MicroEJ Platform Reference Implementation

Developer's Guide

GRPEACH 1.5.1

Reference:	TLT-813-DGI-PlatformReferenceImplementation-GRPEACH
Version:	1.5.1
Revision:	1.5.1

Confidentiality & Intellectual Property

Revision History		
Revision 1.5.1	Decemner 28th 2017	
Missing documentation.		
Revision 1.5.0	October 26th 2017	
Migrated to Architecture 6.17.2. Use pack UI 9.3.1 instead of 9.0.2.		
Revision 1.4.3	July 5th 2017	
Use pack Net 6.1.5 instead of 6.1.4		
Revision 1.4.2	July 5th 2017	
Soft DNS resolver doesn't retry on timeout and doesn't try other servers on errors		
Revision 1.4.1	June 8th 2017	
Increase SSL memory pool		
Revision 1.4.0	Apr 28th 2017	
Upgrade platform architecture		
Revision 1.3.0	Apr 6th 2017	
Upgrade platform architecture		
Revision 1.2.0	March 30th 2017	
Add HAL module low level interface implementation BugFix : Error on SSL certificate check.		
Revision 1.1.1	March 15th 2017	
BugFix : Initialization of serial handler should be done before enable IRQ.		
Revision 1.1.0	March 8th 2017	
Fixed space in MicroEJ repository path causing link errors, Change address MAC customization in flash memory by UID customization, Change output trace baudrate to 115200.		
Revision 1.0.0	January 10th 2017	
Initial version		

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

1.3. Prerequisites

- PC with Windows 7 or later.
- The MicroEJ SDK environment must be installed.
- GRPEACH board.
- A GNU-C development environment. The examples are packaged ready to run using the e2studio 5.1.0.022 C IDE, which this document assumes has been successfully installed. Please visit the Renesas website to obtain a version of the e2studio C IDE. Note, however, that developers are free to use a different CDT packaging.
- GNU toolchain: GNU ARM Embedded Toolchain 4.8.4_20140725

Chapter 2. Create and Use Your First MicroEJ Platform

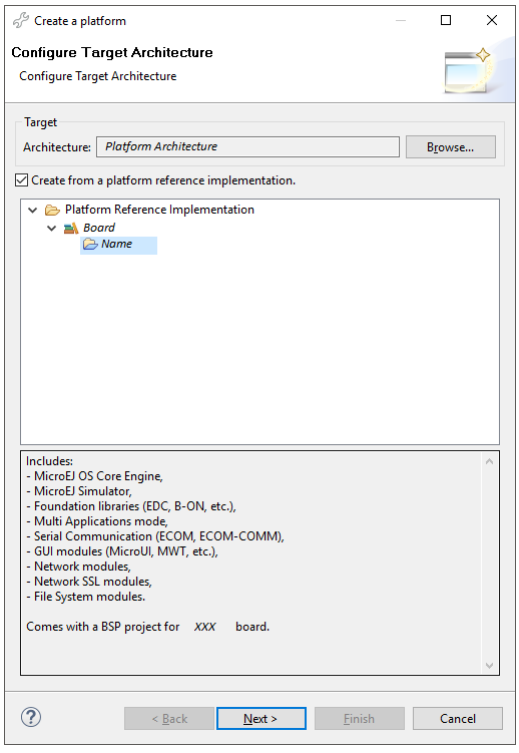
2.1. Create a MicroEJ Platform

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 GRPEACH.

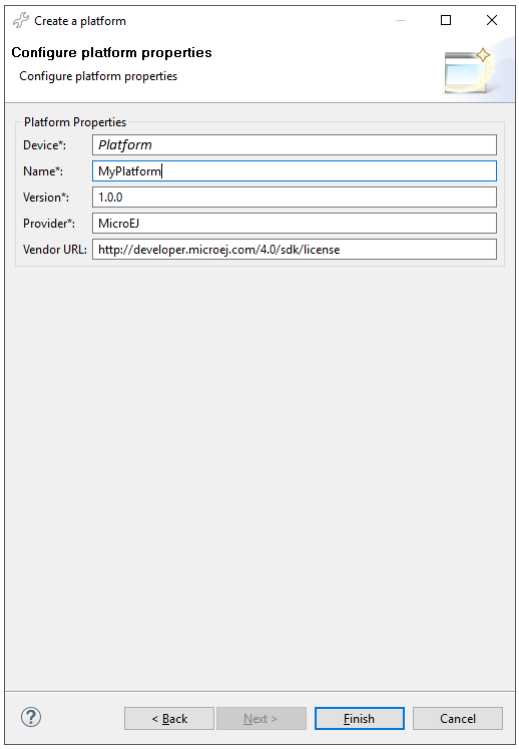
- Open MicroEJ SDK.
- Open the MicroEJ platform wizard: `File > New > MicroEJ Platform Project`.
- Select the MicroEJ architecture ARM Cortex-A9 GCC from the combo box. A MicroEJ Platform Reference Implementation is available:

Figure 2.1. MicroEJ Platform Reference Implementation Selection



- Select the MicroEJ platform SingleApp for the GRPEACH 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



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

  - GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi-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`).

