



**acontis technologies GmbH**

**SOFTWARE**

# **EC-Simulator**

## **Rust Programming Interface**

**Version 3.2**

**Edition: November 6, 2025**

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

© Copyright **acontis technologies GmbH**

Neither this document nor excerpts therefrom may be reproduced, transmitted, or conveyed to third parties by any means whatever without the express permission of the publisher. At the time of publication, the functions described in this document and those implemented in the corresponding hardware and/or software were carefully verified; nonetheless, for technical reasons, it cannot be guaranteed that no discrepancies exist. This document will be regularly examined so that corrections can be made in subsequent editions. Note: Although a product may include undocumented features, such features are not considered to be part of the product, and their functionality is therefore not subject to any form of support or guarantee.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Requirements . . . . .	4
1.2	Architecture . . . . .	4
<b>2</b>	<b>Programmers Guide</b>	<b>6</b>
2.1	Sample Application . . . . .	6
2.2	Sample Code . . . . .	6
2.3	Wrapper . . . . .	6
2.3.1	Modules . . . . .	6
2.3.2	Return code vs. exception handling . . . . .	7
2.4	Supported IDEs . . . . .	7
2.4.1	Visual Studio Code . . . . .	7
<b>3</b>	<b>FAQ</b>	<b>10</b>

## 1 Introduction

The Rust Wrapper provides a Rust interface to use EC-Master, EC-Simulator and RAS Client/Server.

### 1.1 Requirements

#### Rust v1.90.0 and above

- Download “rustup” and install Rust

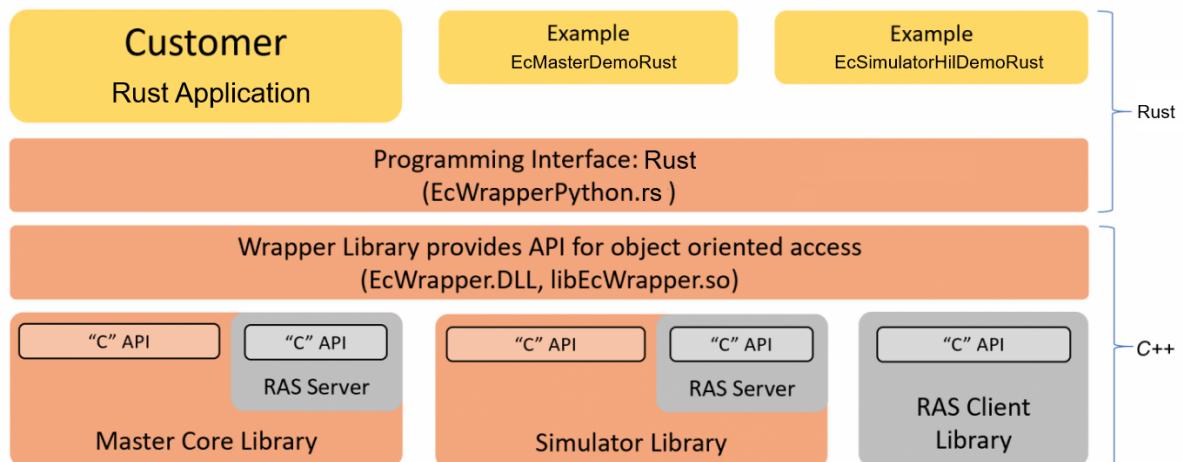
#### Windows (x86/x64)

- Microsoft Windows 10 and above
- Microsoft Visual C++ 2015 Runtime

#### Linux (x86/x64/ARM)

- Ubuntu 18.04 and above

## 1.2 Architecture



The architecture contains 4 basic layers:

#### **Customer Rust application or our examples (EcMasterDemoRust, ...)**

- Demo application, written in Rust

#### **Programming Interface (EcWrapperRust)**

- Provides an object oriented API written in Rust

#### **Wrapper Library (EcWrapper)**

- Native wrapper library, which provides API for object oriented access

## Native Libraries

- Master Core Library
- Simulator Library
- RAS Client Library

## 2 Programmers Guide

### 2.1 Sample Application

There is currently 1 script available:

**EcMasterDemoRust.bat**

Starts the console demo application

### 2.2 Sample Code

The Rust demo application contains of 3 modules:

**EcDemoApp.rs:**

Console demo application

**EcLogging.rs:**

Logging module, which writes asynchronous messages to console, file, ...

