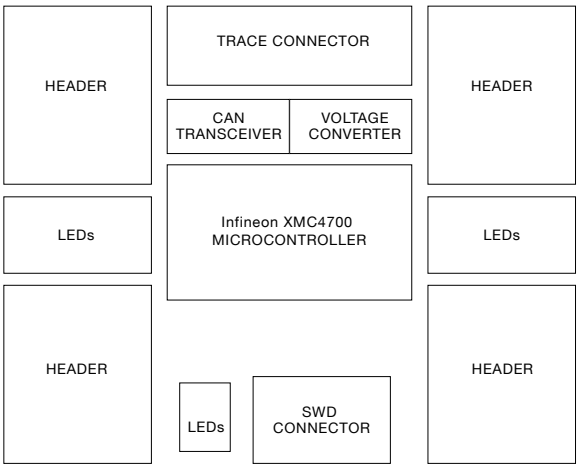



Figure 1b. Block diagram of the evaluation board.

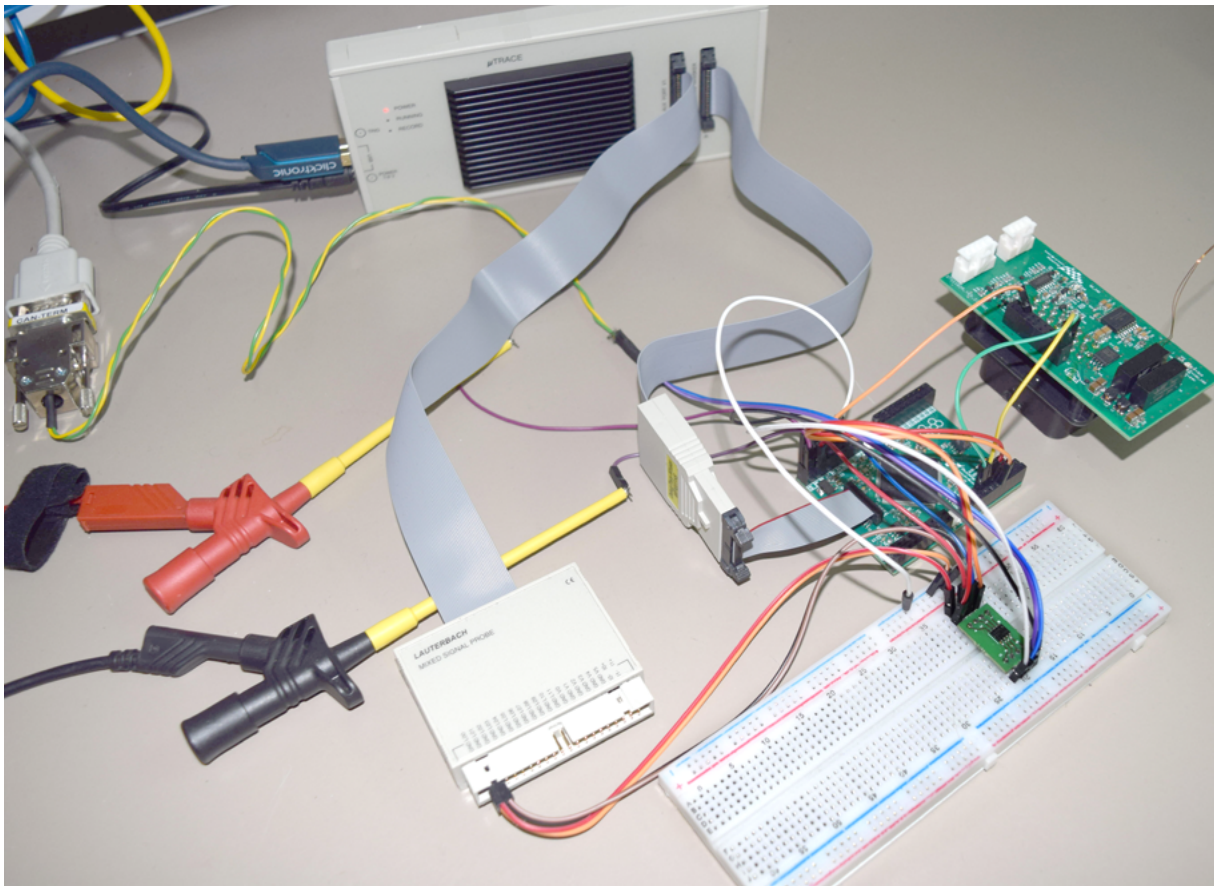



Figure 2: Test setup with self-developed evaluation board.