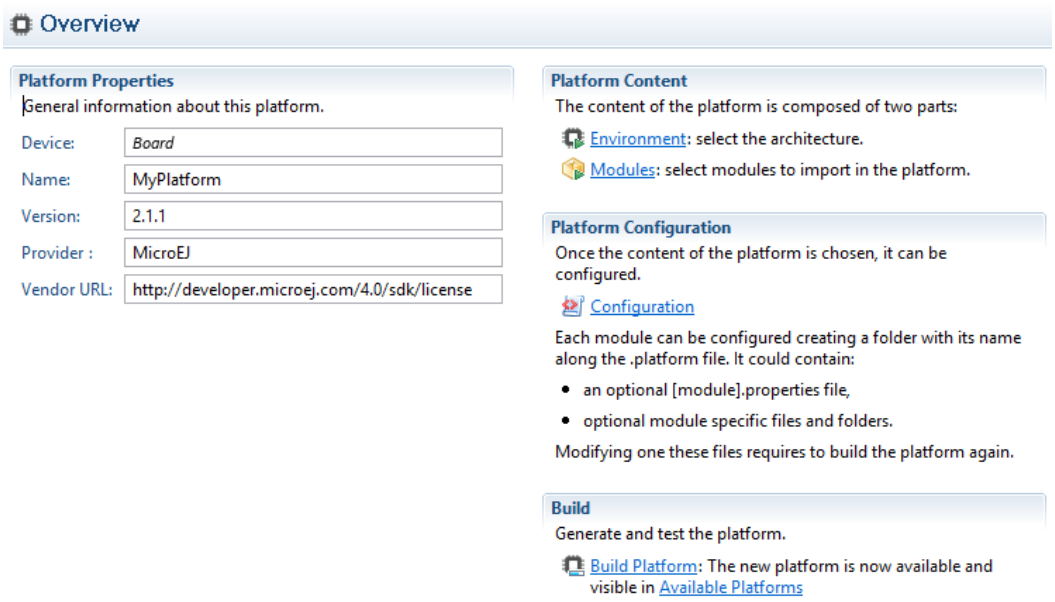
  - GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi-bsp: Contains a ready-to-use BSP software project for the GRPEACH board, including a Renesas e2 studio project, an implementation of MicroEJ core engine (and extensions) port on Mbed-OS RTOS and the GRPEACH board support package.

  - GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi-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



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 GRPEACH board and is ready to be linked into the Renesas e2 studio project. Its name is `GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi`.

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

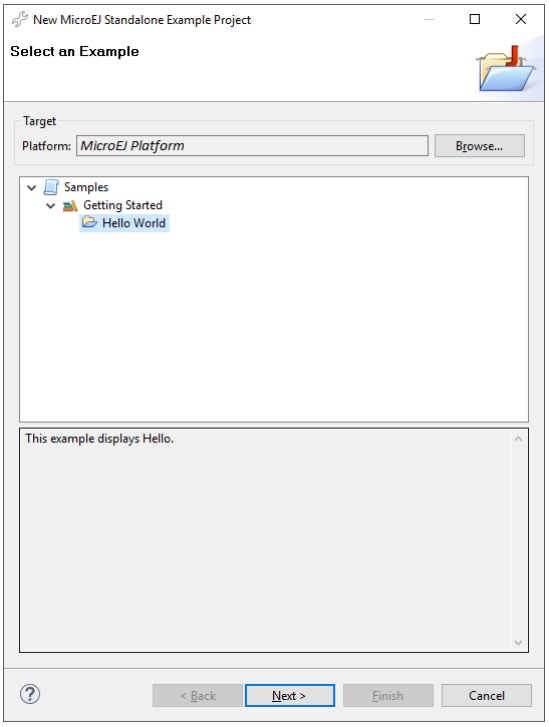
The aim of this chapter is to create a MicroEJ standalone application from a built-in example. Initially, this example will run on the MicroEJ simulator. Then, in the next section, this application will be compiled and deployed on the GRPEACH board using the MicroEJ platform.

2.2.1. Create Example

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

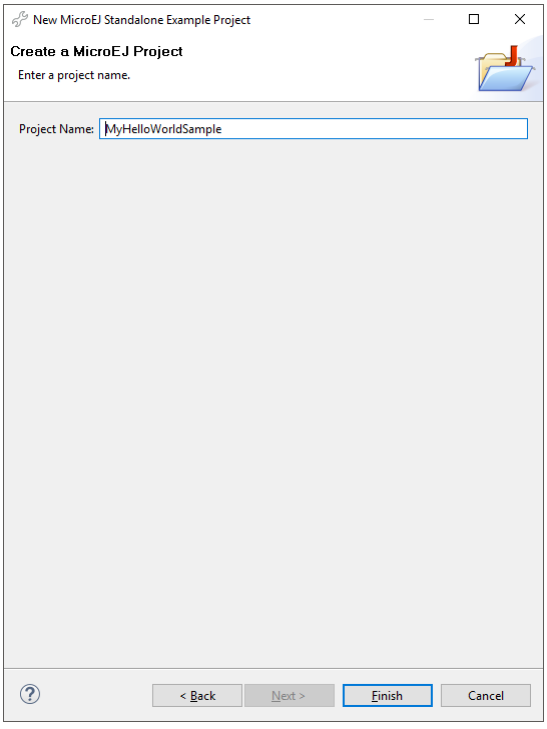
- Select the MicroEJ platform GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi from the combo box.
- Select the example `Samples > Getting Started > Hello World`.

