MicroEJ Platform Reference Implementation

Developer's Guide

ESP32WROVER 1.3.0

Reference:	TLT-0832-DGI-PlatformReferenceImplementation-ESP32WROVER
Version:	1.3.0
Revision:	1.0.0

Revision History		
Revision 1.0.0	September 27th 2018	
First release		

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Intended Audience

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

- This document is for the Espressif ESP-WROVER-KIT V3 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 https://developer.microej.com for more information about ESP-WROVER-KIT V3 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 V3 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 V3 board.

1.3. Prerequisites

- PC with Windows 7 or later.
- The MicroEJ SDK environment must be installed.
- ESP-WROVER-KIT V3 board.
- A GNU-GCC-based C development environment. The examples are packaged ready to run using the MicroEJ SDK IDE (including CDT packaging), which this document assumes has been successfully installed. Please visit the website mentioned above to obtain a version of the MicroEJ SDK IDE. Note, however, that developers are free to use a different CDT packaging.

Chapter 2. Create and Use Your First MicroEJ Platform

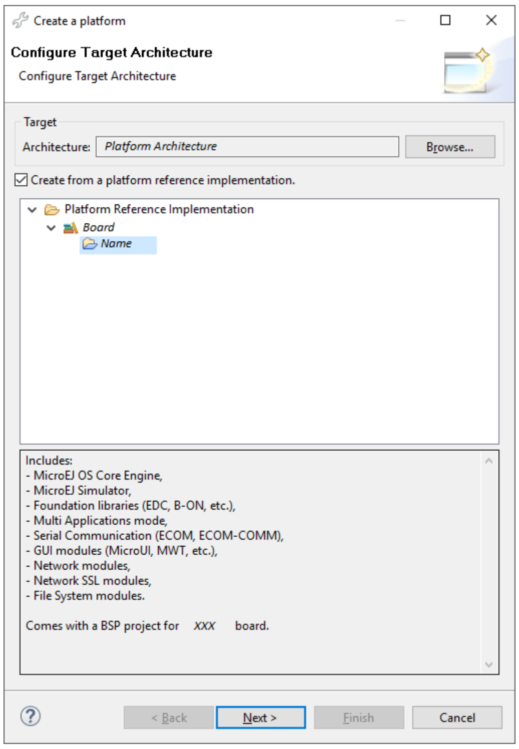
2.1. Create a MicroEJ Platform

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 V3.

- Open MicroEJ SDK.
- Open the MicroEJ platform wizard: `File > New > MicroEJ Platform Project`.
- Select the MicroEJ architecture ESP32 GCC PSRAM from the combo box. A MicroEJ Platform Reference Implementation is available:

Figure 2.1. MicroEJ Platform Reference Implementation Selection



Configure Target Architecture

Configure Target Architecture

Target

Architecture: *Platform Architecture*

☑ Create from a platform reference implementation.

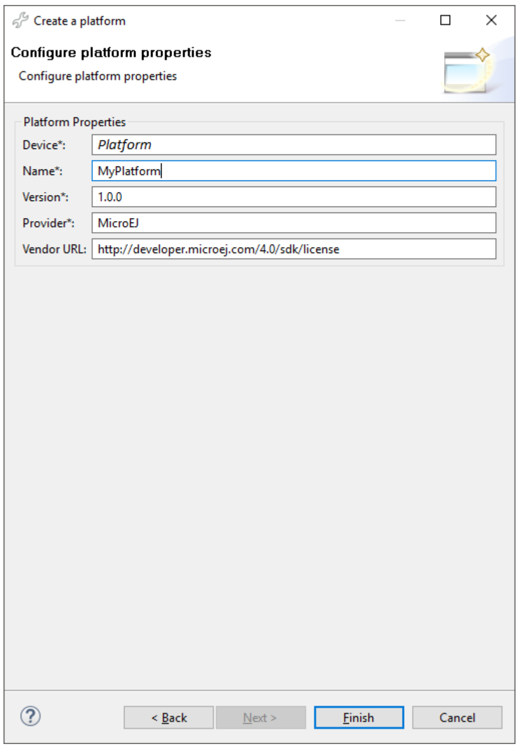
Platform Reference Implementation
Board
Name

Includes:
- MicroEJ OS Core Engine,
- MicroEJ Simulator,
- Foundation libraries (EDC, B-ON, etc.),
- Multi Applications mode,
- Serial Communication (ECOM, ECOM-COMM),
- GUI modules (MicroUI, MWT, etc.),
- Network modules,
- Network SSL modules,
- File System modules.