The complete setup with µTrace debugger and circuit board can be seen in Figures 2. Thanks to the simple configuration of our TRACE32® PowerView software, MPR could quickly put the µTrace® debugger into use with this test setup and the first problems in the software developed by the team were quickly found.

Apart from the µTrace, Figure 2 also shows Lauterbach's Mixed Signal Probe. It allows the recording of digital and analog signals, which can be correlated to the recorded program flow. This way it is easy to verify signal-timings which are initiated by the application software or to calculate the power consumption of specific program parts.

Thanks to the TRACE32® PowerView Software, the MFR team can drive both tools from a single UI with cross-triggering and a common timestamp, allowing the verification of software changes and signal behavior.

The device in the center of Figure 2 is called a whisker. It is the physical link between the Evaluation Board's debug and trace ports and the µTrace. Because the whisker's transceivers are close to the target, it provides the most stable signal at the highest transmission speeds. Each of Lauterbach's whisker models is optimized to perfectly match the physical and electrical specifications of the target's debug and trace ports.

## Use Case Dashboard

Figure 3 shows the dashboard circuit board. The dashboard serves as a central display for a variety of vehicle parameters and statuses, including the battery charge level and the temperature of the cooling circuit. It also offers the option of activating autonomous driving by pressing a rotary knob on the cockpit. The selected driverless mission is signaled by a blue LED and then transmitted to the vehicle via the CAN bus. In addition, the dashboard registers the status of a “Ready to Drive” button and also sends this value to the vehicle via the CAN. Status LEDs and an LC display are mainly used to visualize the data.

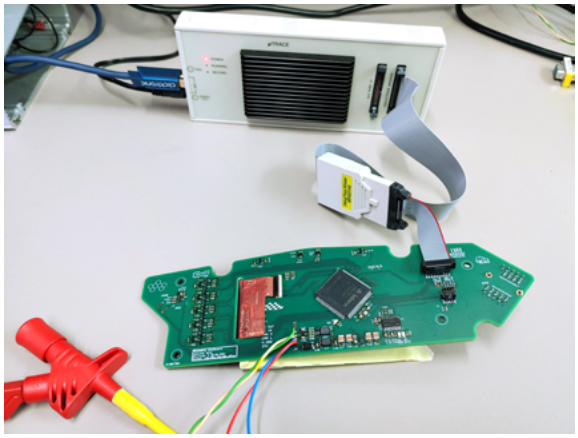


Figure 3: µTrace in use during dashboard development.

The architecture of the dashboard software is shown in Figure 4. The software reads the required information directly from the vehicle CAN via the CAN interface. The hardware interface records the encoder for the driverless missions and the button for the “Ready-to-Drive” function. In the state handler, the retrieved values are assigned to a status LED or converted accordingly so that they can be shown on the LC display. The “Ready-to-Drive” status is sent via CAN to the vehicle and the driver informed via either an LED changing color or new information on the LC display.

The LC display is controlled via SPI, whereby all specific commands for controlling the display are implemented directly in the LC display node. The LVGL graphics library is used to create an appealing and user-friendly user interface. The development process is carried out with the help of GuiGuider, a UI toolbox that enables the intuitive creation of user interfaces. The LVGL-Lib interface serves as a bridge to access the different components of LVGL and GuiGuider. This enables navigation between different screens, the adjustment of vehicle parameters and many other functions.