Figure 2.4. MicroEJ Standalone Application Selection



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

Figure 2.5. MicroEJ Standalone Application Naming

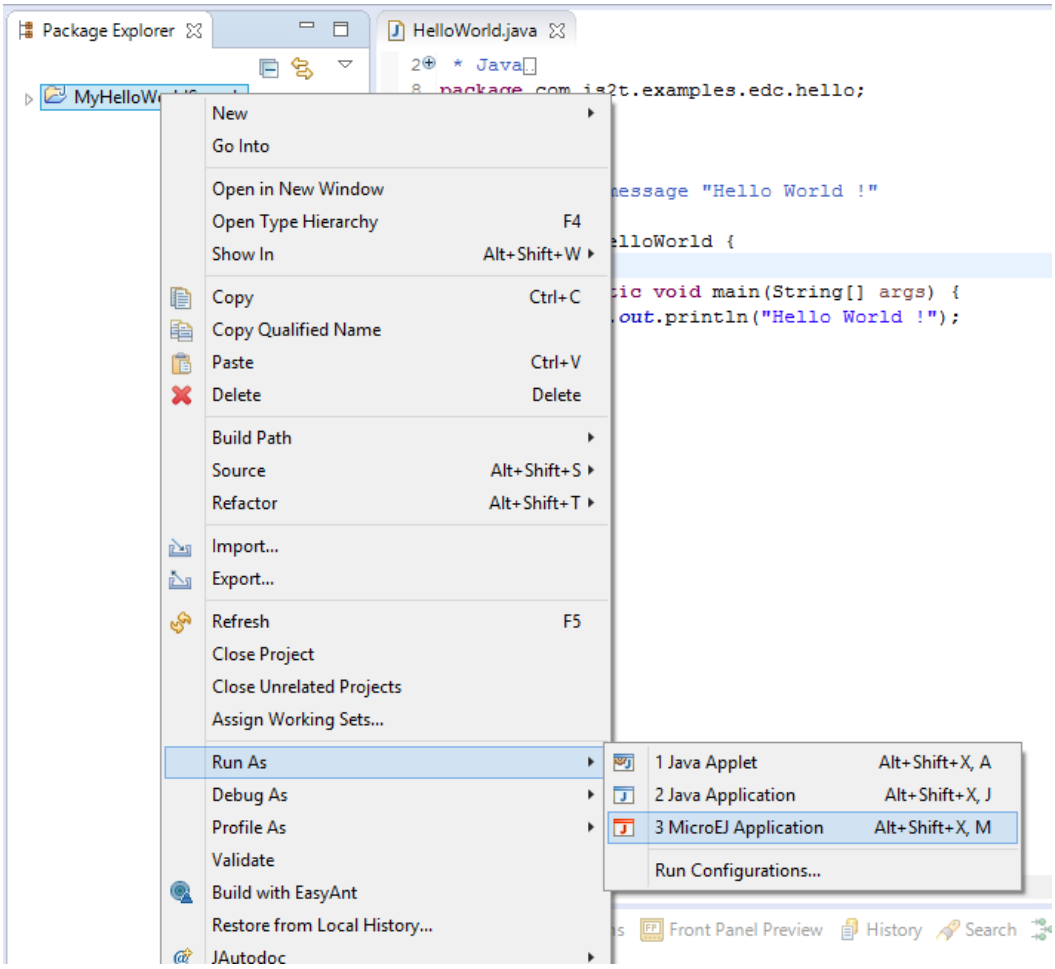


- Click on Finish. The selected example is imported into a project with the given name. The main class (the class which contains the `main()` method) is automatically opened.

2.2.2. 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



The application starts. It is executed on the MicroEJ simulator of the selected MicroEJ platform (GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi). The result of the test is printed in the console:

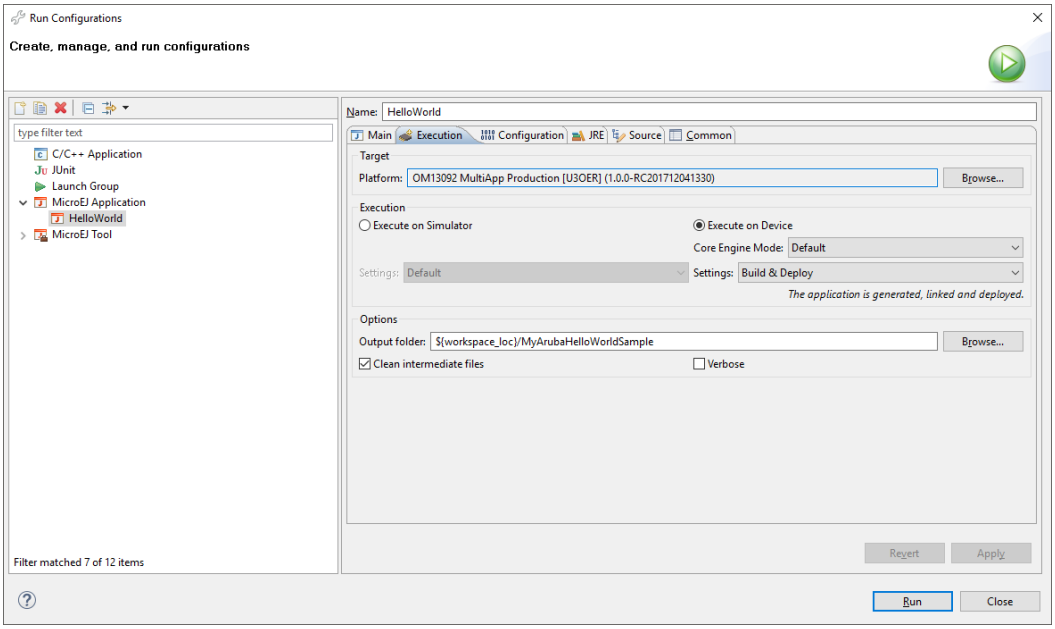
```
Hello World !
```

2.3. Run the Example on the GRPEACH 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



- 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`).
- 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 Renesas e2 studio 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 GRPEACH board.