Comes with a BSP project for *XXX* board.

- Select the MicroEJ platform SingleApp for the ESP-WROVER-KIT V3 from the combo box.
- Click on Next. Give a name which be used as prefix for all MicroEJ platform projects. For instance: `MyPlatform`.

Figure 2.2. New MicroEJ Platform Naming



Device*: Platform
Name*: MyPlatform
Version*: 1.0.0
Provider*: MicroEJ
Vendor URL: http://developer.microej.com/4.0/sdk/license

- Click on `Finish`. The selected example is imported as several projects prefixed by the given name:

    - ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-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`).

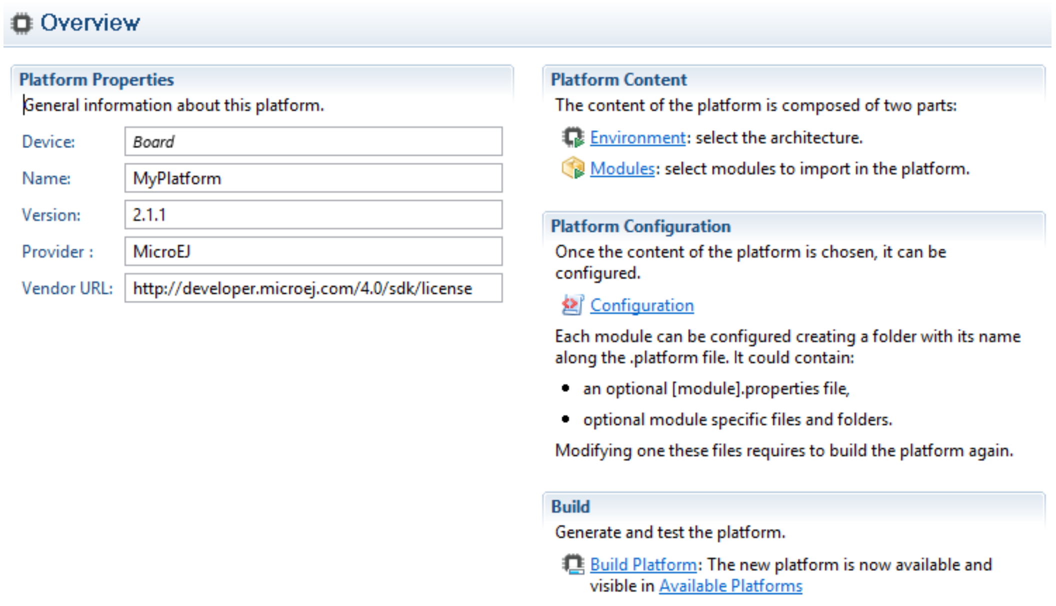
    - ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp: Contains a ready-to-use BSP software project for the ESP-WROVER-KIT V3 board, including a MicroEJ SDK project, an implementation of MicroEJ core engine (and extensions) port on FreeRTOS RTOS and the ESP-WROVER-KIT V3 board support package.

    - ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-fp: Contains the board description and images for the MicroEJ simulator. This project is updated once the platform is built.

  The MicroEJ platform configuration file is automatically opened.

- From the MicroEJ platform configuration file, click on the link `Build Platform`

Figure 2.3. MicroEJ Platform Build



Overview

Platform Properties

General information about this platform.

Device:	Board
Name:	MyPlatform
Version:	2.1.1
Provider :	MicroEJ
Vendor URL:	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

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 V3 board and is ready to be linked into the MicroEJ SDK project. Its name is `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram`.

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

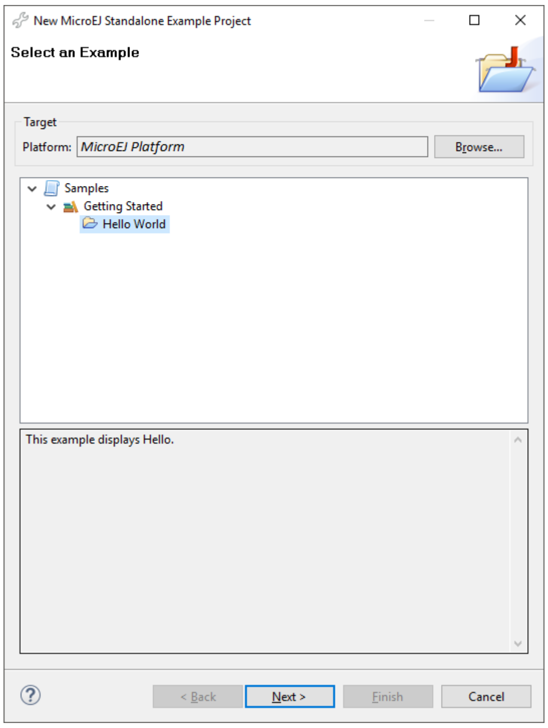
The aim of this chapter is to create a MicroEJ standalone application from a built-in example. Initially, this example will run on the MicroEJ simulator. Then, in the next section, this application will be compiled and deployed on the ESP-WROVER-KIT V3 board using the MicroEJ platform.