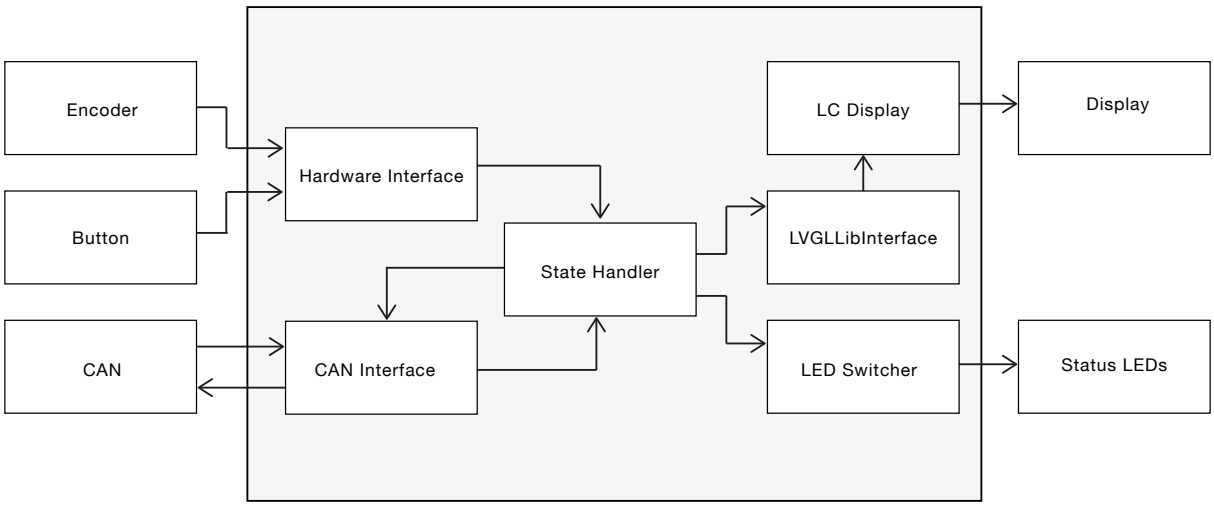



Figure 4: Overview of the dashboard software.

## Application Area of the µTrace Debugger in Dashboard Development

In the software development for the dashboard, the main focus is on the integration of a graphics library with the microcontroller and the system display. The µTrace debugger was able to help find a memory allocation error at dashboard runtime. The error manifested itself in the fact that the software crashed after switching between different pages on the display several times.

Troubleshooting was accelerated by tracing the software sequence and recording it until the software crashed. By analyzing the expired code shortly before the software error, the problem could be narrowed down to a memory problem. In the end, the problem was found in the memory allocation of the graphics library, which uses its own malloc function. The error was rectified by switching to the standard C-malloc function.

## Use Case Battery Management System

Figure 5a shows the control unit of the battery management system (BMS) master. The task of this control unit is to manage the battery management system of the HV battery and to monitor faults within the battery.

The master communicates with the BMS ICs via a UART transceiver using an isolated UART (isoUART) protocol (Fig. 5b). The BMS ICs are connected to the Li-Ion cells of the battery and measure cell voltages and cell temperatures. In addition, discharge resistors can be connected to each individual cell in order to discharge the cells to the same voltage level, known as “balancing”.

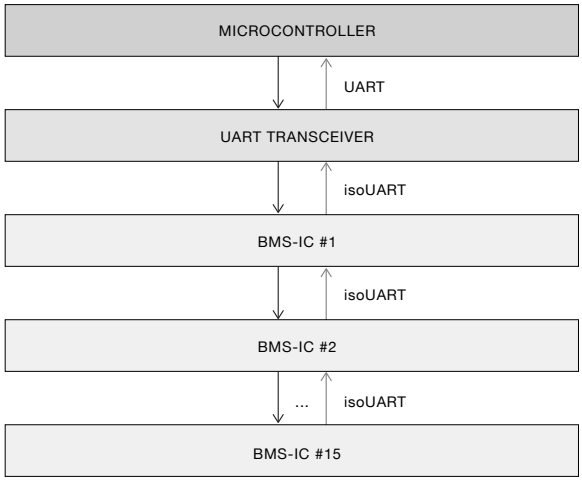



Figure 5b: Communication of the BMS ICs.

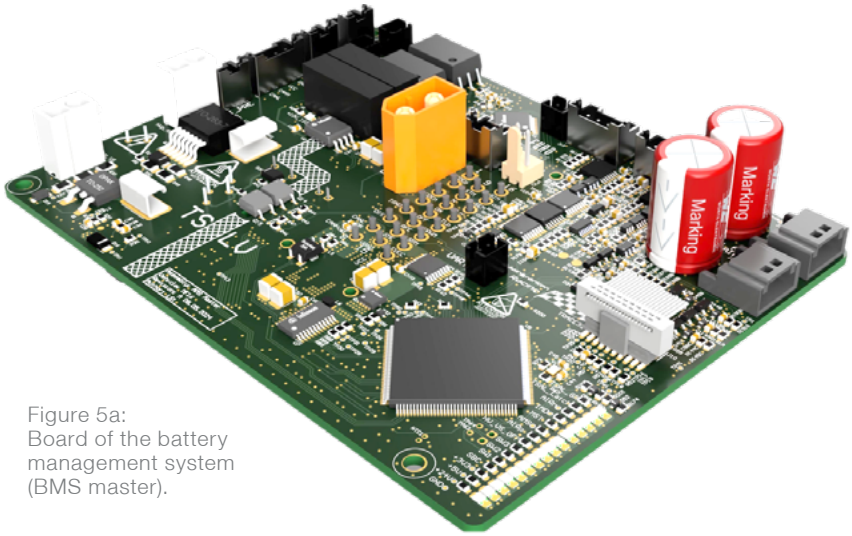



Figure 5a:
Board of the battery management system (BMS master).