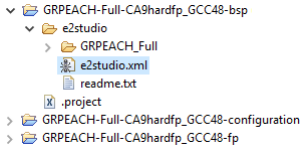
Note

This final step uses Renesas e2 studio.

The following steps are performed within MicroEJ.

- In MicroEJ SDK, expand the project GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi-bsp and the folder `e2studio/`. A Ant script (`e2studio.xml`) is available.

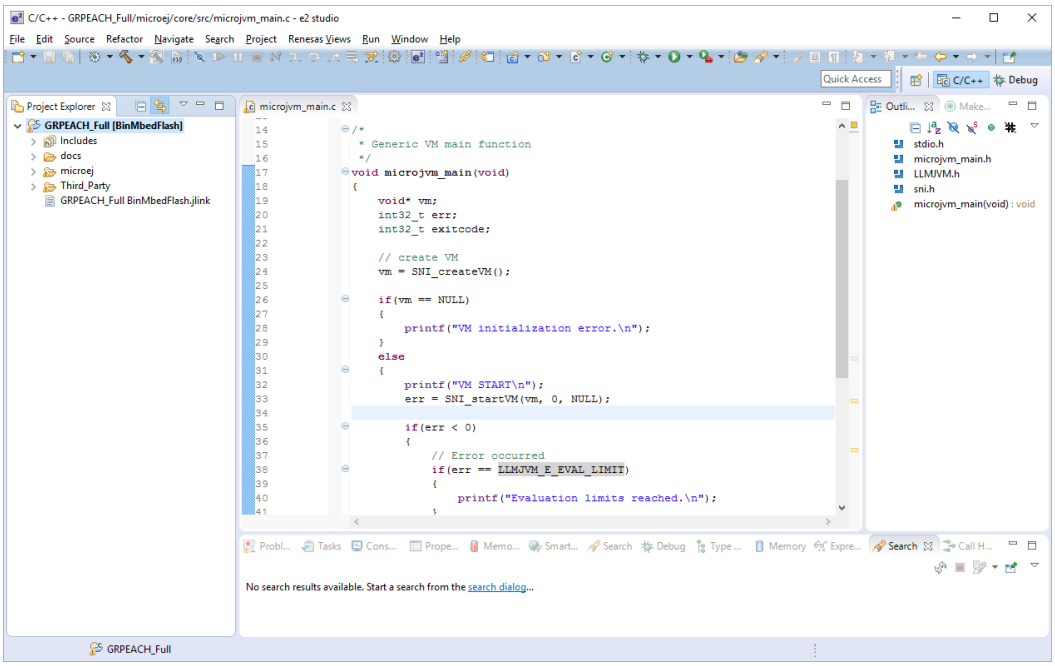
Figure 2.8. Renesas e2 studio Project Selection



- Right-click on the file and select `Run as... > Ant build` to launch Renesas e2 studio. If Renesas e2 studio is not found when the script is launched, use Ant Global Properties in `Window > Preferences > Ant > Runtime > Properties` to specify a new property named "e2studio.exe" with your e2studio.exe file location and retry.

The following steps are performed within Renesas e2 studio.

- Figure 2.9. Renesas e2 studio IDE



Build the Renesas e2 studio project by clicking on the menu `Project > Build Project`. The project is compiled and linked.

- Deploy the link result on the GRPEACH board by clicking on the menu `Run > Debug Configurations... > Renesas GDB Hardware Debug > GRPEACH_Full Bin-MbedFlash`. By default, project is configured for programming with SEGGER JLink probe. 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 GRPEACH is based on board support package provided by mbed. It includes mbed-OS, a TCP/IP network connection, a SSL client stack and a file system on microSD card. MicroEJ platform has been built with e2 studio against the ARM Embedded toolchain version 4.8.4_20140725 and mbed OS v5.1.

3.2. MicroEJ Platform Configuration

MicroEJ platform is based on MicroEJ architecture for ARM Cortex-A9.

Table 3.1. MCU Technical Specifications

MCU architecture	Cortex-A9 (Renesas R7S721001)
MCU Clock speed	400 MHz
Internal RAM	10 MB
External Flash	8 MB (NOR type on SPI bus)

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	6.17.2
UI	9.3.1
Network	6.1.5
File System	3.0.0
HAL	1.0.4

3.3. Platform Output stream

MicroEJ platform uses USB Virtual COM port as output print stream. The virtual COM port is available on USB connector for power and probe and is connected to the MCU USART 2.

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 Mbed-OS 5.1. RTOS objects are statically allocate: tasks stacks, task monitors, semaphores etc. The following table illustrates the available tasks and their stack size.