2.2.1. Create Example

- Open MicroEJ SDK.
- Open the `File > New > MicroEJ Standalone Example Project` menu.

- Select the MicroEJ platform ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram from the combo box.
- Select the example `Samples > Getting Started > Hello World`.

Figure 2.4. MicroEJ Standalone Application Selection



New MicroEJ Standalone Example Project

Select an Example

Target

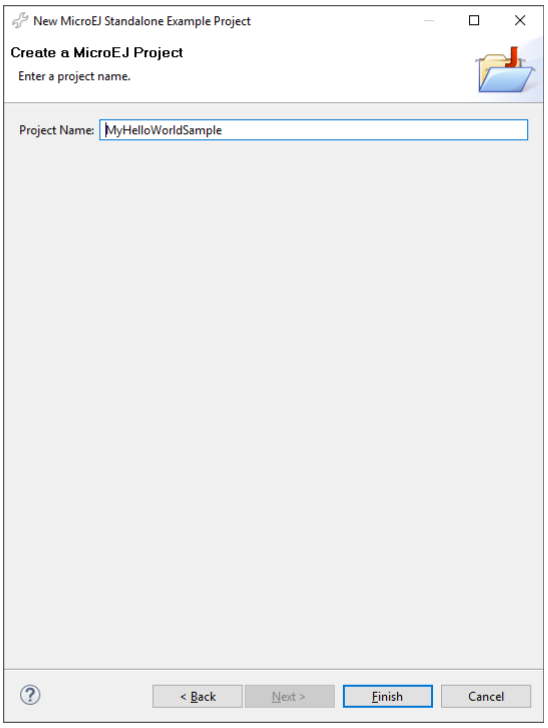
Platform: *MicroEJ Platform*

Samples
Getting Started
Hello World

This example displays Hello.

- Click on Next. The next page suggests a name for the new project.

Figure 2.5. MicroEJ Standalone Application Naming



New MicroEJ Standalone Example Project

Create a MicroEJ Project

Enter a project name.

