

# MicroEJ Platform Reference Implementation

## Developer's Guide



for ESP-WROVER-KIT V4.1

MicroEJ Corp.

Version 1.5.1

May 22, 2020

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Intended Audience . . . . .	1
1.2	Scope . . . . .	1
1.3	Prerequisites . . . . .	1
<b>2</b>	<b>Create and Use Your First MicroEJ Platform</b>	<b>2</b>
2.1	Create a MicroEJ platform . . . . .	2
2.2	Run an Example on the MicroEJ simulator . . . . .	3
2.2.1	Create Example . . . . .	4
2.2.2	Run Example . . . . .	5
2.3	Run the Example on the ESP-WROVER-KIT V4.1 Board . . . . .	6
2.3.1	Compile MicroEJ standalone application . . . . .	6
2.3.2	Link and Deploy the MicroEJ standalone application . . . . .	7
<b>3</b>	<b>Specification</b>	<b>13</b>
3.1	Overview . . . . .	13
3.2	MicroEJ Platform Configuration . . . . .	13
3.3	MicroEJ Platform Output Stream . . . . .	13
3.4	Multiple Application . . . . .	13
3.5	Graphical User Interface . . . . .	14
3.5.1	Display . . . . .	14
3.5.2	Inputs . . . . .	15
3.6	Network . . . . .	15
3.7	SSL . . . . .	15
3.8	File System . . . . .	15
3.9	UART Connector . . . . .	16
3.10	HAL . . . . .	16
3.11	Espressif esp-idf . . . . .	16
<b>4</b>	<b>Board Configuration</b>	<b>17</b>
4.1	Mandatory Connectors . . . . .	17
<b>5</b>	<b>Changelog</b>	<b>19</b>
5.1	[1.5.1] - 2020-05-19 . . . . .	19
5.1.1	Fixed . . . . .	19
5.2	[1.5.0] - 2020-05-13 . . . . .	19
5.2.1	Added . . . . .	19
5.2.2	Changed . . . . .	19
5.2.3	Fixed . . . . .	20
5.3	[1.4.5] - 2019-11-25 . . . . .	20
5.3.1	Fixed . . . . .	20
5.4	[1.4.4] - 2019-11-13 . . . . .	20
5.4.1	Fixed . . . . .	20

5.5	[1.4.2] - 2019-09-12	20
5.5.1	Fixed	20
5.6	[1.4.1] - 2019-09-05	21
5.6.1	Added	21
5.6.2	Fixed	21
5.7	[1.4.0] - 2019-07-26	21
5.7.1	Added	21
5.7.2	Fixed	21
5.8	[1.3.0] - 2018-10-08	21
5.8.1	Added	21
5.8.2	Fixed	22
5.9	[1.2.0] - 2018-08-02	22
5.9.1	Added	22
5.9.2	Fixed	22
5.10	[1.1.0] - 2018-05-30	22
5.10.1	Added	22
5.10.2	Fixed	22
5.11	[1.0.0] - 2018-05-04	22
<b>6</b>	<b>Readme: MicroEJ ESP32-WROVER Platform</b>	<b>23</b>
6.1	Configurations	23
6.1.1	Single application platform	23
6.1.2	Multi application platform	24
6.2	Getting Started	24
6.3	Platform Output Stream	24
6.4	Other files to read	24
<b>7</b>	<b>Release notes: MicroEJ ESP32-WROVER Platform</b>	<b>25</b>
7.1	Description	25
7.2	Versions	25
7.2.1	Platform	25
7.2.2	Dependencies	25
7.3	Environment	26
7.4	Known issues/limitations	26
<b>Index</b>		<b>28</b>

# Chapter 1

## Introduction

### 1.1 Intended Audience

The intended audience for this document are developers who wish to develop their first MicroEJ platform with MicroEJ SDK and deploy a MicroEJ standalone application onto. Notes:

- This document is for the Espressif ESP-WROVER-KIT V4.1 board.
- This document is not a user guide for the C development environment used for the final application link. Please consult the supplier of the C development environment for more information.
- Please visit the website [WEBSITE](#) for more information about ESP-WROVER-KIT V4.1 products (platforms, videos, examples, application notes, etc.).

### 1.2 Scope

This document describes, step by step, how to start your development with MicroEJ SDK

- Create a MicroEJ platform for ESP-WROVER-KIT V4.1 board.
- Run a MicroEJ standalone application on the MicroEJ simulator.
- Run a MicroEJ standalone application on the MicroEJ platform and deploy it on the ESP-WROVER-KIT V4.1 board.

### 1.3 Prerequisites

- PC with Windows 7 or later.
- The MicroEJ SDK environment must be installed.
- ESP-WROVER-KIT V4.1 board.

## Chapter 2

# Create and Use Your First MicroEJ Platform

### 2.1 Create a MicroEJ platform

The aim of this chapter is to create a MicroEJ platform from a MicroEJ architecture. The platform will then be used to run a MicroEJ standalone application in subsequent chapters.

Although it is possible to use MicroEJ SDK to create every aspect of a MicroEJ platform in accordance with specific requirements, in this chapter we will use a pre-packaged example of a MicroEJ platform that is already configured for the ESP-WROVER-KIT V4.1.

Inside MicroEJ SDK, the selected example is imported as several projects prefixed by the given name:

- {PLATFORM}-configuration: Contains the platform reference implementation configuration description. Some modules are described in a specific sub-folder / with some optional configuration files ( `.properties` and / or `.xml` ).
- {PLATFORM}-bsp: Contains a ready-to-use BSP software project for the ESP-WROVER-KIT V4.1 board, including a MicroEJ SDK project, an implementation of MicroEJ core engine (and extensions) port on FreeRTOS and the ESP-WROVER-KIT V4.1 board support package.
- {PLATFORM}-fp: Contains the board description and images for the MicroEJ simulator. This project is updated once the platform is built.

Inside the {PLATFORM}-configuration project, open the `.platform` MicroEJ platform configuration file.

From this MicroEJ platform configuration file, click on the link [Build Platform](#)

## Overview



### Platform Properties

General information about this platform.