Table 3.3. Mbed-OS Tasks

Task name	Size	Priority
Network receive task	1 KB	osPriorityRealtime
File System	2 KB	osPriorityAboveNormal
NET	2 KB	osPriorityAboveNormal
Core Engine	12 KB	osPriorityNormal
TCP-IP	2 KB	osPriorityBelowNormal
DHCP	2 KB	osPriorityBelowNormal
PHY task	1 KB	osPriorityBelowNormal

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: 10 MB

Section Name	Size
MicroEJ standalone application heaps	1 MB [a]
MicroEJ standalone application stack blocks	512 * *n* bytes [b]
MicroEJ platform internal heap	*n* bytes [c]
SSL buffers	128 KB

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

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

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

Table 3.5. External flash: Program Flash (8 MB)

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

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

3.6. Graphical User Interface

This MicroEJ plaform features a reduced user interface. It includes a three-color LEDs (handled as 3 different LED), a user LED and a user buttons.

3.6.1. LEDs

The number of available LEDs on the platform is 4 (Value returned by Leds.getNumberOfLeds()). The LED ID values passed as parameters for LED functions are :

- 0 : User LED
- 1 : 3 colors LED Red
- 2 : 3 colors LED Green
- 3 : 3 colors LED Blue

User LED: The user LED output can be modified using Leds.setLedOn or Leds.setLedOff. The function Leds.setLedIntensity as no effect on this LED.

Implementation Note

The output pin connected to the LED is driven low or high depending on the requested state.

3 color LED: Each LED output can be modified using Leds.setLedIntensity to set the color intensity. (Leds.setLedOn is similar to Leds.setLedIntensity with intensity to 0 and Leds.setLedOff is similar to Leds.setLedIntensity with maximum intensity.)

Implementation Note

To change the intensity of each LED, a mbed library for software PWM is used. This uses software timers to drive the GPIO pins in order to generate the desired PWM value output on each LEDs.

3.6.2. Inputs

User buttons: The user buttons state are transmitted to Java application as button events (pressed and released).

Implementation Note

The user buttons events treatments are performed under interrupt.

3.7. Network

MicroEJ plaform features a network interface. Sockets are limited to 10. A DHCP client can be activated to retrieve an IP address.

Implementation Note

MicroEJ platform uses LwIP v2.0.0 fetched from git repository of the project. This implementation needs a 50 KB internal heap to work. The TCP MSS is 1460 bytes.

The network port uses a BSD (Berkley Software Distribution) API with select feature. A mechanism named dispatch event, with a dedicated task, is used to request non blocking operations and waits for completion or timeout.

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

3.7.1. MAC address customization

The MAC address used is by priority:

- The MAC address provided in the MicroEJ application launcher
- The device MAC address stored at the end of the Flash
- The default MAC address provided in mbed BSP

If no MAC address is supplied by the user in the MicroEJ application launcher, the platform uses a unique ID to generate a MAC address. In order to provide a unique default MAC address for each device, the platform is able to retrieve a custom MAC address using the UID customization written to the serial Flash memory.

The GRPEACH hardware does not provide a unique ID mechanism that can be used to generate a unique ID for each device. The last 20 bytes of the serial Flash are used to store a board defined UID with a summing checksum. If no UID is written to flash memory or if the checksum check fail, the platform automatically generate a new UID in flash memory using analogic reading on pins A0 to A3.

This UID could be defined and written manually to the Flash address 0x187FFFEC.

- Addresses 0x187FFFEC to 0x187FFFFD : 18 bytes of the UID (MSB to LSB)
- Addresses 0x187FFFFE to 0x187FFFFF : 2 bytes checksum (addition off all UID bytes)

UID programming can be done with a JLink probe by modifying and running the batch script on windows `GRPEACH-MyPlatform-GNUv48_cortexa9hf_launchpad_arm-none-eabi-configuration/tools/JLinkWriteUID/Windows`

3.8. SSL

MicroEJ plaform features a network secure interface. Available secured protocols are SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2. Keys and certificates supported formats are PEM and DER. The SSL memory pool allows up to 10 secure socket opened at the same time.

Implementation Note

MicroEJ platform uses mbedtls version 2.4.0. It uses a heap of 512 KB to store certificates and mbedtls contexts.

3.9. File System

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

Implementation Note

MicroEJ platform uses mbed library that relies on FatFs R0.11a library.

3.10. Serial Communications

3.10.1. UART Connector

MicroEJ platform provides one serial connection (ECOM COMM) on UART0 port. UART0 pins are (RTS/CTS mode is not used):

- TX: P2_14; available on connector CN14 D1
- RX: P2_15; available on connector CN14 D0

Implementation Note

This implementation uses interrupts and relies on the MicroEJ `LLCOMM_BUFFERED_CONNECTION` API. This API is FIFO oriented. It requires two distincts software buffers for reception and transmission: reception buffer uses 1024 bytes and transmission buffer uses 5 bytes. These buffers are statically allocated in internal RAM.

3.11. HAL

MicroEJ platform provides several GPIOs programmable via the HAL foundation library. All GPIOs are available on ARDUINO connectors (CN9, CN10, CN14 and CN15). Digital pins are implemented by a GPIO access.

Analog input pins (ADC) are driven by ADC channels.

