MicroEJ Platform

Developer's Guide

MICROEJ®

OM13092 1.1.0

Reference:	TLT-0826-DGI-Platform-OM13092
Version:	1.1.0
Revision:	1.1.0

Confidentiality & Intellectual Property

Revision History		
Revision 1.1.0	December 28th 2017	
Changelog updated for release 1.1.0		
Revision 1.0.0	December 21st 2017	
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 standalone application with MicroEJ SDK. Notes:

- This document is for the NXP OM13092 board.
- Please visit the website https://developer.microej.com for more information about OM13092 products (platforms, videos, examples, application notes, etc.).

1.2. Scope

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

- Run a MicroEJ standalone application on the MicroEJ simulator.
- Run a MicroEJ standalone application on the MicroEJ platform and deploy it on the OM13092 board.

1.3. Prerequisites

- PC with Windows 7 or later.
- The MicroEJ SDK environment must be installed.
- OM13092 board.
- The Segger J-Link software.
- IAR Embedded Workbench for ARM 7.80.4. To get IAR Embedded Workbench for ARM, please visit the IAR website.

Chapter 2. Develop and Run Your First MicroEJ Standalone Application

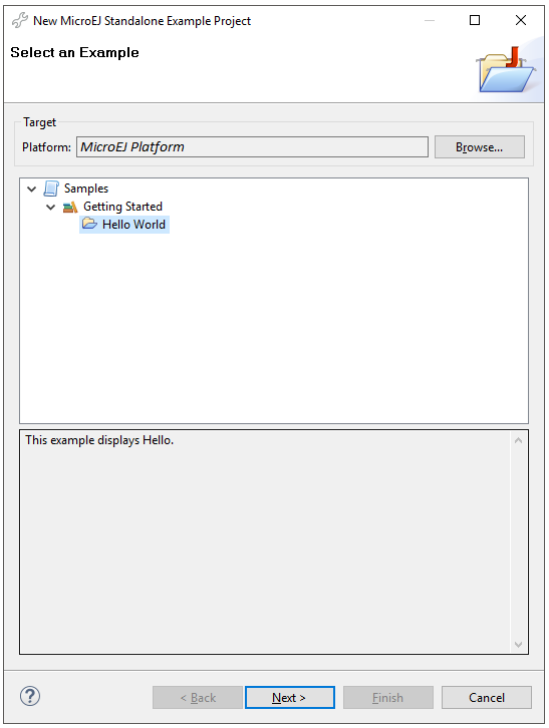
2.1. Run an Example on the MicroEJ Simulator

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 OM13092 board using the MicroEJ platform.

2.1.1. Create Example

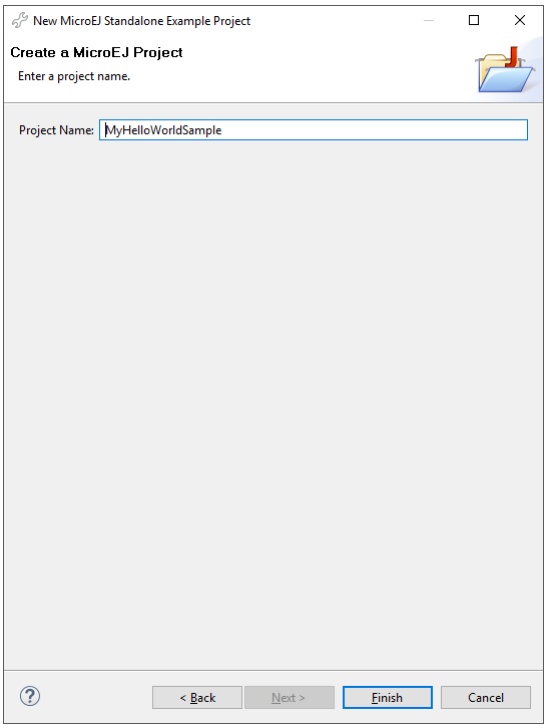
- Open MicroEJ SDK.
- Open the `File > New > MicroEJ Standalone Example Project` menu.
- Select the MicroEJ platform OM13092-U3OER from the combo box.
- Select the example `Samples > Getting Started > Hello World`.