Device:	<input type="text" value="Board"/>
Name:	<input type="text" value="MyPlatform"/>
Version:	<input type="text" value="2.1.1"/>
Provider :	<input type="text" value="MicroEJ"/>
Vendor URL:	<input type="text" value="http://developer.microej.com/4.0/sdk/license"/>

### Platform Content

The content of the platform is composed of two parts:

-  [Environment](#): select the architecture.
-  [Modules](#): select modules to import in the platform.

### Platform Configuration

Once the content of the platform is chosen, it can be configured.

 [Configuration](#)

Each module can be configured creating a folder with its name along the .platform file. It could contain:

- an optional [module].properties file,
- optional module specific files and folders.

Modifying one these files requires to build the platform again.

### Build

Generate and test the platform.


 [Build Platform](#): The new platform is now available and visible in [Available Platforms](#)

Fig. 1: MicroEJ Platform Build

The build starts. This step may take several minutes. You can see the progress of the build steps in the MicroEJ console. Please wait for the final message:

```
BUILD SUCCESSFUL
```

At the end of the execution the MicroEJ platform is fully built for the ESP-WROVER-KIT V4.1 board and is ready to be linked into the MicroEJ SDK project.

The MicroEJ platform is now ready for use and available in the MicroEJ platforms list of your MicroEJ repository ([Windows > Preferences > MicroEJ > Platforms in workspace](#)).

## 2.2 Run an Example on the MicroEJ simulator

The aim of this chapter is to create a MicroEJ standalone application from a built-in example.

Initially, this example will run on the MicroEJ simulator. Then, in the next section, this application will be compiled and deployed on the ESP-WROVER-KIT V4.1 board using the MicroEJ platform.

## 2.2.1 Create Example

- Open MicroEJ SDK.
- Open the **File > New > MicroEJ Standalone Application Project** menu.
- Enter your project name (e.g. HelloWorld).

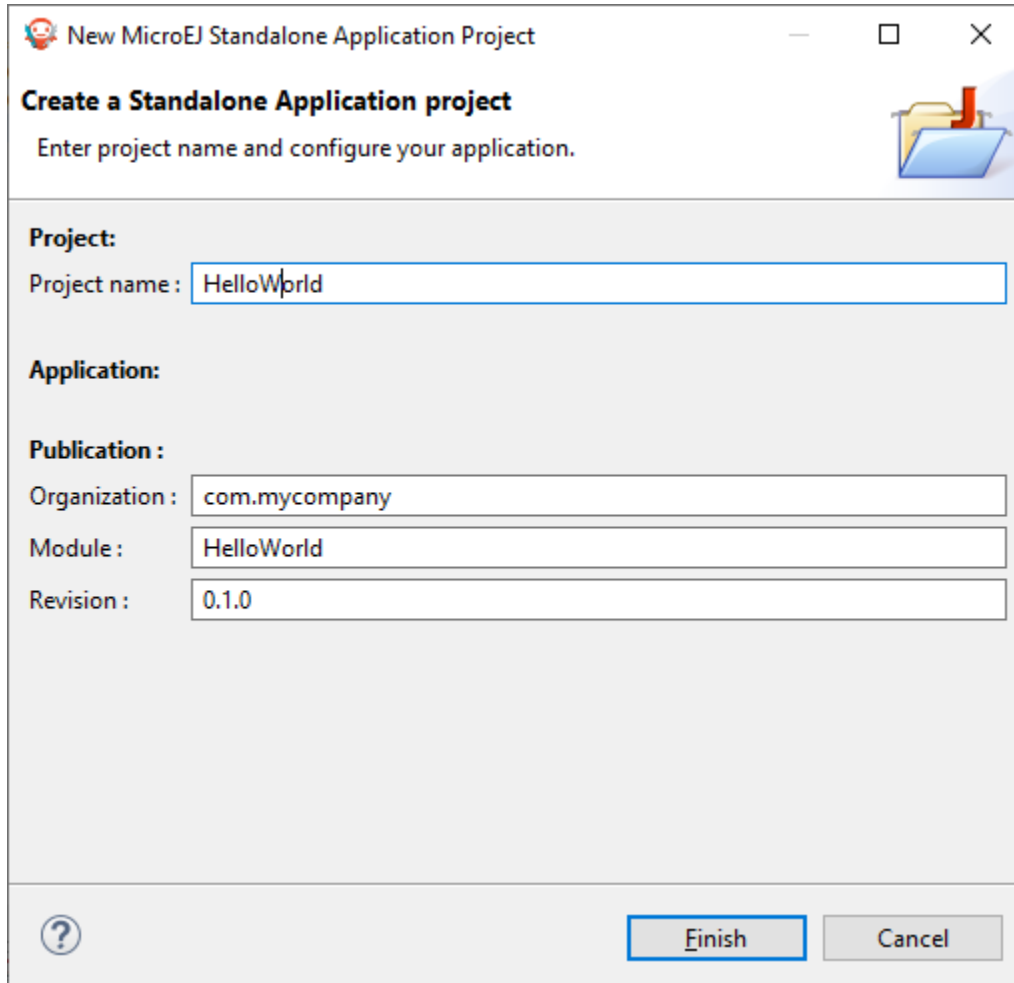


Fig. 2: MicroEJ standalone application Creation

- Click on Finish.

The example is created into a project with the given name. The main class (the class which contains the `main()` method) is automatically generated.

## 2.2.2 Run Example

- Select the project in the Package Explorer tree
- Right-click on this project and select **Run As > MicroEJ Application**