Each GPIO port / pin value is accessible using either:

- The global MCU designation: all pins of all ports are grouped under only one virtual port (port 0) and have consecutive values: P1_0 has the ID 0, P1_1, the ID 1, P1_15 the ID 15, P2_0 the ID 16 and so on. For instance pin *P5_7* is accessible by `(0, 71)`. This designation is useful to target all MCU pins using only one virtual port.
- The standard MCU designation: Port1 has the ID 1, Port2 the ID 2 etc. Each pin of each port is a value between 0 (Port*N*-0) to 15 (Port*N*-15). For instance pin *P5_7* is accessible by `(5, 7)`. This designation is useful to target a specific MCU pin.
- The virtual board connectors designation. Board has 2 virtual connectors: ARDUINO digital port and ARDUINO analog port, with respectively these IDs 30 and 31. For instance pin *P1_15* is accessible on connector ARDUINO analog, pin P1_15: `(31, 5)`. This designation is useful to target a virtual connector pin without knowing which MCU pin it is and on which physical connector pin is connected.
- The physical board connectors designation. Board has 3 connectors: CN9, CN14 and CN15 (CN10 is not connected to the MCU), with respectively these IDs: 69, 74 and 75. For instance pin *P1_15* is accessible on connector CN15, pin6: `(75, 6)`. This designation is useful to target a physical connector pin without knowing which MCU pin it is.

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

Table 3.6. HAL GPIOs Ports and Pins

Port name	HAL port ID	Pins range
Global MCU virtual port	0	0 to 324
MCU port 1	1	0 to 15
MCU port 2	2	0 to 15
MCU port 3	3	0 to 15
MCU port 4	4	0 to 15
MCU port 5	5	0 to 11
MCU port 7	7	0 to 15
MCU port 8	8	0 to 15
MCU port 9	9	0 to 7
MCU port 10	10	0 to 15
MCU port 11	11	0 to 15
Board virtual port "ARDUINO digital"	30	0 to 15
Board virtual port "ARDUINO analog"	31	0 to 5
Board physical port "CN9"	69	1 to 10

Port name	HAL port ID	Pins range
Board physical port "CN14"	74	1 to 8
Board physical port "CN15"	75	1 to 6

The following table shows the exhaustive list of GPIOs connected to the HAL library, their IDs according the ports IDs and pins IDs (see before):

Table 3.7. HAL GPIOs Pins Designation Mapping

Port / Pin	MCU virtual port (1)	MCU port (2)	Board virtual port (3)	Board physical port (4)
P1_2	0, 3	1, 2	31, 15	69, 1
P1_3	0, 4	1, 3	31, 14	69, 2
P1_8	0, 9	1, 8	30, 0	75, 1
P1_9	0, 10	1, 9	30, 1	75, 2
P1_10	0, 11	1, 10	30, 2	75, 3
P1_11	0, 12	1, 11	30, 3	75, 4
P1_13	0, 14	1, 13	30, 4	75, 5
P1_14	0, 15	1, 14	30, 5	75, 6
P2_14	0, 31	2, 14	31, 1	74, 7
P2_15	0, 32	2, 15	31, 0	74, 8
P4_4	0, 53	4, 4	31, 5	74, 3
P4_5	0, 54	4, 5	31, 4	74, 4
P4_6	0, 55	4, 6	31, 3	74, 5
P4_7	0, 56	4, 7	31, 2	74, 6
P8_11	0, 119	8, 11	31, 7	74, 1
P8_13	0, 121	8, 13	31, 6	74, 2
P8_14	0, 122	8, 14	31, 9	69, 9
P8_15	0, 123	8, 15	31, 8	69, 10
P10_12	0, 144	10, 12	31, 13	69, 5
P10_13	0, 165	10, 13	31, 10	69, 8
P10_14	0, 146	10, 14	31, 11	69, 7
P10_15	0, 147	10, 15	31, 12	69, 6

The following table lists the hardware analog devices (ADC channel) used by HAL analog pins:

Table 3.8. HAL Analog IOs Pins Designation Mapping

Port / Pin	ADC channel	PWM / channel
P1_8	0	-

Port / Pin	ADC channel	PWM / channel
P1_9	1	-
P1_10	2	-
P1_11	3	-
P1_12	4	-
P1_13	5	-
P1_14	6	-
P1_15	7	-
P4_4	-	PWM2E
P4_5	-	PWM2F
P4_6	-	PWM2G
P4_7	-	PWM2H
P8_11	-	PWM1D
P8_13	-	PWM1F
P8_14	-	PWM1H
P8_15	-	PWM1G

Chapter 4. Board Configuration

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

4.1. Mandatory Connectors

GRPEACH provides three connectors used as:

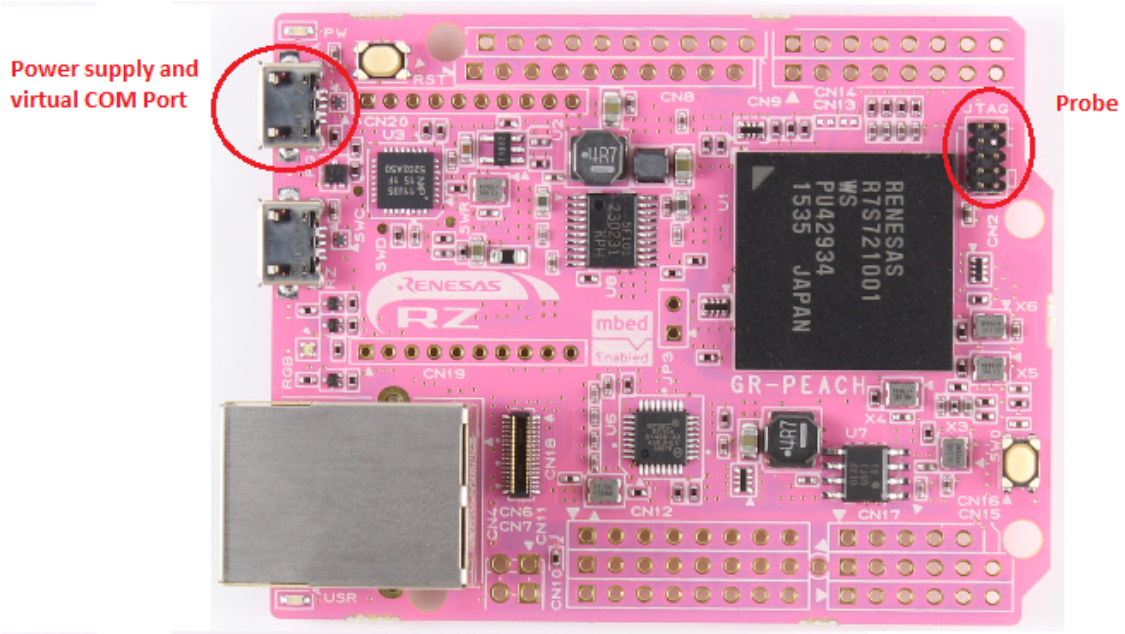
- Power supply connector and Virtual COM port
- Probe connector