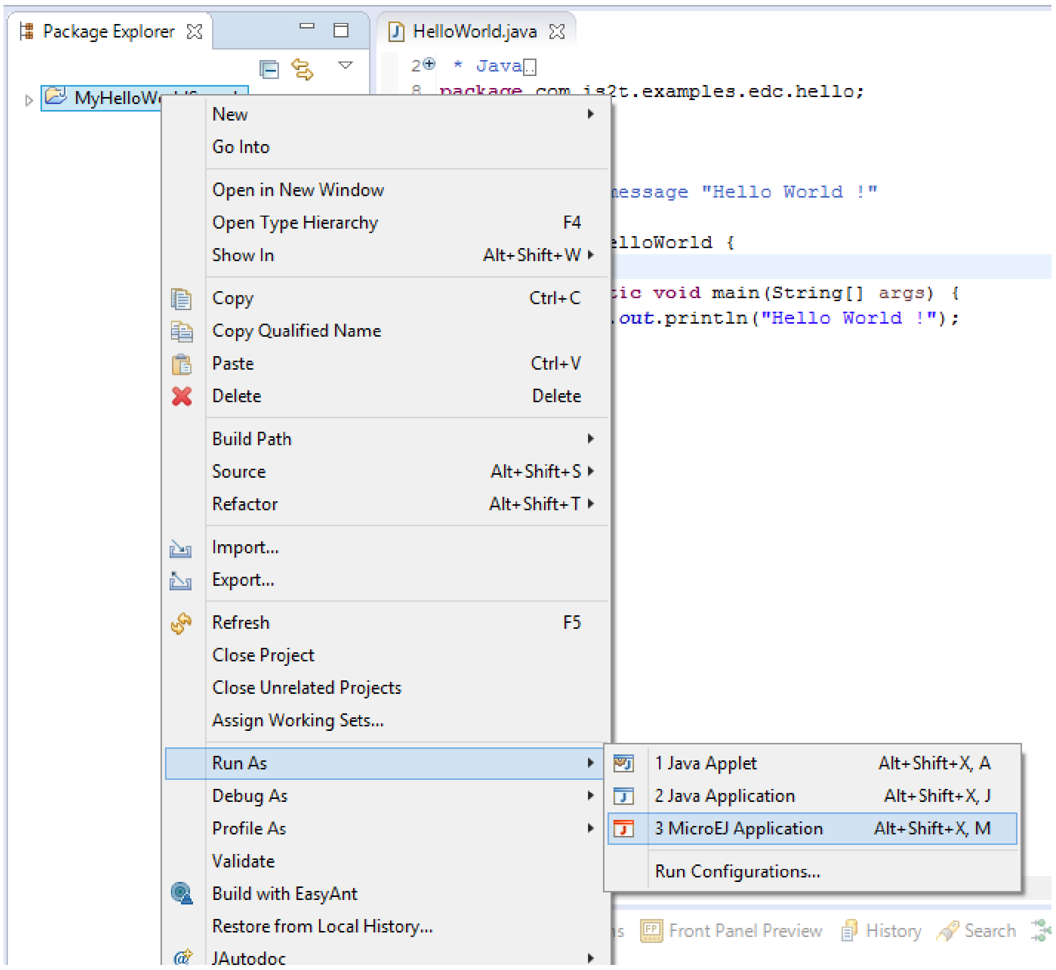
Project Name: MyHelloWorldSample

- Click on Finish. The selected example is imported into a project with the given name. The main class (the class which contains the `main()` method) is automatically opened.

2.2.2. Run Example

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

Figure 2.6. MicroEJ Standalone Application Running



```
2⊕ * Java
8 package com.is2t.examples.edc.hello;
essage "Hello World !"
elloWorld {
ic void main(String[] args) {
.out.println("Hello World !");
```

The application starts. It is executed on the MicroEJ simulator of the selected MicroEJ platform (ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram). The result of the test is printed in the console:

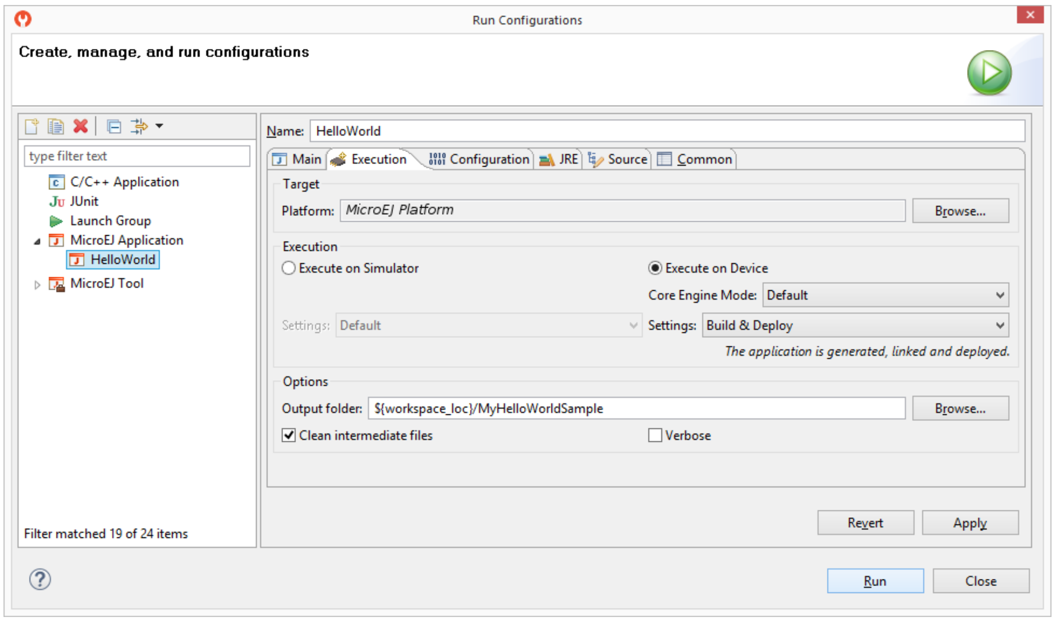
```
Hello World !
```

2.3. Run the Example on the ESP-WROVER-KIT V3 Board

2.3.1. Compile MicroEJ Standalone Application

- Open the run dialog (`Run > Run configurations...`).
- Select the MicroEJ Application launcher `HelloWorld`.
- Open `Execution` tab.
- Select `Execute on Device`.