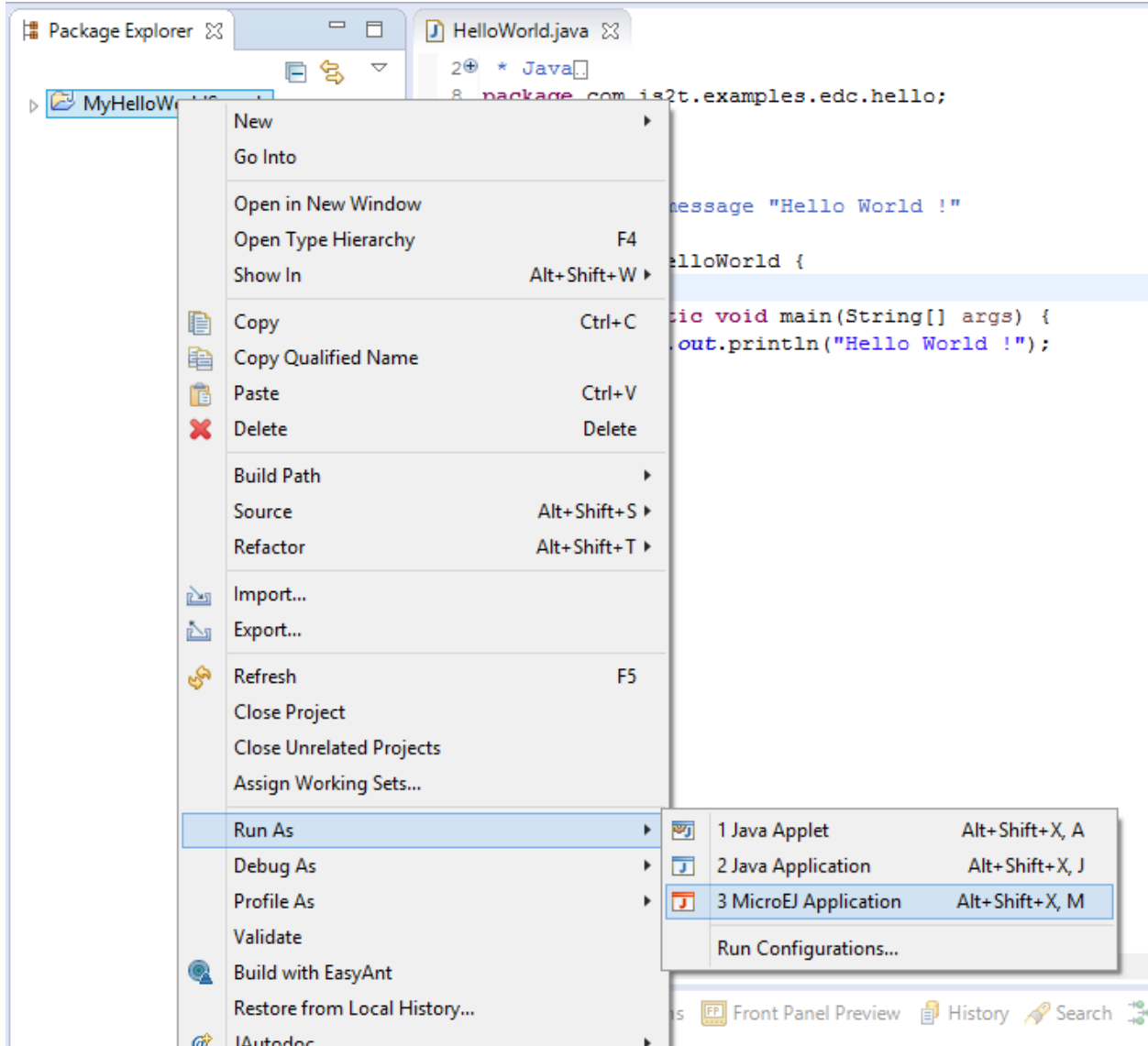


Fig. 3: MicroEJ standalone application Running

The application starts. It is executed on the MicroEJ simulator of the selected MicroEJ platform. The result of the test is printed in the console:



```
Hello World!
```

## 2.3 Run the Example on the ESP-WROVER-KIT V4.1 Board

### 2.3.1 Compile MicroEJ standalone application

- Open the run dialog (`Run > Run configurations...`).
- Select the MicroEJ Application launcher `EXAMPLENAME Main` that is created by the previous chapter.
- Open `Execution` tab.
- Select `Execute on Device`.

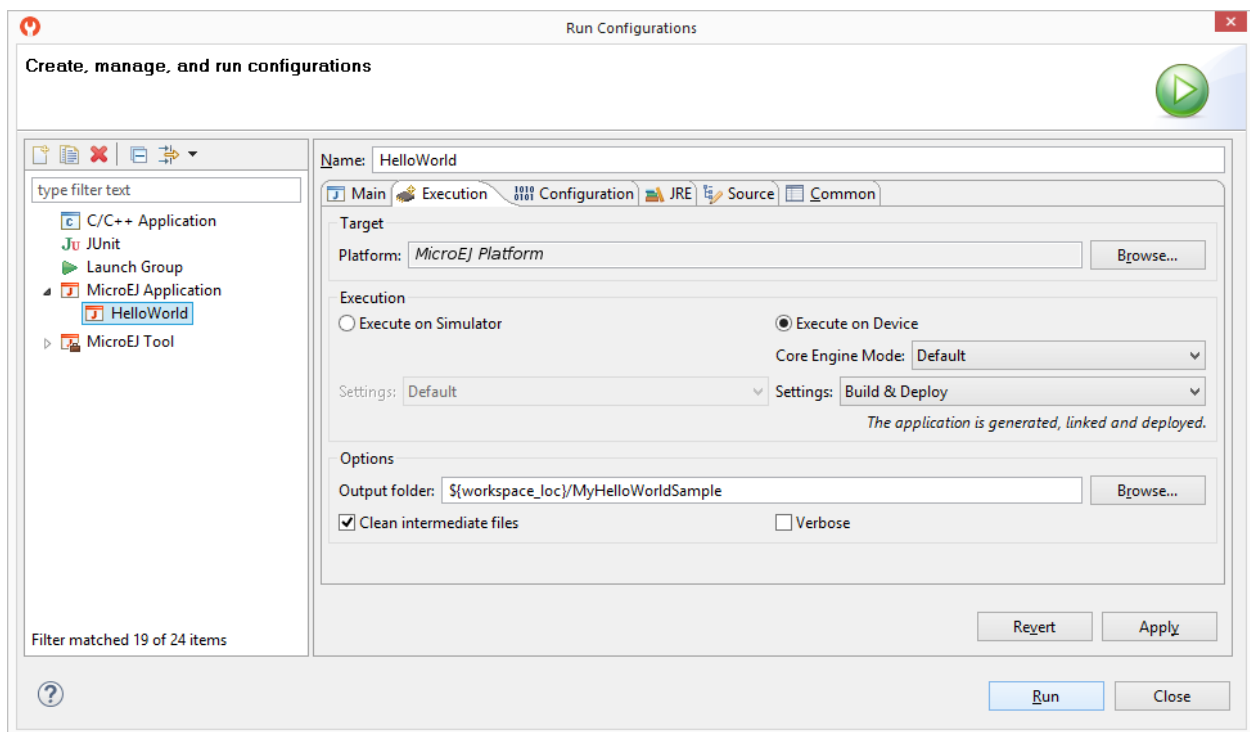


Fig. 4: Execution on Device

- Click `Run`: the application is compiled, and the compilation result (an ELF file) is copied into a well-known location in the example project. The Esptool tool has to be used to load the program on the board.