Plug a micro-B USB cable to a computer to power the board and be able to see the MicroEJ standalone application `System.out.print` traces. The virtual port configuration is 115200 baud 8N1.

Prerequisites

Download mbed Windows serial port driver from `https://developper.mbed.org/handbook/Windows-serial-configuration/` and install it on your machine.

Figure 4.1. Mandatory Connectors

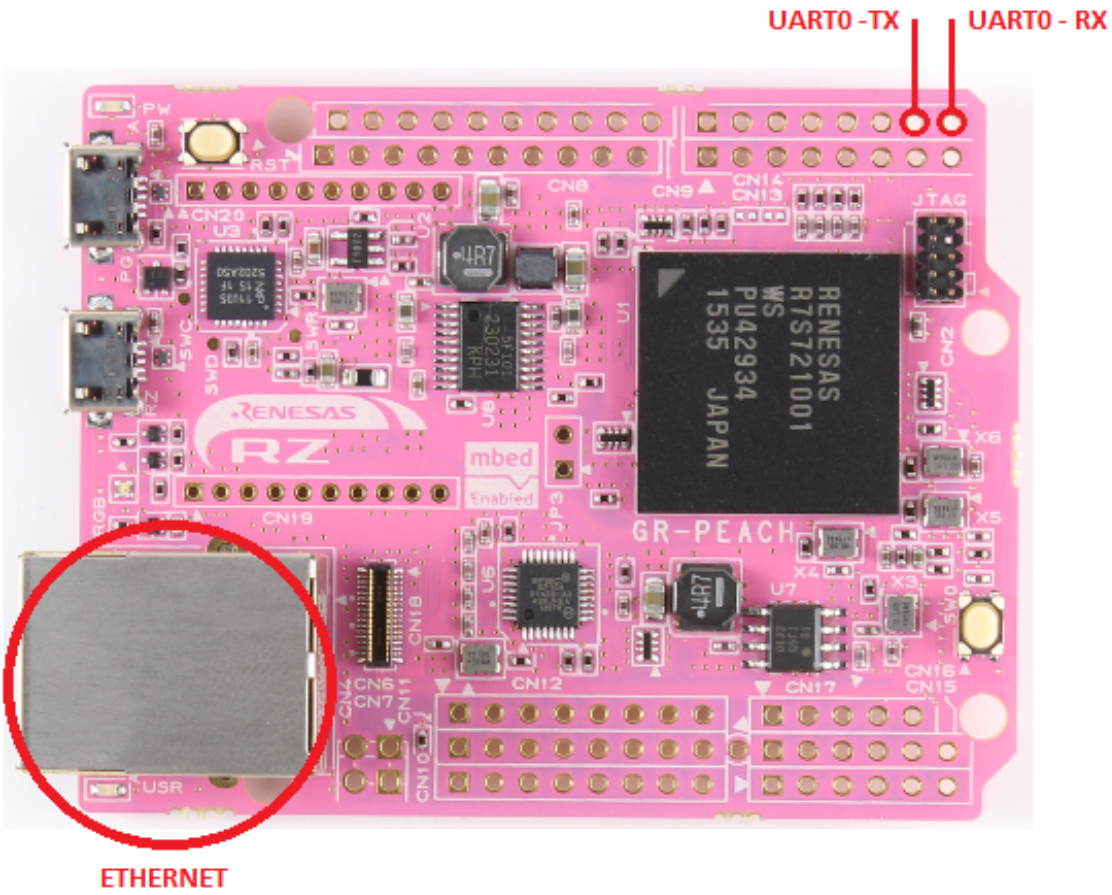


4.2. Communication Connectors

GRPEACH provides several communication ports:

- Ethernet
- Serial communication

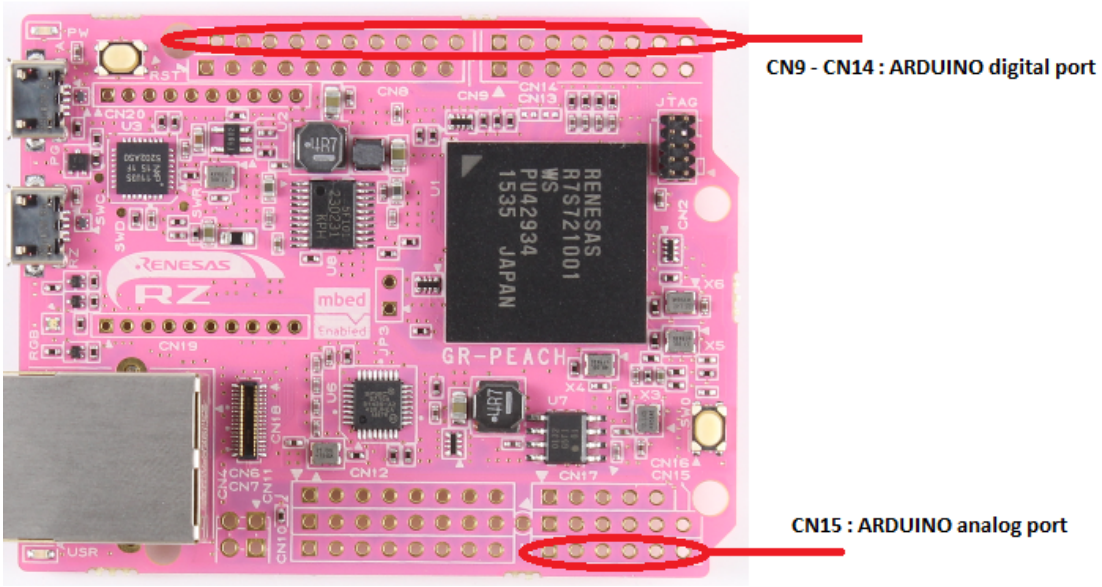
Figure 4.2. Communication Connectors



4.3. HAL Connectors

GRPEACH provides several HAL GPIOs on connector CN9, CN14 and CN15.

Figure 4.3. HAL Connectors



Chapter 5. Renesas e2 studio

5.1. Install Renesas e2 studio

This section describes how to install e2 studio from Renesas.

5.1.1. Download and install Renesas e2 studio

- Go to `https://www.renesas.com/en-us/products/software-tools/tools/ide/e2studio.html#`.
- In the `Downloads` tab, select the desired version. You will be redirected to a page describing the tool.
- Download the executable file (e.g. `setup_e2_studio_5_0_0_043.exe`) at the bottom of the page.
- Run executable file and follow the installation steps. Install additional software and drivers if proposed. A new application named `e2 studio` shall have been installed.

5.1.2. Download and install GNU Toolchain

- Download the version GNU ARM Embedded Toolchain 4.8.4_20140725.
- Run executable file and follow the installation steps.
- During the next startup of e2studio, the installed GNU toolchain will be automatically detected. If not, you should add it manualy in the preference menu : `Windows > Preferences > C/C++ > Renesas > Renesas Toolchain Management`

5.2. BSP Project Structure

The e2 studio BSP project folder is included in a MicroEJ standard project. This project is visible from the MicroEJ workspace and uses the same tree as the computer file system:

- e2studio: the MicroEJ platform project itself

The e2 studio BSP project file is `e2studio/GRPEACH_Full/.project`. This e2 studio BSP project has been written for e2 studio v5.1.0.022. The project has the following file structure:

- microej/*: all MicroEJ platform implementation files
- Third_Party/*: all the mbed, mbedtls and lwip related files

The MicroEJ platform implementation files are grouped by MicroEJ features:

- microej/core/inc: Core Engine implementation over mbed-OS include files (always required)
- microej/core/src: Core Engine implementation over mbed-OS source files (always required)
- microej/fs/inc: File system implementation over mbed-OS and FatFs library include files
- microej/fs/src: File system implementation over mbed-OS and FatFs library source files
- microej/net/inc: Network implementation over mbed-OS and lwIP include files
- microej/net/src: Network implementation over mbed-OS and lwIP source files
- microej/ssl/inc: SSL implementation over mbedTLS include files
- microej/ssl/src: SSL implementation over mbedTLS source files
- microej/ui/inc: UI implementation over mbed-OS include files
- microej/ui/src: UI implementation over mbed-OS source files
- microej/mbed_adapt: mbed files that have been modified for MicroEJ platform implementation

The Third_Party folder is divided by library sources:

- Third_Party/LwIP: lwIP library souces and include file
- Third_Party/mbed: mbed souces and include file with specific files for target RZ_A1H
- Third_Party/mbed-lib: mbed libraries souces and include files for Ethernet interface, SD file system and software PWM. Theses folders also include specific files for target RZ_A1H
- Third_Party/mbed-rtos: mbed-OS library souces and include files with specific files for target RZ_A1H
- Third_Party/mbedtls: mbedTLS library souces and include file

Chapter 6. Changelog

6.1. Version 1.5.1

- Migrated to Architecture 6.17.2.
- Use pack UI 9.3.1 instead of 9.0.2.

6.2. Version 1.4.3

- BugFix : Use pack Net 6.1.5 instead of 6.1.4

6.3. Version 1.4.2

- BugFix : Soft DNS resolver doesn't retry on timeout and doesn't try other servers on errors

6.4. Version 1.4.1

- Increase SSL memory pool

6.5. Version 1.4.0

- Upgrade platform architecture

6.6. Version 1.3.0

- Upgrade platform architecture

6.7. Version 1.2.0

- Add HAL module low level interface implementation
- BugFix : Error on SSL certificate check.

6.8. Version 1.1.1

- BugFix : Initialization of serial handler should be done before enable IRQ.

6.9. Version 1.1.0

- Fixed space in MicroEJ repository path causing link errors

- Change address MAC customization in flash memory by UID customization
- Change output trace baudrate to 115200

6.10. Version 1.0.0

Initial release of the platform.