Figure 2.1. MicroEJ Standalone Application Selection



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

Figure 2.2. MicroEJ Standalone Application Naming

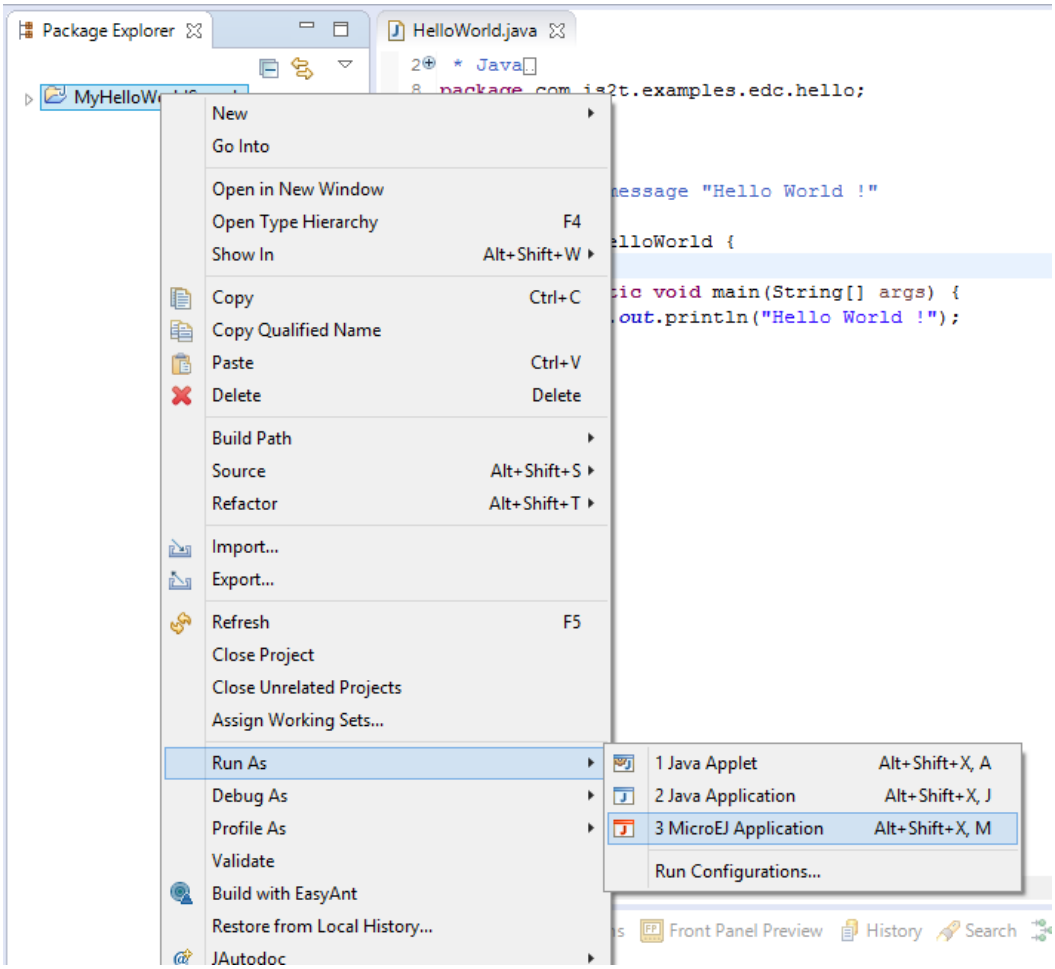


- Click on Finish. The selected example is imported into a project with the given name. The main class (the class which contains the `main()` method) is automatically opened.

2.1.2. Run Example

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

Figure 2.3. MicroEJ Standalone Application Running



The application starts. It is executed on the MicroEJ simulator of the selected MicroEJ platform (OM13092-U3OER). The result of the test is printed in the console:

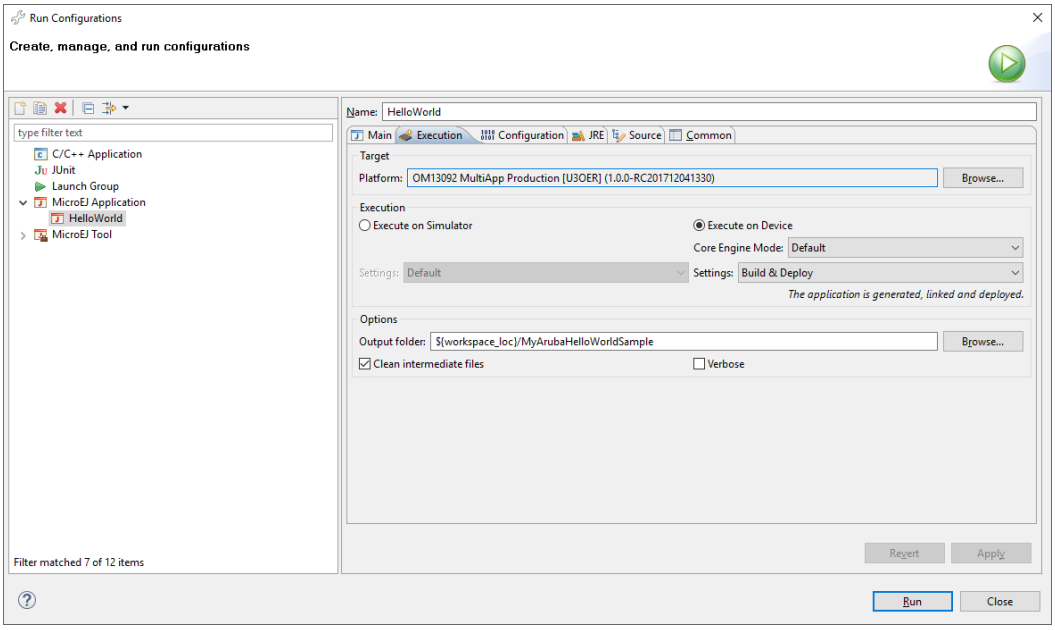
```
Hello World !
```

2.2. Run the Example on the OM13092 Board

2.2.1. 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 2.4. Execution on Device



- In the JRE tab, pass the following VM argument: `-Dtoolchain.dir` and set its value to the path to the toolchain directory of IAR (typically `C:\Program Files (x86)\IAR Systems\Embedded Workbench 7.80\arm\bin`).

  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 SEGGER J-Link ARM tool has to be used to load the program on the board.

2.2.2. SEGGER J-Link Programming Tool

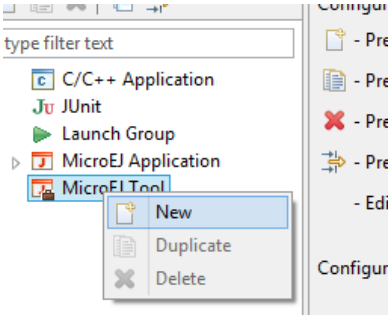
The aim of this section is to program a binary on the OM13092 board.

Prerequisites

Download SEGGER J-Link ARM software and documentation pack from `https://www.segger.com/jlink-software.html` and install it on your machine.

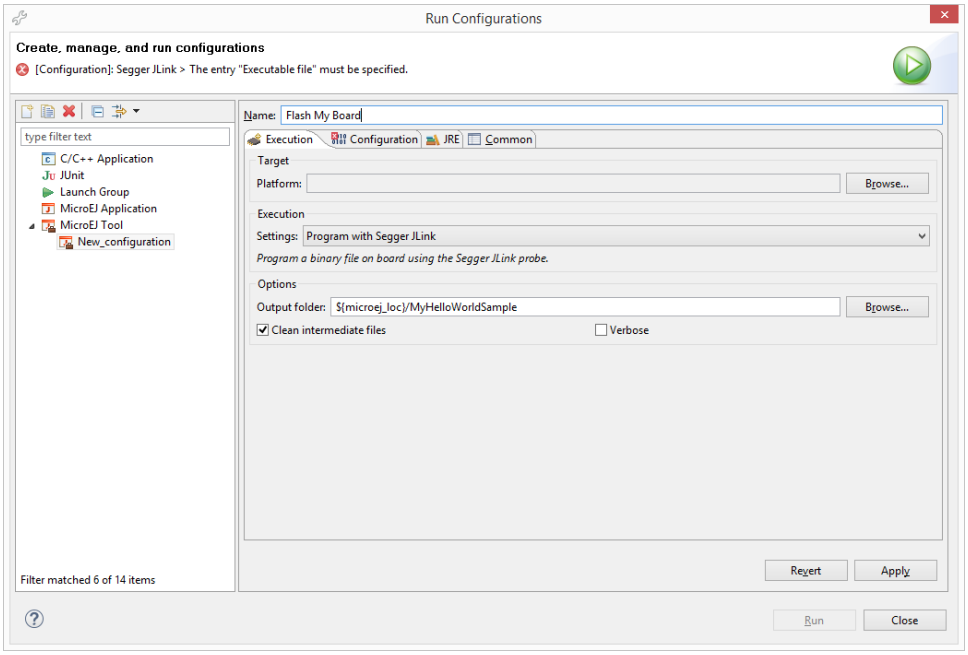
- Click on `Run > Run Configurations...`. Then right click on sub menu of `MicroEJ Tool` and select `New` to create a new MicroEJ Tool launcher:

Figure 2.5. MicroEJ Tool Launcher Creation



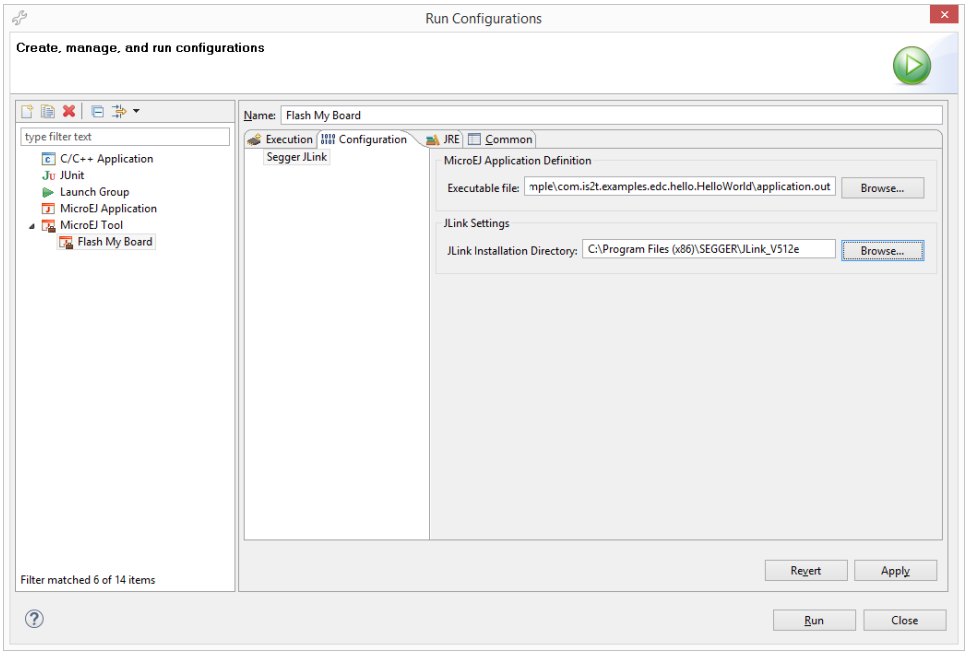
- A new window appears. Give a name to the launcher and set the `MicroEJ platform` field to `OM13092-U3OER` and the `Settings` field to `Program with Segger J-Link`

Figure 2.6. SEGGER J-Link MicroEJ SDK Tool Window



- Click on `Configuration` tab select the `application.out` file available in the MicroEJ project.

Figure 2.7. SEGGER J-Link MicroEJ SDK Tool Configuration Window



- Click on Run to program the binary.

  At the end of the execution the following message appears:

```
Flash programming complete successfully.
```

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 OM13092 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 ARM Cortex-M4.

Table 3.1. MCU Technical Specifications

MCU architecture	Cortex-M4 (LPC546xx)
MCU Clock speed	180 MHz
Internal Flash	512 KB
Internal RAM	256 KB
Internal EEPROM	16 KB
External Flash	16 MB (QSPI)
External RAM	8 MB

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	6.18.0
UI	10.0.0
Network	6.1.5
File System	3.0.2
HAL	1.0.6

3.3. Platform Output stream

MicroEJ platform uses a USB Virtual COM port as output print stream. The virtual COM port is available on the USB Debug-Link/J8 connector.

The COM port uses the following parameters:

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

3.4. 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 3.3. Internal RAM (256 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 3.4. External RAM: SDRAM (8 MB)

Section Name	Size
Display buffers	510 KB
MicroUI working buffer	2586 KB
Multi applications working buffer	3 MB
SSL buffers	130 KB
Ethernet buffers	77 KB

Table 3.5. Internal flash: Program Flash (512 KB)

Section Name	Size
Any RO	*n* bytes [a]

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

Table 3.6. External flash: QSPI (16 MB)

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

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

3.5. Multi Applications

This MicroEJ platform includes the Multi applications mode. Multi applications mode allows to build a firmware that can manage MicroEJ sandboxed applications. Multi applications mode requires a specific memory area to load MicroEJ sandboxed applications. This memory area is located in external SDRAM and its default size is 3 MB.