Figure 2.7. Execution on Device



Name: HelloWorld

Target
Platform: MicroEJ Platform

Execution
( ) Execute on Simulator [✓ (•) Execute on Device]
Core Engine Mode: Default
Settings: Default
Settings: Build & Deploy
The application is generated, linked and deployed.

Options
Output folder: ${workspace_loc}/MyHelloWorldSample
[✓] Clean intermediate files
[ ] Verbose

- Open `Configuration` tab and sub menu `Target > Deploy`. By default, an option is set to deploy the application library at a location known by the third-party IDE. If you want to deploy it elsewhere, unselect this option and enter your output path in the field below.
- Click Run: the application is compiled, and the compilation result (an ELF file) is copied into a well known location in the workspace. The MicroEJ SDK BSP project will search for it there when it performs the final link.

2.3.2. Link and Deploy 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 V3 board.

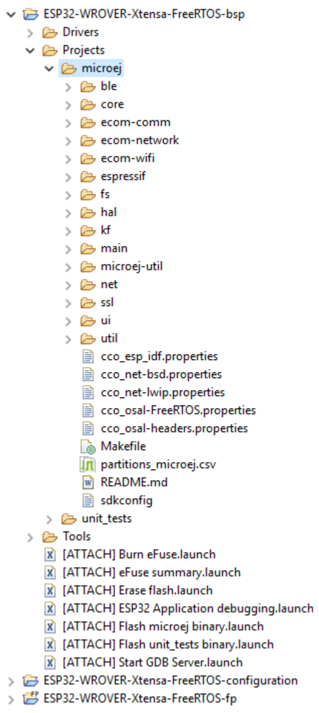
Note

This final step uses MicroEJ SDK.

The following steps are performed within MicroEJ.

- In MicroEJ SDK, expand the project ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp and the folder `Projects/microej`. A MicroEJ SDK project file (`.cproject`) is available.

Figure 2.8. MicroEJ SDK Project Selection



```
ESP32-WROVER-Xtensa-FreeRTOS-bsp
  Drivers
  Projects
    microej
      ble
      core
      ecom-comm
      ecom-network
      ecom-wifi
      espressif
      fs
      hal
      kf
      main
      microej-util
      net
      ssl
      ui
      util
      cco_esp_idf.properties
      cco_net-bsd.properties
      cco_net-lwip.properties
      cco_osal-FreeRTOS.properties
      cco_osal-headers.properties
      Makefile
      partitions_microej.csv
      README.md
      sdkconfig
    unit_tests
  Tools
  [ATTACH] Burn eFuse.launch
  [ATTACH] eFuse summary.launch
  [ATTACH] Erase flash.launch
  [ATTACH] ESP32 Application debugging.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
```

Espressif tools installation procedure:

- Download https://dl.espressif.com/dl/esp32_win32_msys2_environment_and_toolchain-20180110.zip and unzip its content to `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Tools`
- Download https://dl.espressif.com/dl/openocd-esp32-win32-a859564.zip and unzip its content to `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Tools`
- Download https://zadig.akeo.ie/downloads/zadig-2.3.exe and 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 refers to the following documentation: https://docs.espressif.com/projects/esp-idf/en/v3.0.4/get-started/index.html#setup-toolchain.

