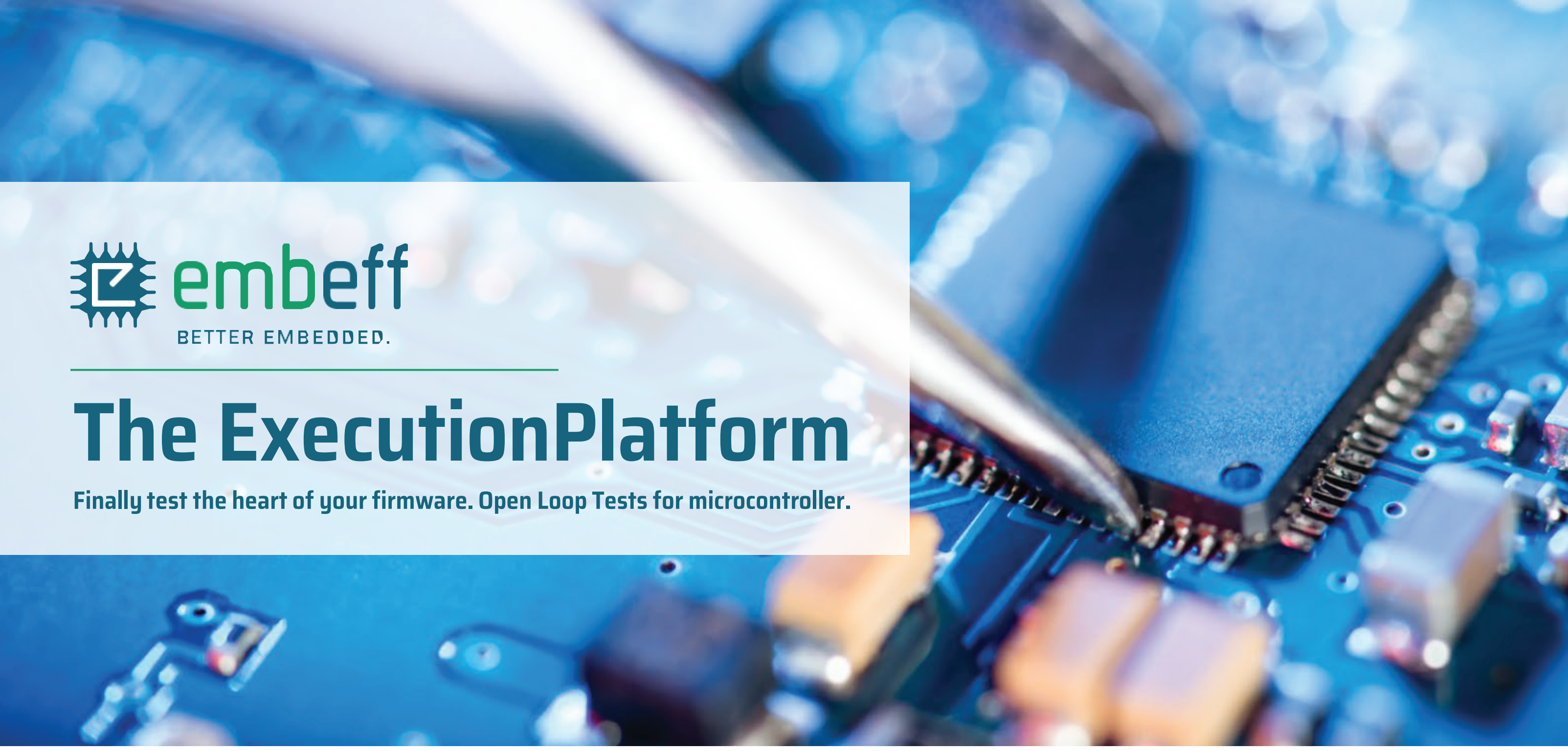


embeff
BETTER EMBEDDED.

The ExecutionPlatform

Finally test the heart of your firmware. Open Loop Tests for microcontroller.

Are you a software developer for microcontroller?

Now, for the first time ever you can test the heart of this software, i.e. the interaction of the microcontroller with its environment. By doing this you can ensure that what you want to happen on the pins happens exactly the way you expect. You can already perform testing during development and testing is fully automated.

Advantage: stable software with a solid foundation which does not cause any surprises at a later date or any delays later on when system testing. Embedded software is characterized by the fact that it interacts with its environment via sensors and actuators. On the microcontroller level, this happens via pins.

Embedded software uses hardware peripherals (for example GPIO, CAN, SPI) for this purpose.
Our test device controls single pins and supports whole protocols like CAN or SPI. This way you can check if your embedded software works correctly with the hardware periphery without any additional lab equipment.

A new way of testing

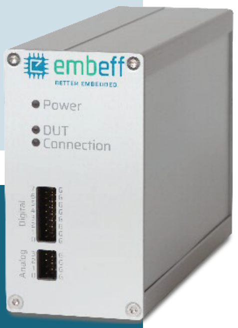
Imagine that you need a driver for your project that generates precise pulses on a pin. Your microcontroller offers timer functionalities for this, which can directly control an output. You want to check if this driver works correctly.

Old way: Manual steps, time-consuming
1. Prepare PCB, power supply, debugger, and oscilloscope.
2. Make the pin accessible and connect it to the oscilloscope.
3. Create special firmware which calls the driver with different lengths.
4. Flash the firmware with a debugger.
5. Debug through the firmware and check with oscilloscope that the generated pulses are within the tolerance.