### 2.3.2 Link and Deploy the MicroEJ standalone application

The aim of the final step is to:

- Compile the BSP project (such as drivers).
- Link the BSP and the others libraries (MicroEJ Core Engine, C stacks, MicroEJ standalone application etc.).
- Deploy a MicroEJ standalone application on the ESP-WROVER-KIT V4.1 board.

---

**Note:** This final step uses MicroEJ SDK.

---

#### Select the BSP project

- In MicroEJ SDK, expand the project ESP32WROVER-MyPlatform-GNUv52\_xtensa-esp32-psram and the folder [Projects/microej](#).
- A MicroEJ SDK project file ( [.cproject](#) ) is available:

- > ESP32-WROVER-Xtensa-FreeRTOS-bsp
    - > Drivers
    - > Projects
      - > coremark
      - > microej
        - > .settings
        - > ble
        - > core
        - > ecom-comm
        - > ecom-network
        - > ecom-wifi
        - > espressif
        - > fs
        - > hal
        - > kf
        - > main
        - > microej-util
        - > net
        - > ssl
        - > trace
        - > ui
        - > util
        - .cproject
        - .project
        - cco\_esp\_idf.properties
        - cco\_net-bsd.properties
        - cco\_net-lwip.properties
        - cco\_osal-FreeRTOS.properties
        - cco\_osal-headers.properties
        - cco\_trace-systemview.properties
        - Makefile
        - partitions\_microej.csv
        - README.md
        - sdkconfig
      - > unit\_tests
        - .gitignore
    - > Tools
      - .gitignore
      - .project
      - [ATTACH] Burn eFuse.launch
      - [ATTACH] eFuse summary.launch
      - [ATTACH] Erase flash.launch
      - [ATTACH] ESP32 Application debugging.launch
      - [ATTACH] Flash coremark binary.launch
      - [ATTACH] Flash microej binary.launch
      - [ATTACH] Flash unit\_tests binary.launch
      - [ATTACH] Start GDB Server.launch
  - > ESP32-WROVER-Xtensa-FreeRTOS-configuration
  - > ESP32-WROVER-Xtensa-FreeRTOS-fp

Fig. 5: MicroEJ SDK Project Selection

- By default, `.cproject` files are hidden. To list them in the package explorer:
  1. click on: "view menu" button > "Filters..."
  2. uncheck `.* resources`
  3. click on: "ok"

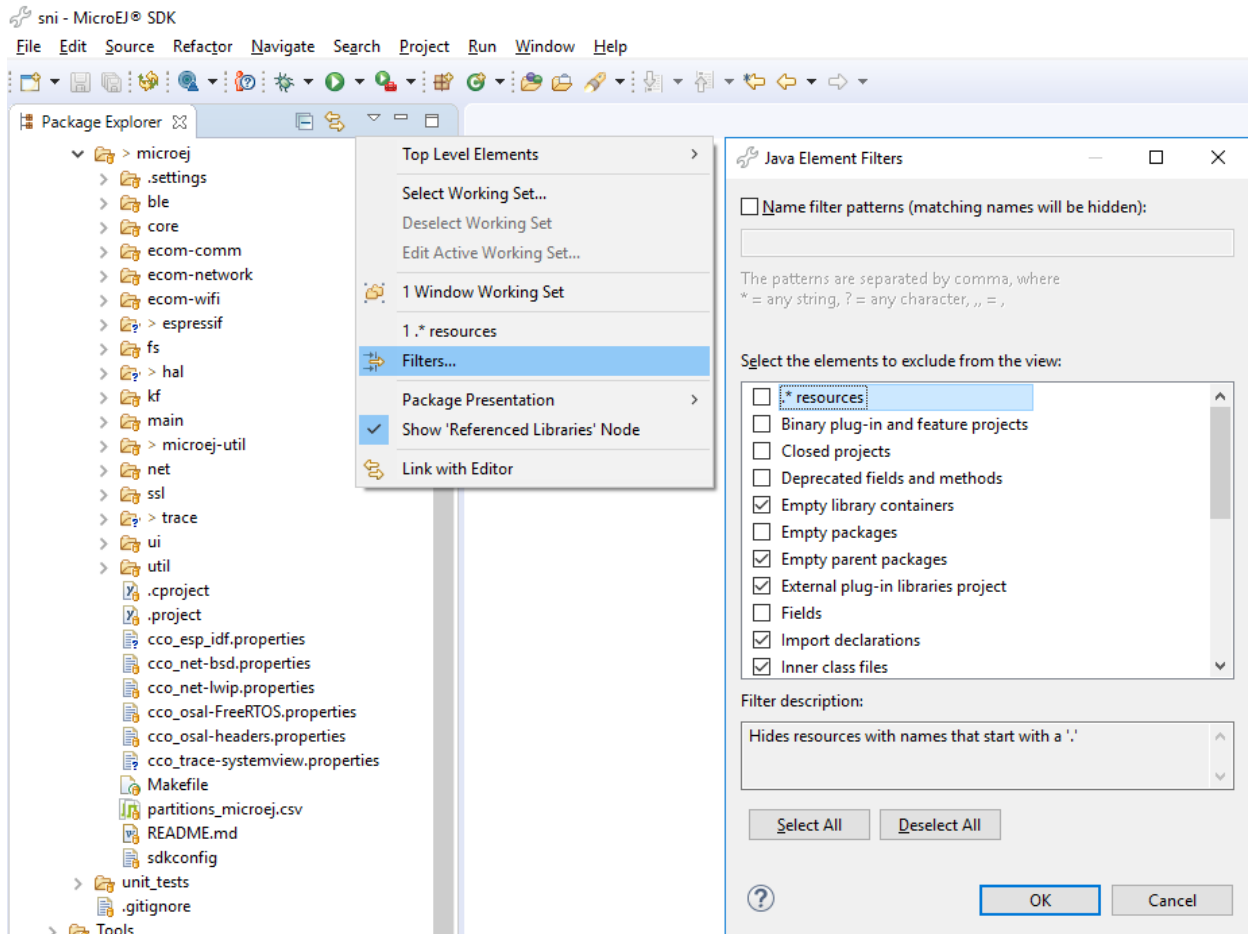


Fig. 6: MicroEJ SDK Filter out resources in Package manager

Espressif tools installation procedure:

- Espressif ESP32 toolchain:
  - Download [https://dl.espressif.com/dl/esp32\\_win32\\_msys2\\_environment\\_and\\_toolchain-20181001.zip](https://dl.espressif.com/dl/esp32_win32_msys2_environment_and_toolchain-20181001.zip)
  - Unzip its content to `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Tools`
  - Edit environment variable `PATH` to add the directory: `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Tools/msys32/opt/xtensa-esp32-elf/bin`
  - Close your MicroEJ SDK and reopen it
- OpenOCD (Open On-Chip Debugger):
  - Download <https://github.com/espressif/openocd-esp32/releases/download/v0.10.0-esp32-20190313/openocd-esp32-win32-0.10.0-esp32-20190313.zip>

- Unzip its content to `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Tools`
- Zadig (generic USB driver installer):
  - Download <https://github.com/pbatard/libwdi/releases/download/v1.3.0/zadig-2.3.exe>
  - Copy it to `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Tools/zadig`
- Refresh the `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Tools` folder in Eclipse. Then, ESP32 tools is now installed.

---

**Note:** For more details, please refer to the following documentation: <https://docs.espressif.com/projects/esp-idf/en/v3.3.1/get-started/index.html#setup-toolchain>.

---

Import this project in your MicroEJ SDK workspace:

1. Go to `File > Import... > General > Existing Projects into Workspace`
2. click on `Next` button.
3. Click on the `Select root directory:` radio button and then on `Browse` button.
4. Select the path associated to `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Project/microej` and click on `OK` button.
5. Select the `microej` project in the Projects list.
6. Verify that the `Copy projects into workspace` checkbox is not checked
7. Click on `Finish` button.

The BSP project is now imported in your current workspace. It can be built by selecting it in the workspace projects list and click on `Project > Build Project`.

## Build and Link the Platform

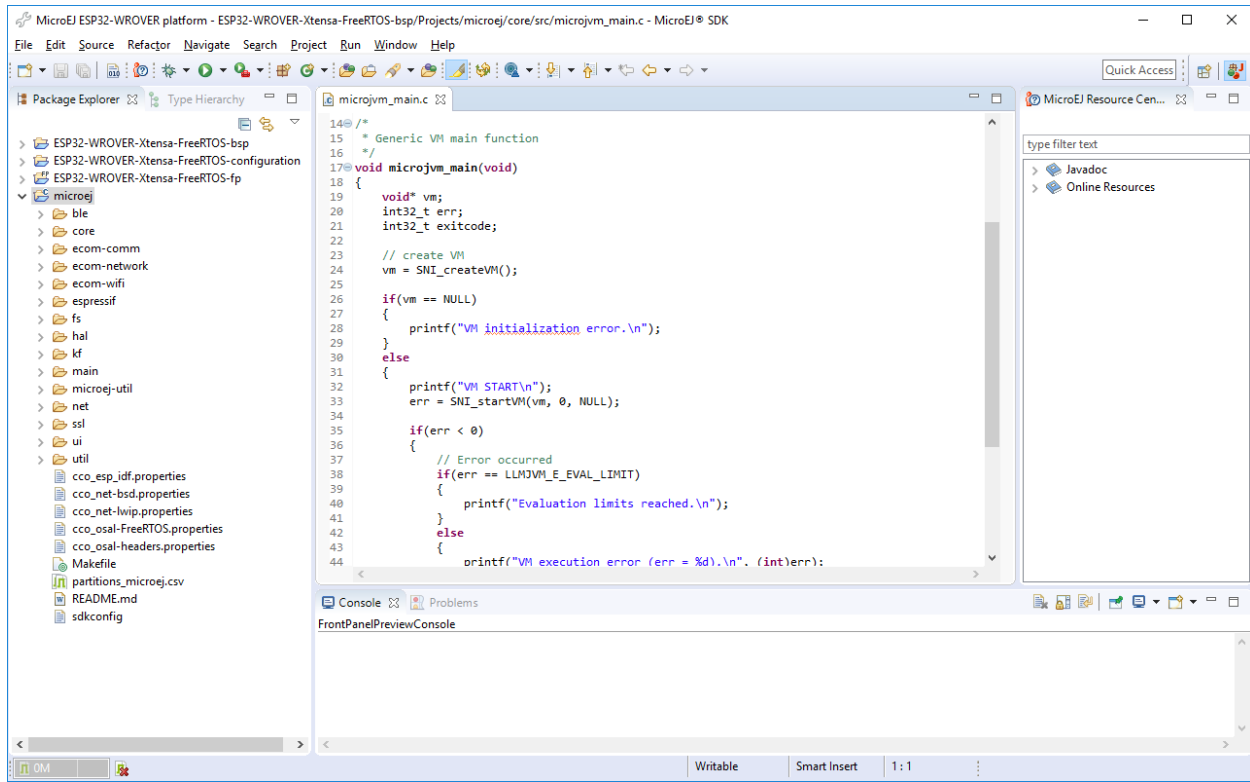


Fig. 7: MicroEJ SDK IDE

Build the MicroEJ SDK project by: clicking on the menu **Project > Build target**.

The project is compiled and linked.

See *Mandatory Connectors* to use the right connectors.

## Deploy the Application

**Note:** The target board can be flashed using Espressif bootloader.

Please follow the Espressif documentation but replace **make** with:

- `./build_singleapp` for single application (standalone) firmware
- `./build_multiapp` for multi application firmware

Espressif documentation for firmware flashing is located at: (<https://docs.espressif.com/projects/esp-idf/en/v3.3.1/get-started/index.html#build-and-flash>)

**Congratulation: you run the Application**

The application starts. The result of the execution is output on printf COM port. (See *Mandatory Connectors* to use the right connectors).

Congratulations, you have deployed a MicroEJ standalone application on a MicroEJ platform.

# Chapter 3

## Specification

### 3.1 Overview

MicroEJ platform on ESP-WROVER-KIT V4.1 includes FreeRTOS, a graphical user interface, a TCP/IP network connection, a file system on SD-Card and some custom GPIOs.