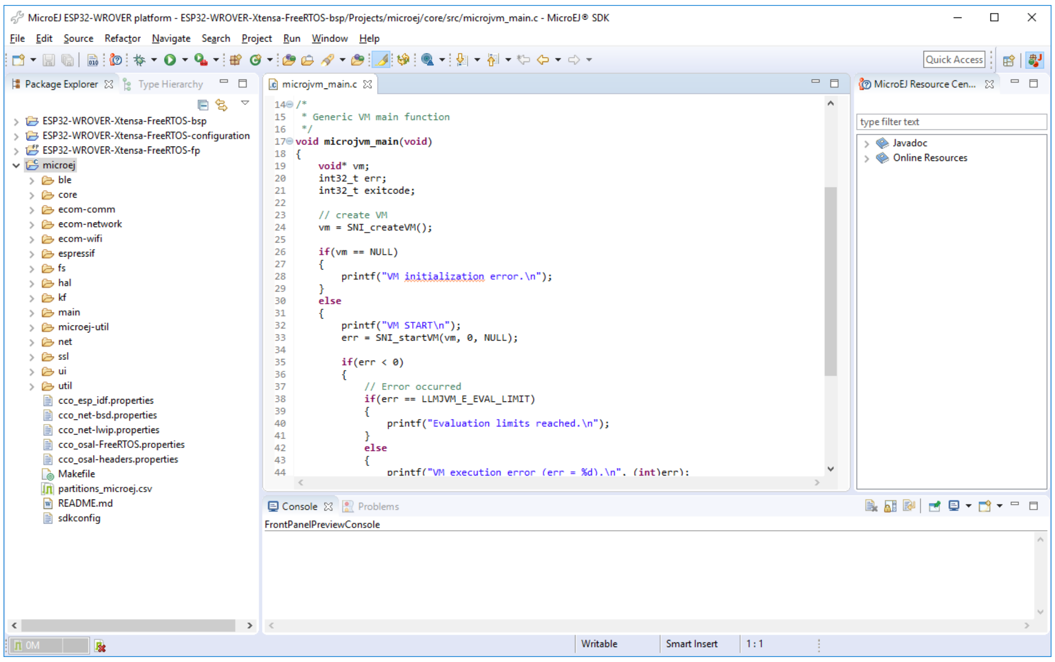
Import this project in your MicroEJ SDK workspace. Go to `File > Import... > General > Existing Projects into Workspace` and click on `Next` button. Click on the `Select root directory:` radio button and then on `Browse` button. Select the path associated to `ESP32WROVER-MyPlatform-GNUv52_xtensa-esp32-psram-bsp/Projects/microej` and click on `OK` button. Select the `microej` project in the Projects list. Verify that the `Copy projects into workspace` checkbox is not checked and 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`.

The following steps are performed within MicroEJ SDK.

Note

The target board can be flashed using Espressif bootloader. Please follow the Espressif documentation for more details (https://docs.espressif.com/projects/esp-idf/en/v3.0.4/get-started/index.html#build-and-flash)

- Figure 2.9. MicroEJ SDK IDE



MicroEJ ESP32-WROVER platform - ESP32-WROVER-Xtensa-FreeRTOS-bsp/Projects/microej/core/src/microjvm_main.c - MicroEJ® SDK

```
ESP32-WROVER-Xtensa-FreeRTOS-bsp
ESP32-WROVER-Xtensa-FreeRTOS-configuration
ESP32-WROVER-Xtensa-FreeRTOS-fp
microej
    ble
    core
    ecom-comm
    ecom-network
    ecom-wifi
    espressif
    fs
    hal
    kf
    main
    microej-util
    net
    ssl
    ui
    util
    cco_esp_idf.properties
    cco_net-bsd.properties
    cco_net-lwip.properties
    cco_osal-FreeRTOS.properties
    cco_osal-headers.properties
    Makefile
    partitions_microej.csv
    README.md
    sdkconfig
```

```c
/*
 * Generic VM main function
 */
void microjvm_main(void)
{
    void* vm;
    int32_t err;
    int32_t exitcode;

    // create VM
    vm = SNI_createVM();

    if(vm == NULL)
    {
        printf("VM initialization error.\n");
    }
    else
    {
        printf("VM START\n");
        err = SNI_startVM(vm, 0, NULL);

        if(err < 0)
        {
            // Error occurred
            if(err == LLMJVM_E_EVAL_LIMIT)
            {
                printf("Evaluation limits reached.\n");
            }
            else
            {
                printf("VM execution error (err = %d).\n", (int)err);
```

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.

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 V3 is based on board support package provided by Espressif: (esp-idf-3.0.zip SourceForge website [https://github.com/espressif/esp-idf/releases/tag/v3.0]). It includes FreeRTOS, a graphical user interface, a TCP/IP network connection, a file system on SD-Card and some custom GPIOs. MicroEJ platform has been built MicroEJ SDK 4.1.5 IDE.

3.2. MicroEJ Platform Configuration

MicroEJ platform is based on MicroEJ architecture for ESP32.

Table 3.1. MCU Technical Specifications

MCU architecture	Xtensa LX6 dual-core (ESP32-D0WDQ6)
MCU Clock speed	240 MHz
Internal RAM	520 KB
External Flash	4 MB (QSPI)
External RAM	4 MB (PSRAM)

MicroEJ platform uses several architecture extensions. The following table illustrates the MicroEJ architecture and extensions versions.