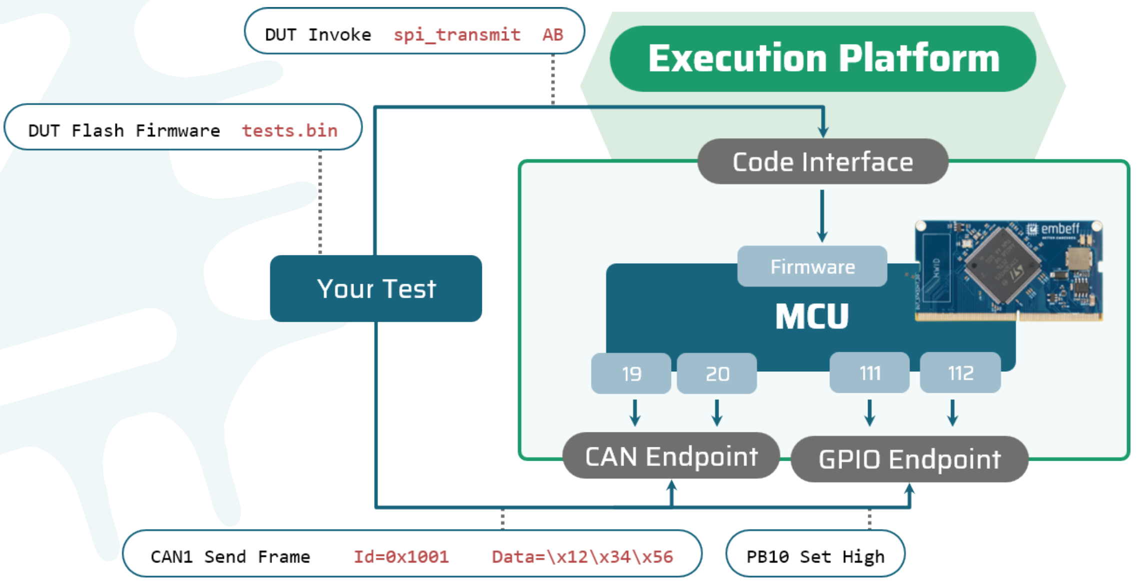
New way: Automated, quick
1. Connect the ExecutionPlatform to your network.
2. Write an open-loop test that calls the driver, performs the measurement on the pin and evaluates the result.
3. Start the test on your ExecutionPlatform.

Typical use cases
- Test drivers in isolation without lab equipment.
- Quickly explore hardware features.
- Easily reproduce failure scenarios.

Your benefits
- Risk minimization through early testing.
- Simplified microcontroller replacement through reusable tests.
- Designed for remote work.
- Low investment costs.



Get your ExecutionPlatform
1. You name us a microcontroller.
2. We produce your ExecutionPlatform with this chip.
3. You get your ExecutionPlatform with full access to all microcontroller pins.



Endpoints

Endpoints are the interface between a test and a specific peripheral (GPIO, CAN, SPI, ...). Each endpoint provides specific functionality. For example, the GPIO endpoint can read and set digital pin levels or perform timing measurements. The CAN endpoint can send and receive frames and trigger extensive bus errors. Endpoints can be configured for one or more pins. All the pins of your microcontroller are available for this purpose. Checking or stimulating pins is traditionally done manually via an oscilloscope, logic analyzer and signal generator. The functions of these devices are already integrated with the endpoints. No more devices are required. Visit our website for an up-to-date list of supported Endpoints.

Code Interface

Functions on the microcontroller are called at test time via the code interface.
For this purpose, functions are registered in the firmware. These functions can then be called conveniently from the test sequence when required. A debug probe is integrated in order to put the firmware onto the installed microcontroller.

Example

The ExecutionPlatform is used via the Robot Framework. This automation framework is open source and industry-proven. Endpoints and code interface are accessed directly from the Robot Framework.
In the example below, the create_pulse function is to be tested. This is to create a precise pulse on pin PB10. For the test, you have configured a GPIO endpoint on this pin.

The test

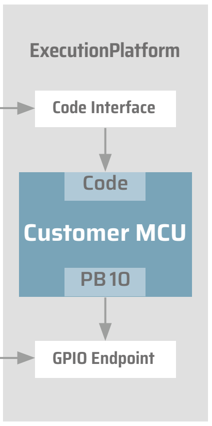
1. starts a time measurement via the GPIO endpoint,
2. calls the function on the microcontroller via the code interface,
3. reads the measured time via the GPIO endpoint,
4. compares the measured time with the expected time.

```
*** Test Cases ***
Long 7ms Pulse Is Within 1% Tolerance
  # 1. Prepare a measurement of a single pulse on PB10
  GPIO Measure Between Rising And Falling Edge  PB10  PB10

  # 2. Call MCU function that produces a pulse of 7000us
  DUT Invoke    create_pulse    7000

  # 3. Read back the measured pulse
  ${result_us} =  GPIO Get Measurement In Microseconds

  # 4. Check that measurement result is within 1% bounds
  Check That Within 1% Tolerance    7000    ${result_us}
```



Try it from your browser

Do you want to see for yourself how easy driver testing is? Then get your personal test environment with a single click on embeff.com/ep-demo
This environment consists of a browser-based version of Visual Studio Code which is connected to one of our demo devices. It is ready-to-use with examples for driver testing.

embeff GmbH
Willy-Brandt-Allee 31 b · D-23554 Lübeck · Germany
Tel. +49 451 16088690 · Fax +49 451 16088699
info@embeff.com · www.embeff.com

embeff
BETTER EMBEDDED.