**EcUtil.rs:**

Utility module, which contains some helper functions

## 2.3 Wrapper

### 2.3.1 Modules

The Rust Wrapper contains of 5 modules:

**EcWrapperRust.rs**

```
class CEcWrapperRust
    EC-Wrapper base class

class CEcMasterRust
    EC-Master

class CEcMasterMbxGatewayClientRust
    Mailbox Gateway Client for EC-Master

class CEcMasterMbxGatewayServerRust
    Mailbox Gateway Server for EC-Master

class CEcSimulatorRust
    EC-Simulator

class CEcSimulatorRasServerRust
    RAS Server for EC-Simulator

class CEcRasClientRust
    RAS Client for EC-Master / EC-Simulator
```

**EcMotionRust.rs**

```
class CEcMotionRust
    EC-Motion interface
```

**EcWrapperRustTypes.rs**

Rust types

**EcWrapper.rs**

C Rust interface (internal)

**EcWrapperTypes.rs**

C Rust types (internal)

### 2.3.2 Return code vs. exception handling

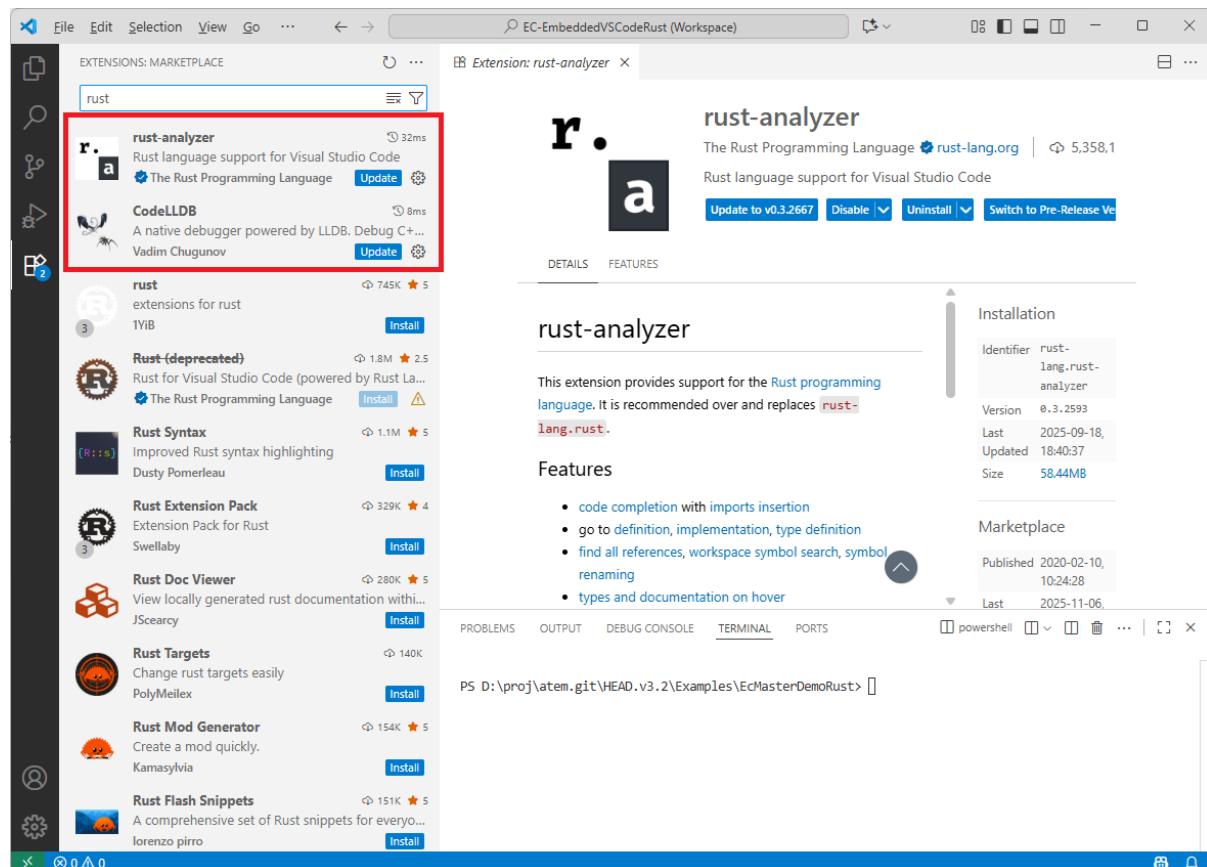
The most of all API functions returns a return code for error handling. This behaviour can be changed to throw an exception in error case by simply setting:

```
CeCWrapperRust_EnableExceptionHandling = true // default is false
```