Table 3.2. MicroEJ Configuration

Name	Version
MicroEJ architecture	7.9.0
UI	11.1.2
Network	8.1.4
File System	4.0.2
HAL	2.0.1
Network Addons	2.0.3

3.3. Platform Output stream

MicroEJ platform uses JP4 connector as output print stream. This COM port is connected to the DC0 for device TX and GND.

Implementation Note

The COM port is also used as the output stream for the *printf* calls.

The COM port uses the following parameters:

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

3.4. RTOS Configuration

MicroEJ platform uses FreeRTOS 8.2.0. RTOS uses a heap to allocate all its objects: tasks stacks, task monitors, semaphores etc. The heap size is: 45 KB and is allocated in internal RAM. The following table illustrates the available tasks and their stack size.

Table 3.3. FreeRTOS Tasks

Task name	Size	Priority
RTOS idle task	1 KB	0
RTOS timer	2 KB	24
Core Engine	21 KB	8
SPI Master	7 KB	9
Network Delegate	4 KB	9
Async Netconn	2 KB	12
Filesystem Delegate	7 KB	12
LCD Transfer	3 KB	9
Framerate	3 KB	3
System Monitor	4 KB	23
LwIP TCP/IP	2.5 KB	18
BlueDroid btc	3.5 KB	19
BlueDroid btu	4.5 KB	20
BlueDroid hciH4	2.5 KB	21
BlueDroid hci	2.5 KB	22
Application main	4 KB	1
CPU dport	768 B	5
ESP Timer	4 KB	22
Event loop	2.5 KB	20
CPU ipc	1 KB	24

3.5. Memories

MicroEJ Plaform uses several internal and external memories. The following table illustrates the MCU and board memory layouts and sizes fixed by the MicroEJ platform.

Table 3.4. Internal RAM (520 KB)

Section Name	Size
MicroEJ standalone application stack blocks	512 * *n* bytes [a]
Pre-installed MicroEJ sandboxed application	*n* bytes [b]
MicroEJ platform internal heap	*n* bytes [c]
Any RW	*n* bytes [d]
MicroEJ standalone application heaps	1536 KB [e]

[a] *n* is the number of stack blocks defined in MicroEJ Application launcher options.

[b] *n* depends on the size defined in MicroEJ Application launcher options.

[c] *n* depends on memory configuration set in MicroEJ Application launcher options.

[d] *n* depends on MicroEJ application libraries used.

[e] Maximum size of the addition of MicroEJ heap size and MicroEJ immortal heap size. These sizes are defined in MicroEJ Application launcher options.

Table 3.5. External RAM: PSRAM (4 MB)

Section Name	Size
Display buffers	150 KB
Multi applications working buffer	1 MB
SSL buffers	Linked to C malloc heap

Table 3.6. External flash: QSPI (4 MB)

Section Name	Size
Any RO	*n* bytes [a]
MicroEJ standalone application resources	*n* bytes [b]

[a] *n* depends on MicroEJ application, MicroEJ libraries, Board support package, RTOS, drivers, etc.

[b] *n* is the size of all MicroEJ standalone application resources.

3.6. Graphical User Interface

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

3.6.1. 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.

Implementation 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 `150 KB`.

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.

Implementation Note

This buffer is called "images heap". An image buffer size follows the same rule than the LCD buffer (see before).

3.6.2. 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

3.7. Network

MicroEJ plaform 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.

Implementation 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.

3.8. 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.

Implementation Note

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

3.9. File System

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

Implementation Note

MicroEJ platform uses FatFS R0.13a.

3.10. Serial Communications

3.10.1. UART Connector

MicroEJ platform does not provide any serial connection.

3.11. HAL

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

3.12. 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 V3 provides several connectors, each connector is used by the MicroEJ Core Engine itself or by a foundation library.