### 3.2 MicroEJ Platform Configuration

MicroEJ platform is based on MicroEJ architecture for ESP32.

Documentation regarding the hardware can be found on <https://docs.espressif.com/projects/esp-idf>

- [Board functional description](#)
- [ESP32-WROVER module description](#)

MicroEJ platform configuration is detailed in *Release notes: MicroEJ ESP32-WROVER Platform*

### 3.3 MicroEJ Platform Output Stream

MicroEJ platform Output Stream is described in: *Platform Output Stream*

### 3.4 Multiple Application

MicroEJ provides two different platforms:

- The single application platform
- The multi application platform

To identify if the installed platform supports single or multi application:

- Click on [Windows > Preferences](#)
- In the left list of the *Preferences* window, click on [MicroEJ > Platforms](#)
- Platforms are identify by:



- `MultiApp` for multi application platforms
- `SingleApp` for single application platforms

If the MicroEJ platform supports the multi application mode, a multi application firmware can be built. This firmware can manage MicroEJ sandboxed applications. multi application mode requires a specific memory area to load the MicroEJ sandboxed applications. This memory area is located in external SDRAM and its default size is 1 MB.

## 3.5 Graphical User Interface

MicroEJ platform features a graphical user interface. It includes a display, two user LEDs and a runtime PNG decoder.

### 3.5.1 Display

The display module drives a 320 x 240 LCD display. The pixel format is 16 bits-per-pixel. The display device is clocked at 60Hz and it is connected to the MCU via a SPI link, clocked at 80MHz for ST7789V LCD display and at 33MHz for ILI9341V LCD display.

---

**Note:** The display stack implementation uses the direct single-buffering mode, as the LCD is connected via SPI and cannot access any memory mapped in the MCU address space. Because of the HW implementation, updating the display buffer will most likely result in “noisy” rendering and flickering, as the LCD displays the current frame from its own memory, while the MCU sends the content of the new frame via SPI.

The back buffer is located in external PSRAM. The size depends on the display size in pixels and on the number of bits-per-pixel (BPP):

`bufferSize = width * height * bpp / 8;` , where:

- `width` is the display width in pixels: 320
- `height` is the display width in pixels: 240
- `bpp` is the number of bits-per-pixel: 16

The buffers size is 0 .

---

MicroUI requires a RAM buffer to store the dynamic images data. A dynamic is an image decoded at runtime (PNG image) or an image created by the MicroEJ application thanks the API `Image.create(width, height)` . This buffer is located in external RAM.

---

**Note:** This buffer is called “images heap”. An image buffer size follows the same rule than the LCD buffer (see before).

---

### 3.5.2 Inputs

User leds: The board provides an RGB matrix with 3 colored LEDs (red, green, blue), but for the user the green LED is not available, as it uses a GPIO multiplexed with a MicroSD pin. So, from the user perspective:

- LED 0 : Blue LED of the RGB matrix
- LED 1 Red LED of the RGB matrix

## 3.6 Network

MicroEJ platform features a network interface with Wi-Fi as an underlying hardware media. A limited number of 16 sockets could be used for TCP connections, 16 for TCP listening (server) connections and 16 for UDP connections. A DHCP client could be activated to retrieve IP address. All DNS requests could be handled by a MicroEJ software implementation or a native one.

---

**Note:** MicroEJ platform uses LwIP v2.0.0 contained in the Espressif esp-idf SDK. This implementation uses the Espressif SDK heap dynamic memory allocator for all its memory allocation. The TCP MSS is 1436 bytes.

The network portage use a BSD (Berkley Software Distribution) API with select feature. A mechanism named async netconn, with a dedicated task, is used to request non blocking operations and wait for completion or timeout.

The DHCP client is handled by LwIP and the DNS features use a MicroEJ software implementation.

---

## 3.7 SSL

MicroEJ platform features a network secure interface. Available secured protocols are SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2. Supported keys and certificates formats are PKCS#5 and PKCS#12, PEM or DER encoded.

---

**Note:** MicroEJ platform uses mbedTLS v2.6.0. mbedTLS uses a dynamic heap linked to Espressif implementation to store certificates.

---

## 3.8 File System

MicroEJ platform features a file system interface. A SD card is used for the storage (previously formatted to a FAT32 file system). Up to 2 files could be opened simultaneously.

---

**Note:** MicroEJ platform uses FatFS R0.13a.

---

### 3.9 UART Connector

MicroEJ platform does not provide any serial connection.

### 3.10 HAL

MicroEJ platform includes a stubbed implementation of HAL library low level API at present.

### 3.11 Espressif esp-idf

MicroEJ platform includes a Java foundation library that directly mapped few C functions of the Espressif [esp-idf](#) board support package (BSP).

## Chapter 4

# Board Configuration

ESP-WROVER-KIT V4.1 provides several connectors, each connector is used by the MicroEJ Core Engine itself or by a foundation library.

### 4.1 Mandatory Connectors

ESP-WROVER-KIT V4.1 provides a multi function USB port used as:

- Power supply connector
- Probe connector
- Virtual COM port

Ensure the *Power Supply jumper JP7* is fit to the second option: *USB 5V*. Check that the J3 connector is placed on ON. Then just plug a mini-USB cable to a computer to power on the board, be able to program an application on it and to see Espressif bootloader traces.

