µLC Test System | Control unit signal simulator

Hardware in the loop test automation

BOSCH
Technik fürs Leben

Test Automation

Particular advantages of the µLC test system are its compact dimensions and the possibility of using it flexibly and space-saving. It allows the user a very easy initial test setup and offers among other things a multitude of possibilities with test automation. Due to the low space requirement of the µLC Test System each ECU project can be equipped with an individual device.

The **µLC Test System** offers the possibility of both complete automation through Lua scripting and an application programming interface (API) through which the µLC Test System can be controlled by other programs as well.

Consequently, the control software does not only provide a graphical interface for manual operations but there is also the possibility to operate the µLC Test System with other programs.

Top Features

- Compact and easy device offering the possibility for test automation
- Easy accessibility of further programs through .NET- and COM- interface of API
- Usage with typical tools for test automation possible (TPT, ECU-Test, LabVIEW, CANoe etc.)
- Generation of closed-loop models by test automation frameworks on the PC

Continuous Testing und continuous Integration

The µLC Test System can be used for Continuous Testing (CT) and Continuous Integration (CI). In Continuous Testing the µLC Test System is utilized as a tool for immediate interaction. Dependent on the stage of expansion the current status of the software is tested progressively trough Continuous Testing. By means of Continuous Integration tests are getting initiated on the server and the possibility for automation of manual steps is offered reaching from the program state to test execution. The µLC Test System can be used for the following tests in the field of Continuous Testing und Continuous Integration:

- Base Function Test
- Tests for In- and Outputs (I/O test)
- Unittest

For Continuous Testing the µLC Test System can be automatized with other software tools.

Application programming interface (API)

In addition to the graphical interface of the MicroLC Software, the µLC Test System provides an application programming interface (API) that enables other programs to make use of the functions of the MicroLC Software. Through this interface the access to test automation and to programs made up by the applicant is possible. The API offers a .NET-interface as well as a COM-interface allowing a wide range of programs access.

Basically, the µLC Test System supports all common programs and tools for test automation. So far, the µLC Test System is already used with the following programs:

- ECU-Test (Trace Tronic)
- TPT (Piketec)
- PROVEtech (MBtech)
- LabVIEW (NI)
- CANoe (Vector)
- Excel (Microsoft)
- Matlab Simulink (MathWorks)

With test automation frameworks test procedures can be generated and run fully automatized. Malfunctions can be documented with test reports. Since other devices and software (e.g. LabCar PT, CANalyzer, INCA) can be actuated by test automation frameworks as well, it is possible to integrate the µLC Test System into greater test arrangements and to use all devices simultaneously.

Due to the possibility to access the API with own programs closed-loop models can be arranged as well for specific tests which prepare returned values and generate new ones as response. Owing to the COM-interface one is not bound to .Net languages, but can additionally make use of other languages (Java, VBS, VBA, C++, Python, etc.).

Lua Scripting

Another possibility to program processes and to run them automatized is offered by Lua scripting. In addition to the API the device can be operated with Lua scripting and a test automation can be generated. Lua scripting is in this case a more flexible and simple alternative and allows for a faster and less complicated test automation.

In the following you can find an example of a Lua script.

```
-- PWM Example --

-- User Config----------------------------------------------------------
voltage = 5                     -- set output voltage to 5V or 12V
channel = 0                     -- set the channel, values from 0 to 3 are possible
duration = 2                    -- A common need is to pause (sleep) a program for a certain number of seconds
subsignals = 3                  -- number of subsignals
------------------------------------------------------------------------------
SetPWMSignalTypeComplex(channel)
SetPWMSignalOutputLevel(channel, voltage)
SetPWMSubsignalCount(channel,subsignals)

SetPWMSubsignalFrequency(channel, 0, 25)
SetPWMSubsignalDutycycle(channel, 0, 12.5)

SetPWMSubsignalFrequency(channel, 1, 20)
SetPWMSubsignalDutycycle(channel, 1, 10)

SetPWMSubsignalFrequency(channel, 2, 100)
SetPWMSubsignalDutycycle(channel, 2, 50)

SetPWMState(channel,true)      -- Enable PWM channel

sleep(duration)

SetPWMState(channel,false)     -- Disable PWM channel
```