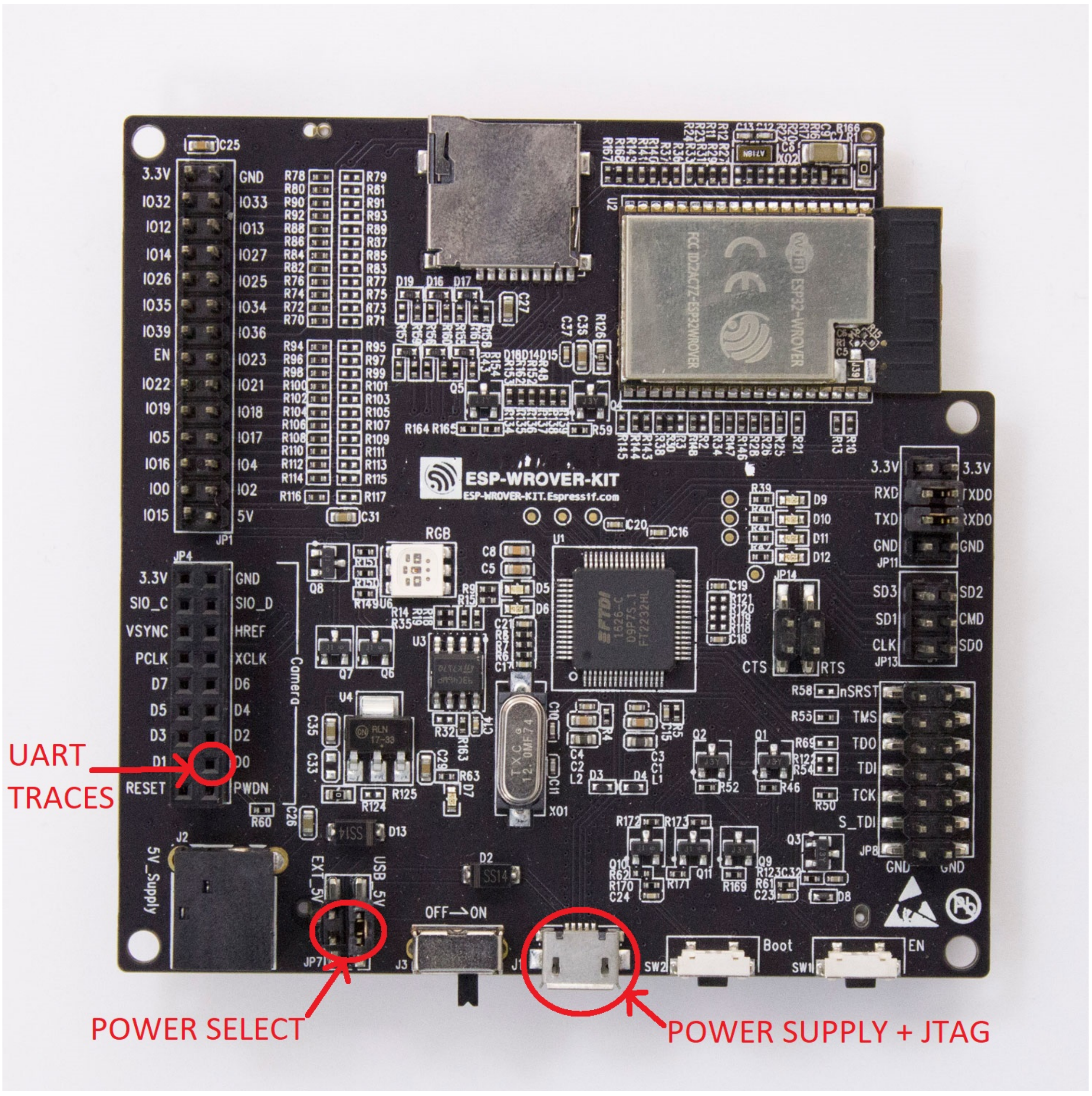
4.1. Mandatory Connectors

ESP-WROVER-KIT V3 provides a multi function USB port used as:

- Power supply connector
- Probe connector
- Virtual COM port (for Espressif bootloader traces only)

First of all, take a FTDI USB wire. Connect it to your PC and launch a the serial sniffer software of your choice. Link FTDI RX on D0 pin of the JP4 connector of the ESP-WROVER-KIT V3 and do not forget the ground. 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.

Figure 4.1. Mandatory Connectors



Chapter 5. MicroEJ SDK Configuration

5.1. Install MicroEJ SDK

This section describes how to install a MicroEJ SDK development environment.

5.1.1. Download MicroEJ SDK

- Go to `https://developer.microej.com/getting-started-sdk.html`.
- Press the `Download MicroEJ SDK` button.
- Download the executable file (e.g. `MicroEJ-SDK-Installer-Win64-4.1.5.exe`).
- Run executable file and follow the installation steps. A new application named `MicroEJ SDK 4.1.5` shall have been installed.

Chapter 6. Changelog

6.1. Version 1.3.0

Features:

- 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

Bug fixes:

- 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

6.2. Version 1.2.0

Features:

- 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.

Bug fixes:

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

6.3. Version 1.1.0

Features:

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

Bug fixes:

- Failures when trying to attach a GDB debug session.

6.4. Version 1.0.0

Initial release of the platform.