## 2.4 Supported IDEs

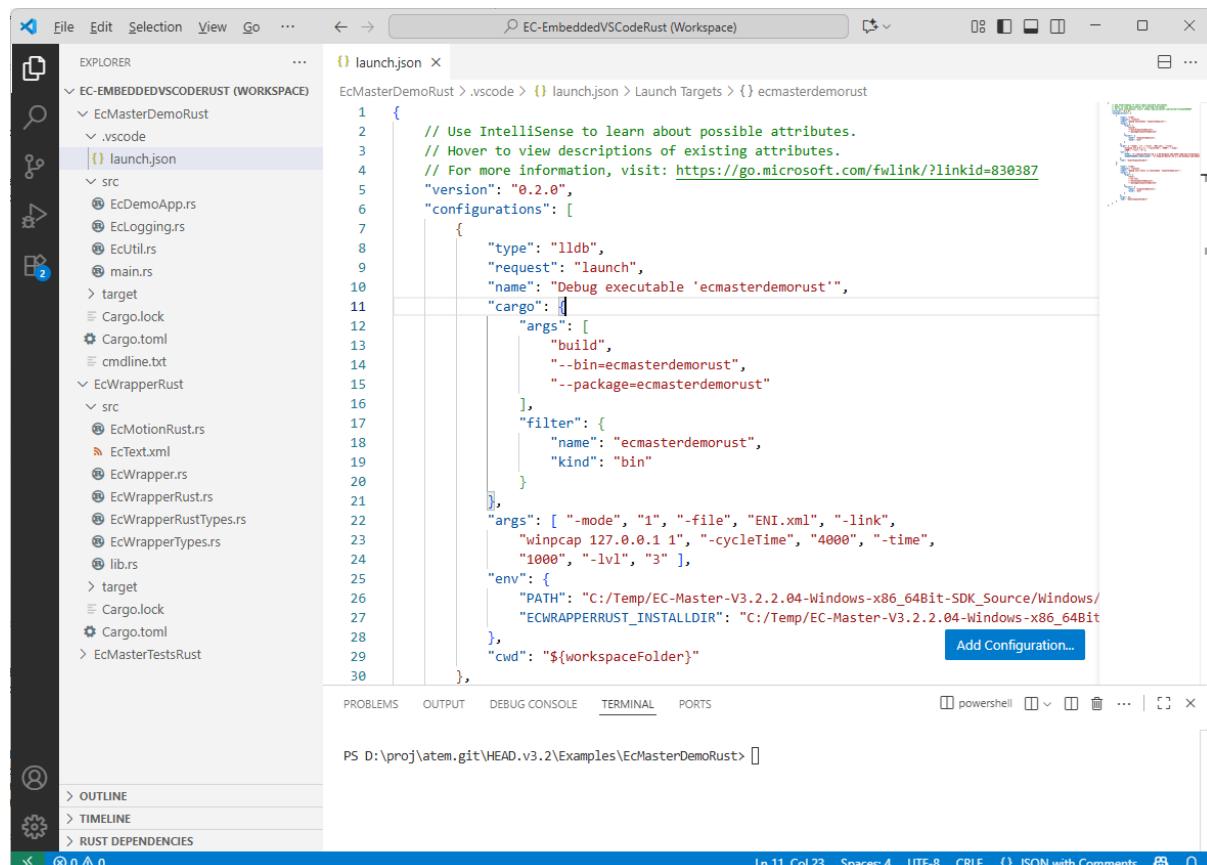
### 2.4.1 Visual Studio Code

Install rust extension by open extension tab and enter *rust*:



Open package folder e.g. **EC-Master-V3.2.2.04-Windows-x86\_64Bit-SDK\_Source** and configure the **launch.json**:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "lldb",
            "request": "launch",
            "name": "Debug executable 'ecmasterdemorust'",
            "cargo": {
                "args": [
                    "build",
                    "--bin=ecmasterdemorust",
                    "--package=ecmasterdemorust"
                ],
                "filter": {
                    "name": "ecmasterdemorust",
                    "kind": "bin"
                }
            },
            "args": [ "-mode", "1", "-file", "ENI.xml", "-link", "winpcap 127.0.0.1 1", "-cycleTime", "4000", "-time", "1000", "-lvl", "3" ],
            "env": {
                "PATH": "C:/Temp/EC-Master-V3.2.2.04-Windows-x86_64Bit-SDK_Source/Windows/x64/",
                "ECWRAPPERRUST_INSTALLDIR": "C:/Temp/EC-Master-V3.2.2.04-Windows-x86_64Bit-SDK_Source/Windows/x64/"
            },
            "cwd": "${workspaceFolder}"
        }
    ]
}
```



Start debugging and the demo output will be written into the terminal:

The screenshot shows the VS Code interface with the following details:

- Editor:** The main area displays the `EcDemoApp.rs` file. The code implements the `EcDemoApp` trait with an `Initialize` method.
- Sidebar:** The left sidebar includes sections for **VARIABLES**, **WATCH**, **CALL STACK**, **BREAKPOINTS** (with a checkbox for "Rust: on panic" and one for "EcWrapperRust.rs" which is checked), **TRACE INSPECTOR**, **MODULES**, and **EXCLUDED CALLERS**.
- Terminal:** The bottom right panel is the terminal, which contains the following demo output:

```
Master: Type='2', Code='1', Msg='Master state changed from <UNKNOWN> to <INIT>'  
Master: Type='2', Code='1', Msg='Master state changed from <INIT> to <PREOP>'  
Master: Type='2', Code='1', Msg='Master state changed from <PREOP> to <SAFEOP>'  
Master: Type='2', Code='1', Msg='Master state changed from <SAFEOP> to <OP>'  
EcMasterDemoRust runtime: is ...  
Master: Type='2', Code='1', Msg='Master state changed from <OP> to <INIT>'
```
- Bottom Status Bar:** Shows the current line (Ln 1135, Col 1), spaces (Spaces: 4), encoding (UTF-8), and file type (Rust). It also includes icons for powershell, Codell..., and Debug exec... (which is highlighted).

### 3 FAQ

I installed Rust and the demo crashes with strange errors. What can I do?

This might be a problem of mixing x86 with x64 binaries. Verify that if you have installed the Rust runtime for x64 bit, please install also EC-Master for x64 bit.