3.6. Graphical User Interface

MicroEJ plaform features a graphical user interface. It includes a display, a touch panel, an user button and a runtime PNG decoder.

3.6.1. Display

The display module drives a 480 x 272 LCD display. The pixel format is 16 bits-per-pixel: 5 bits for red color component, 6 bits for green color component and 5 bits for blue color component. The display device is clocked at 60Hz and the MicroEJ application drawings are synchronized on this display tick.

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 SDRAM and the reserved size is 2586 Kbytes.

3.6.2. Inputs

Touch panel: All touch panel events are sent to the MicroEJ application thanks a `Pointer` event generator.

User buttons: The user buttons are reserved to the multi applications feature: they allow to force the kill of a sandboxed application.

3.7. Network

MicroEJ plaform features a network interface. A limited number of 10 sockets could be used for TCP connections, 5 for TCP listening (server) connections and 6 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.

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, DTLS 1.0, DTLS 1.2. Keys and certificates supported formats are PEM, DER and PKCS#12.

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.

3.10. Serial Communications

3.10.1. UART Connector

MicroEJ platform does not provide any serial connection.

3.11. HAL

MicroEJ platform includes four expansion connectors (J9, J10, J12 and J13) incorporate an Arduino Uno revision 3 footprint in their inner rows. These connectors provide access to the CAN interfaces, additional digital microphone support signals, I2S, I2C, USART, SPI and GPIO/INT/PWM connections. Note that several of the signals available at these connectors are shared with other connectors or board functions, so may not be usable if those other functions are being used.

MicroEJ platform provides several GPIOs to connect HAL foundation library. All GPIOs that are available on the expansion connectors (J9, J10, J12 and J13), and can be used for external access, are usable through the HAL foundation library. Digital pins are implemented by a GPIO access, and none of the available GPIOs are connected to any ADC channels, so they cannot operate in analog mode.

Each GPIO port / pin value is accessible by several ways:

1. Using the global MCU declaration: all pins of all ports are grouped under only one virtual port (port 0) and have consecutive values: PT0_0 has the ID 0, PT0_1, the ID 1, PT1_0 the ID 32 and so on. For instance pin *PT3_11* is accessible by `(0, 107)`. This declaration is useful to target all MCU pins using only one virtual port.

2. Using the standard MCU declaration: PORT0 has the ID 1, PORT1 the ID 2 etc. Each pin of each port is a value between 0 (Port*N*-0) to 31 (Port*N*-31). For instance pin *PT3_11* is accessible by `(4, 11)`. This declaration is useful to target a specific MCU pin.

3. Using the virtual board connectors: `ARDUINO_DIGITAL` and `ARDUINO_ANALOG`. These connectors make an abstraction between the MicroEJ application and the HAL implementation. For instance pin `PT3_11` is accessible on connector `ARDUINO_DIGITAL`, pin 47 (there are 20 pins on connector J9, 20 pins on connector J10, and `PT3_11` is connected to J12, pin7).

4. 4. Using the physical board connectors. The board has 4 connectors: J9, J10, J12 and J13, with respectively these IDs: 64, 65, 66 and 67. For instance pin `PT3_11` is accessible on connector J12, pin7: `(66, 7)`. this declaration is useful to target a physical connector pin without knowing which MCU pin it is.

The following table summaries the exhaustive list of GPIOs ports accessible from HAL library, and the ranges of pins IDs:

Table 3.7. HAL GPIOs Ports and Pins

Port name	HAL port ID	Pins range
Global MCU virtual port	0	0 to 170
MCU port 0	1	0 to 31
MCU port 1	2	0 to 31
MCU port 2	3	0 to 31
MCU port 3	4	0 to 31
MCU port 4	5	0 to 31
MCU port 5	6	0 to 10
Board virtual port `DIGITAL`	30	1 to 72
Board virtual port `ANALOG`	31	1 to 72
Board physical port `J9`	64	1 to 20
Board physical port `J10`	65	1 to 20
Board physical port `J12`	66	1 to 12
Board physical port `J13`	67	1 to 20

The following table illustrates the exhaustive list of GPIOs connected to the HAL library, their IDs according the ports IDs and pins IDs (see before). This table indicates too the useful ADC / DAC channels for HAL analog pins:

Table 3.8. HAL GPIOs Declaration (port, pin)

Port / Pin	MCU virtual port (1)	MCU port (2)	Digital virtual connector (3)	Analog virtual connector (3)	Board physical port (4)
P1_17	0, 50	2, 17	30, 4	--	64, 4
P1_18	0, 51	2, 18	30, 2	--	64, 2
P2_17	0, 82	3, 17	30, 72	--	67, 20
P3_11	0, 108	4, 11	30, 47	--	66, 7
P3_12	0, 109	4, 12	30, 49	--	66, 9
P3_16	0, 113	4, 16	30, 43	--	66, 3
P3_17	0, 114	4, 17	30, 70	--	67, 18
P3_28	0, 125	4, 28	30, 68	--	67, 16

Port / Pin	MCU virtual port (1)	MCU port (2)	Digital virtual connector (3)	Analog virtual connector (3)	Board physical port (4)
P3_29	0, 126	4, 29	30, 66	--	67, 14
P4_2	0, 130	5, 2	30, 10	--	64, 10
P4_4	0, 132	5, 4	30, 71	--	67, 19
P4_6	0, 134	5, 6	30, 41	--	66, 1

None of the pins available on the Arduino ports are connected to any ADC channel, therefore they cannot be used in analog configuration.

Chapter 4. Foundation Libraries

4.1. List

This table illustrates the available foundation libraries in the MicroEJ platform, and their versions.

Table 4.1. Foundation Libraries

Name	Version
EDC	1.2
BON	1.3
ECOM	1.1
ECOM-COMM	1.1
NLS	2.0
KF	1.4
MicroUI	2.2
NET	1.1
SSL	2.0
FS	2.0

Chapter 5. Board Configuration

OM13092 provides several connectors, each connector is used by the MicroEJ Core Engine itself or by a foundation library.

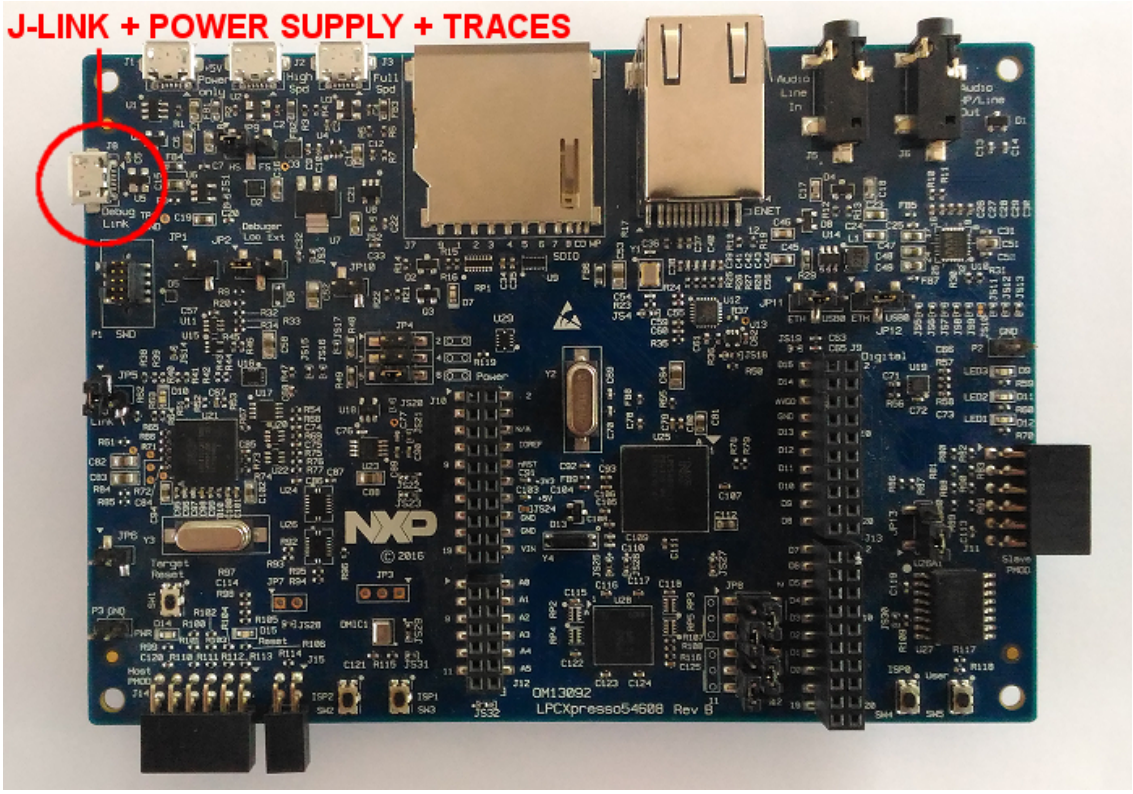
5.1. Mandatory Connectors

OM13092 provides a multi function USB port used as:

- Power supply connector
- Probe connector

Plug a USB type B cable to a computer to power on the board, be able to program an application on it and to see the MicroEJ standalone application `System.out.print` traces.

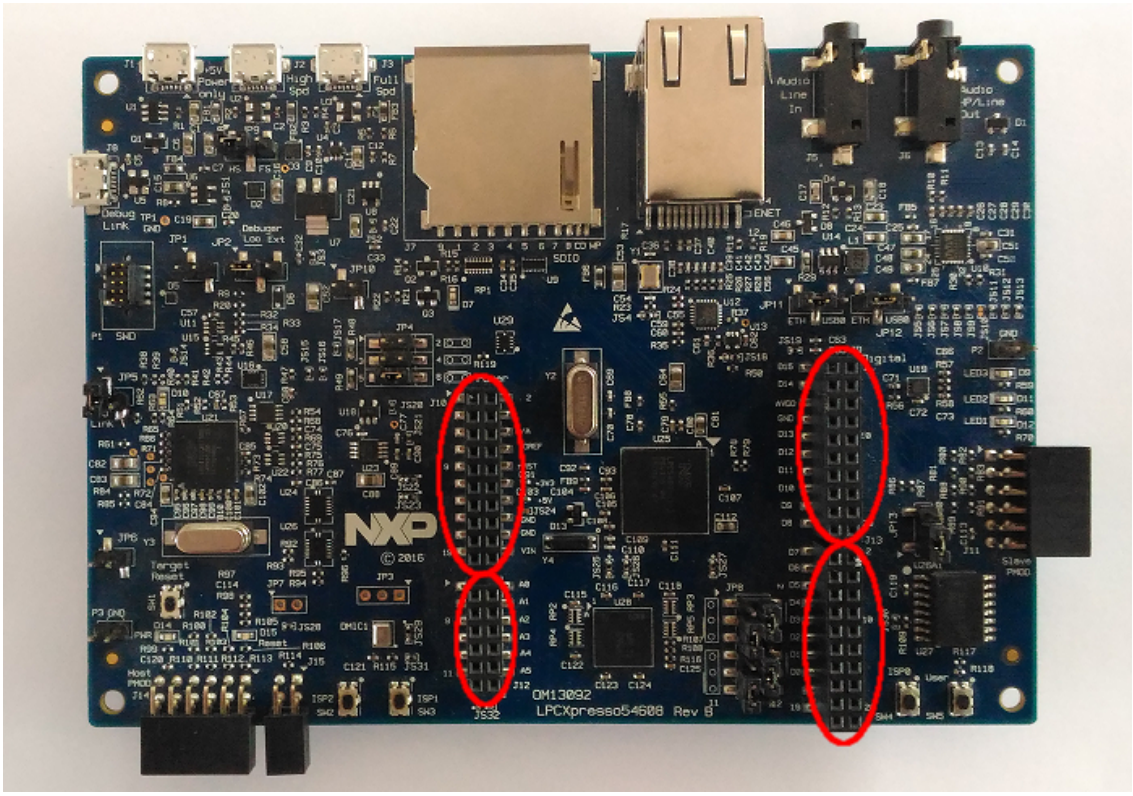
Figure 5.1. Mandatory Connectors



5.2. HAL Connectors

OM13092 provides several HAL GPIOs on Arduino connectors

Figure 5.2. HAL Connectors



Chapter 6. Changelog

6.1. Version 1.1.0

This release brings an update for these modules:

- Architecture: Updated from 6.9.0 to 6.18.0, bringing performance improvements and bug fixes.
- UI: Updated from 9.0.2 to 10.0.0, bringing performance improvements and bug fixes.
- FS: Updated from 3.0.0 to 3.0.2, bringing bug fixes.
- HAL: Updated from 1.0.4 to 1.0.6, bringing bug fixes.

6.2. Version 1.0.0

Initial release of the platform.