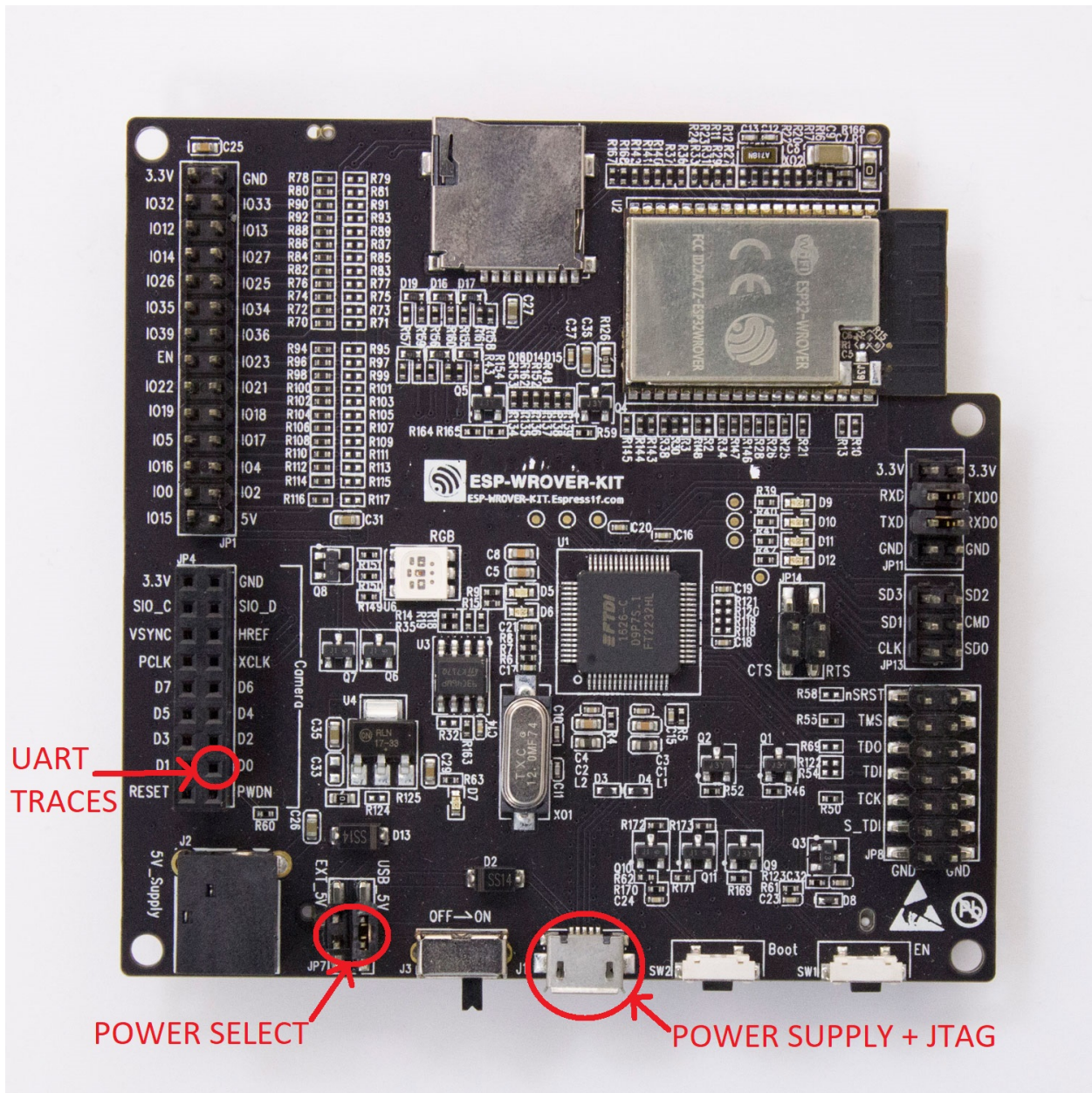


Fig. 1: Mandatory Connectors

# Chapter 5

## Changelog

### 5.1 [1.5.1] - 2020-05-19

#### 5.1.1 Fixed

- Net : update net pack to version 9.2.1.

### 5.2 [1.5.0] - 2020-05-13

#### 5.2.1 Added

- UI: Add support for LCD mode in portrait (via compile switch).
- Core: Print reset reason at startup.
- Device : Add device pack version 1.1.0.

#### 5.2.2 Changed

- Update esp-idf version to 3.3.1.
- Optimized display driver by moving the functions in IRAM.
- Optimized the JVM RTOS task scheduling.
- Architecture : update architecture to version 7.12.1.
- Net : update net pack to version 9.2.0 and addons pack to version 2.1.6.
- Bluetooth: update bluetooth pack to version 2.0.0.
- Bluetooth: update LLBLUETOOTH implementation.

### 5.2.3 Fixed

- Issue with some file names case on Linux
- Wifi Driver : allow to enable softAP in OPEN auth mode.
- Wifi Driver : Fix SSID length too long (null terminated string).
- Ecom-network: Fix isEnabled and isStarted natives.
- Filesystem: Fix get free space and get total space computation.
- Filesystem: Fix the verification of writing data to a file.
- Net: Fix memory leak by freeing the netconn when closing a socket.
- SSL: Reset the ssl session after closing the ssl connection.
- Documentation: Link to Getting-Started page.
- Documentation: Board image.
- Documentation: Instructions to get the traces.

## 5.3 [1.4.5] - 2019-11-25

### 5.3.1 Fixed

- Documentation is now compliant with MicroEJ distribution 19.05.
- Getting Started URL in the README.rst.
- SDK URL in the RELEASE\_NOTES.rst.

## 5.4 [1.4.4] - 2019-11-13

### 5.4.1 Fixed

- An issue that prevents building the platform from sources in MicroEJ SDK.

## 5.5 [1.4.2] - 2019-09-12

### 5.5.1 Fixed

- An issue that prevents linking application on linux hosts.

## 5.6 [1.4.1] - 2019-09-05

### 5.6.1 Added

- Bluetooth stub mock-up

### 5.6.2 Fixed

- An issue that sometime prevents loading application in the simulator.
- Improve the WiFi mock-up user interface.

## 5.7 [1.4.0] - 2019-07-26

This version differentiate configuration (sdkconfig) between singleapp (HDAHT) and multiapp (9C5H4) platforms.

### 5.7.1 Added

- Add Bluetooth support.
- Add new Mock Wi-Fi.
- Add SEGGER SytemView support for singleapp only.
- Disable OTA for multiapp only.
- Fix URLs to espressif.doc.
- Update OpenOCD version to win32-0.10.0-esp32-20190313,
- Improve documentation clarity.

### 5.7.2 Fixed

- LwIP socket leak with esp-idf ESP\_THREAD\_SAFE option enabled

## 5.8 [1.3.0] - 2018-10-08

### 5.8.1 Added

- Java `System.out.println` trace output on a new UART (USB COM port no longer used).
- Add Java ESP32 `esp-idf` foundation library.
- Enable SNI 1.3 non immortal access feature



### 5.8.2 Fixed

- LwIP issue that leads to lockup
- Net multi-thread access issues
- UI low level port do not support all LCD modules that can be included in ESP32-WROVER-KIT V3
- C stack overflow during complex TLS handshake

## 5.9 [1.2.0] - 2018-08-02

### 5.9.1 Added

- Wi-Fi throughput enhancement.
- Add UI MicroEJ pack and provides a device port.
- Add FS MicroEJ pack and provides a device port on SD card.