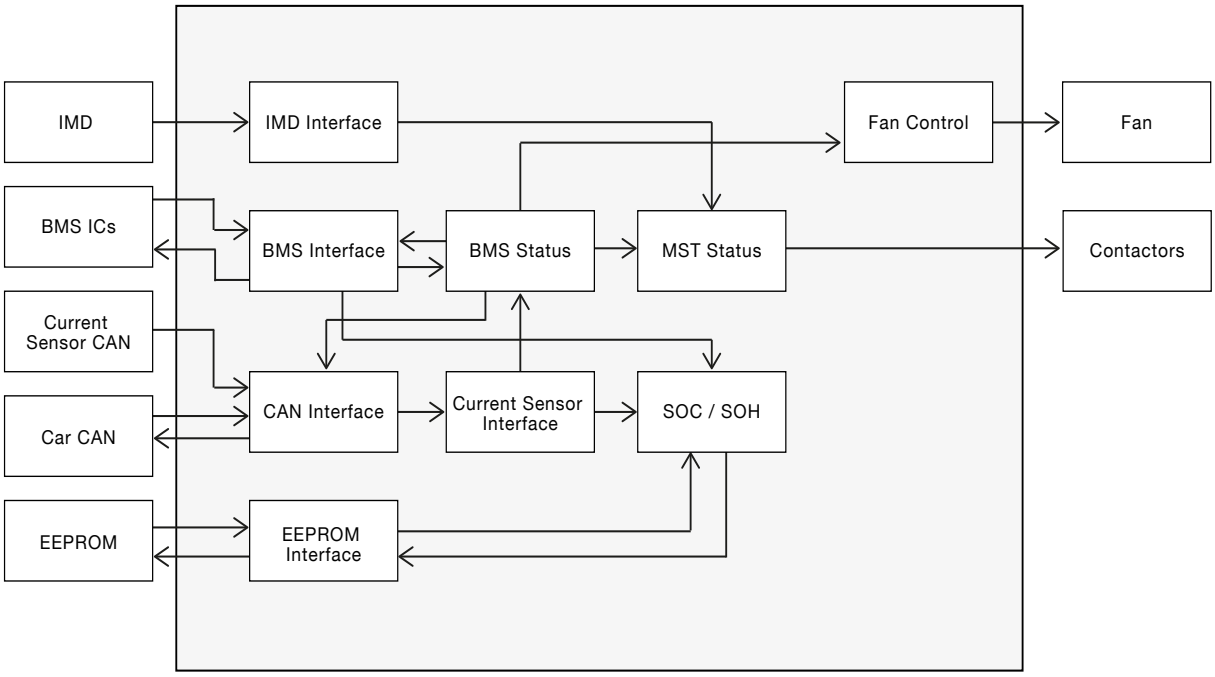



Figure 6: Battery management software modules.

Figure 6 shows the software modules of the battery management system. Inputs to the software are shown on the left. This is the insulation monitoring system (IMD), the BMS ICs are connected via a bus system. The microcontroller is also connected to the vehicle CAN and to a current sensor. An EEPROM is available for storing non-volatile data. The BMS system communicates with the ICs via the BMS interface and thus reads in measurement data such as cell temperatures and voltages. The State of Charge (SOC) and the State of Health (SOH) are calculated from the cell data and the charging/discharging current.

## Application area of the µTrace debugger in BMS development

A large part of the BMS master's software deals with communication with the BMS ICs and processing the measurement data. The BMS ICs are connected to the microcontroller via an isoUART bus. This allows data such as the individual cell voltages and temperatures to be read out and all cells to be discharged to the same voltage level.

The µTrace debugger was used to adjust the communication timings so that all the required data can be queried at the required frequency without overloading the BMS ICs and the isoUART bus.

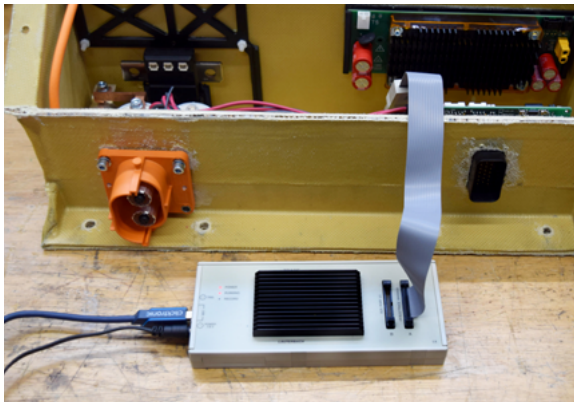


Figure 7: BMS master installed in the battery container.

## Use Case Traction Control and Slip Control

In order to get the greatest possible acceleration out of the Formula Student vehicle, a certain amount of slip on the driven tires must not be exceeded, otherwise the transmitted force and thus the acceleration will be reduced. A controller is required to set this optimum slip.

The input variables of the controller are the vehicle speed and the rotational speed of the driven tires. The variable to be controlled, the slip, is calculated from these. The output variable of the controller is the torque specification for the inverter.

The “ETC” (Electronic Throttle Control) control unit, which is primarily used for reading the accelerator pedal and sending the torque specification, is equipped with an Infineon XMC1404 microcontroller. The software part of the traction control is located on this control unit. The initial parameterization of the controller takes place in the structure shown in Fig. 8.

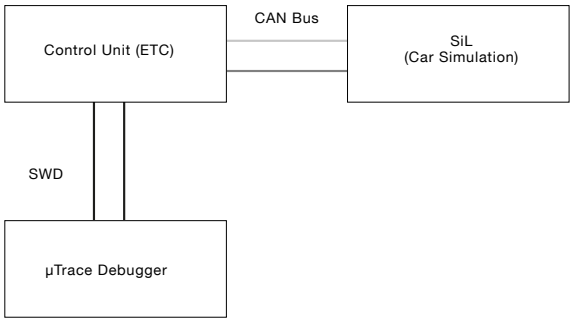



Figure 8: Schematic structure & integration of the debugger into the test setup.

Figure 9 shows a photo of the setup described. Here, the control unit is connected to an SiL system via a CAN bus. The vehicle simulation on the SiL system supplies the input variables for the controller and reacts to the manipulated variable.

The µTrace debugger is connected to the microcontroller via SWD, but without trace functionality, as the XMC1404 does not support this.

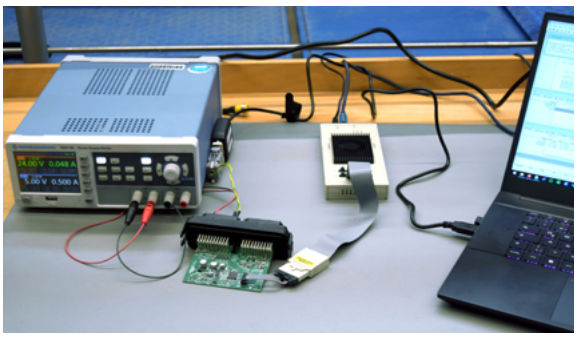


Figure 9: Setup and integration of the µTrace debugger into the test setup.

## Application Area of the µTrace Debugger in ETC Development

The main task of the debugger is to record the internal controller variables and to be able to adjust the controller parameters live. In addition, the utilization of the microcontroller’s computing power is measured in order to find the optimum point between controller computing power and controller performance.

## Conclusion

With our TRACE32-µTrace debugger, the Mainfranken racing team was able to realize and complete the electronics and software development for its new MF16 race car faster and more smoothly than planned, which contributed to the great successes in the 2024/2025 season. We would like to congratulate all the students who contributed to this and are delighted that the collaboration worked smoothly and purposefully at all times. In line with the motto “After the race car is before the race car”, we are excited to see what innovations the MF16 successor, MF17, will bring and how our TRACE32® tools can make a small contribution to leading this vehicle to great success once again.

## Formula Student - a Student Design Competition

Formula Student is a competition in which university teams design and build a single-seater race car. The teams have from October to August to plan and build the vehicle. The teams then compete against each other with their vehicles in various disciplines at different racing events. However, it is not just speed on the specified tracks that determines the overall winner. The so-called static disciplines also play an important role. These include, for example, the presentation of the design and business plan, as well as a cost breakdown.

Before the actual competition begins, the teams must pass the technical scrutineering. For Formula Student Electric vehicles, i.e. with an electric drive train, the electrical safety as well as the technology and safety in general are tested. In addition, the vehicles undergo a static test known as a “tilt table”, a rain test and a brake test. Only when all these tests have been successfully passed are the vehicles admitted to the competition. Further details on the individual disciplines can be found at [2].

References:

[1] Web Page of Mainfranken Racing e.V.: https://mainfranken-racing.de/

[2] Disciplines in the Formula Student: https://www.formulastudent.de/about/disciplines/

[3] XMC Microcontroller from Infineon: https://www.infineon.com/cms/en/product/microcontroller/32-bit-industrial-microcontroller-based-on-arm-cortex-m/

[4] µTrace Debugger from Lauterbach: https://www.lauterbach.com/products/debugger/microtrace