### 5.9.2 Fixed

- Failure when trying to launch a Wi-Fi scan after mount and dismount.

## 5.10 [1.1.0] - 2018-05-30

### 5.10.1 Added

- Add HAL MicroEJ pack and provides a stubbed implementation.
- Add JPF MicroEJ platform.
- Update esptool MicroEJTool error messages.

### 5.10.2 Fixed

- Failures when trying to attach a GDB debug session.

## 5.11 [1.0.0] - 2018-05-04

Initial release of the platform.

## Chapter 6

# Readme: MicroEJ ESP32-WROVER Platform

This project is used to build MicroEJ reference implementation platform for ESP32-WROVER-KIT development board.

### 6.1 Configurations

The ESP32 WROVER platform reference implementation v1.5.1 is declined into:

- a single application platform (UID: HDAHT),
- a multi application platform (UID: 9C5H4).

Those declinations are themselves declined into:

- an evaluation platform (eval):
  - requires a SDK evaluation license from <https://license.microej.com/>.
  - When limitation is reached, the MicroEJ runtime will terminate. Exit code is described in the MicroEJ Device Developer's Guide provided with your SDK.
- a development platform (dev):
  - requires a SDK USB dongle license.
  - Development platforms do not implement limitation. If used properly, the platform can be run indefinitely or within the limits of the development board.

#### 6.1.1 Single application platform

The single application platform is designed to give optimum performances and resources for single application firmwares. It embeds a specific MicroEJ virtual machine for this purpose.

It also supports SEGGER SystemView and OTA (Over The Air) update management.

### 6.1.2 Multi application platform

The multi application platform is designed for multiple application firmwares. It embeds a specific MicroEJ virtual machine for this purpose.

SEGGER SystemView and OTA have been disabled in order to reduce the memory footprint.

## 6.2 Getting Started

The ESP32-WROVER [SDK Getting Started](#) explains how to start the SDK using the MicroEJ reference implementation platform with the ESP32-WROVER-KIT.

## 6.3 Platform Output Stream

MicroEJ platform uses the virtual UART from the USB port.

The COM port uses the following parameters:

- Baudrate: 115200
- Data bits bits: 8
- Parity bits: None
- Stop bits: 1
- Flow control: None

## 6.4 Other files to read

This directory also contains

- [CHANGELOG](#) to track the changes in the MicroEJ ESP32-WROVER-KIT reference implementation
- [RELEASE NOTES](#) to list:
  - the supported hardware,
  - the known issues and the limitations,
  - the development environment,
  - the list of the dependencies and their versions.

## Chapter 7

# Release notes: MicroEJ ESP32-WROVER Platform

### 7.1 Description

This is the release note of the ESP32 WROVER platform reference implementation v1.5.1. This reference platform is designed for the ESP32-WROVER-Kit.

### 7.2 Versions

#### 7.2.1 Platform

ESP32 WROVER platform reference implementation v1.5.1.

#### 7.2.2 Dependencies

The ESP32 WROVER platform reference implementation v1.5.1 contains the following dependencies:

- MicroEJ ESP32 specific packs:
  - simikou2 (Architecture): 7.12.1
  - simikou2UI (User Interface): 12.1.2
  - simikou2Standalone: 6.0.2
- MicroEJ generic packs:
  - net: 9.2.1
  - net-addons: 2.1.6
  - hal (Hardware Abstraction Layer): 2.0.1
  - hal-stub (stubs for Hardware Abstraction Layer): 1.1.1
  - bluetooth: 2.0.0
  - fs (File System): 4.0.2

- microej-util: 1.3.0
- trace-systemview: 2.0.1
- wadapps (Multi Application Framework): 1.0.1
- wadapps.impl (extra implementation for Multi Application Framework): 1.0.2
- device: 1.1.0
- BSP specific packs:
  - esp\_idf (ESP\_IDF Wrapper): 1.0.0
  - esp\_idf API (ESP\_IDF Wrapper API): 1.0.1
  - esp\_idf-impl (ESP\_IDF extra implementation): 1.0.0
  - esp\_idf-mock (ESP\_IDF Mock-Up): 1.0.0
  - esp-idf: 3.3.1
- Third party tools
  - esptool (Flashing tool): 1.2.0

## 7.3 Environment

The ESP32 WROVER platform reference implementation v1.5.1 requires the following environment:

For development:

- MicroEJ SDK (<http://developer.microej.com/packages/SDK/19.05/>),
- Espressif ESP32 toolchain ([https://dl.espressif.com/dl/esp32\\_win32\\_msys2\\_environment\\_and\\_toolchain-20181001.zip](https://dl.espressif.com/dl/esp32_win32_msys2_environment_and_toolchain-20181001.zip)).

For debugging:

- OpenOCD (Open On-Chip Debugger) (<https://github.com/espressif/openocd-esp32/releases/download/v0.10.0-esp32-20190313/openocd-esp32-win32-0.10.0-esp32-20190313.zip>),
- Zadig (generic USB driver installer) (<https://github.com/pbatard/libwdi/releases/download/v1.3.0/zadig-2.3.exe>).

## 7.4 Known issues/limitations

- P0065ESP32WROVER-192: LLNET\_CHANNEL\_IMPL\_setOption cannot change the socket send and receive buffer sizes,
- P0065ESP32WROVER-184: Provided Filesystem pack does not support file write/read with offset from/to immortal arrays,
- P0065ESP32WROVER-183: Provided Filesystem pack does not support file backward skip,
- P0065ESP32WROVER-188: IPV6 is not supported,
- SystemView is enabled only on the single application platform (UID: HDAHT)
- OTA is enabled only on the single application platform (UID: HDAHT),
- ESP32 WROVER can be debugged with OpenOCD only if SD Card interface is not used,

- As described in espressif documentation, LCD and microSD cannot be used at the same time without unsoldering the resistor R167 (<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-wrover-kit.html#allocation-of-esp32-pins>).

# Index

## E

environment variable  
    PATH, 9

## P

PATH, 9