

MicroEJ Platform Reference Implementation

Developer's Guide



STM32F746GDISCO 3.1.5

Reference:	TLT-0789-DGI-PlatformReferenceImplementation-STM32F746GDISCO
Version:	3.1.5
Revision:	3.1.5

Confidentiality & Intellectual Property

All rights reserved. Information, technical data and tutorials contained in this document are confidential and proprietary under copyright Law of Industrial Smart Software Technology (IS2T S.A.) operating under the brand name MicroEJ®. Without written permission from IS2T S.A., *copying or sending parts of the document or the entire document by any means to third parties is not permitted*. Granted authorizations for using parts of the document or the entire document do not mean IS2T S.A. gives public full access rights.

The information contained herein is not warranted to be error-free. IS2T® and MicroEJ® and all relative logos are trademarks or registered trademarks of IS2T S.A. in France and other Countries.

Java™ is Sun Microsystems' trademark for a technology for developing application software and deploying it in cross-platform, networked environments. When it is used in this documentation without adding the ™ symbol, it includes implementations of the technology by companies other than Sun.

Java™, all Java-based marks and all related logos are trademarks or registered trademarks of Sun Microsystems Inc, in the United States and other Countries.

Other trademarks are proprietary of their authors.

Revision History		
Revision 3.1.5	June 8th 2017	
BugFix: Soft DNS resolver doesn't retry on timeout and doesn't try other servers on errors		
Revision 3.1.4	June 8th 2017	
MicroEJ 4.1 alignment		
Revision 2.3.0	December 27th 2016	
Fixed typo in Ethernet buffers size		
Revision 2.2.0	June 4th 2016	
First release		

Table of Contents

1. Introduction	1
1.1. Intended Audience	1
1.2. Scope	1
1.3. Prerequisites	1
2. Create and Use Your First MicroEJ Platform	2
2.1. Create a MicroEJ Platform	2
2.2. Run an Example on the MicroEJ Simulator	4
2.2.1. Create Example	4
2.2.2. Run Example	5
2.3. Run the Example on the STM32F746G-DISCO Board	6
2.3.1. Compile MicroEJ Standalone Application	6
2.3.2. Link and Deploy MicroEJ Standalone Application	7
3. Specification	10
3.1. Overview	10
3.2. MicroEJ Platform Configuration	10
3.3. Platform Output stream	10
3.4. RTOS Configuration	11
3.5. Memories	11
3.6. Graphical User Interface	12
3.6.1. Display	13
3.6.2. Inputs	13
3.7. Network	14
3.8. SSL	14
3.9. File System	14
3.10. Serial Communications	15
3.10.1. UART Connector	15
3.10.2. USB Connector	15
3.11. HAL	15
4. Board Configuration	18
4.1. Mandatory Connectors	18
4.2. Communication Connectors	18
4.3. HAL Connectors	19
5. Keil MDK-ARM Configuration	20
5.1. Install Keil MDK-ARM	20
5.1.1. Download Keil MDK-ARM	20
5.1.2. Activate the Evaluation License	20
5.1.3. Install Microcontroller Specific Pack	22
5.2. BSP Project Structure	23
6. Changelog	25
6.1. Version 3.1.5	25
6.2. Version 3.1.4	25
6.3. Version 2.2.0	25

List of Figures

2.1. MicroEJ Platform Reference Implementation Selection	2
2.2. New MicroEJ Platform Naming	3
2.3. MicroEJ Platform Build	4
2.4. MicroEJ Standalone Application Selection	5
2.5. MicroEJ Standalone Application Naming	5
2.6. MicroEJ Standalone Application Running	6
2.7. Execution on Device	7
2.8. Keil uVision Project Selection	8
2.9. Keil uVision IDE	8
4.1. Mandatory Connectors	18
4.2. Communication Connectors	19
4.3. HAL Connectors	19
5.1. Keil MDK-ARM Computer ID	21
5.2. Keil MDK-ARM License Installation	22
5.3. Keil MDK-ARM Microcontroller Pack	22
5.4. Microcontroller Pack Update	23
5.5. Microcontroller Pack Selection	23

List of Tables

3.1. MCU Technical Specifications	10
3.2. MicroEJ Configuration	10
3.3. FreeRTOS Tasks	11
3.4. Internal RAM: DTCM (64 KB)	12
3.5. Internal RAM: SRAM1 (240 KB)	12
3.6. External RAM: SDRAM (8 MB)	12
3.7. Internal flash: AXIM interface (1 MB)	12
3.8. External flash: QSPI (16 MB)	12
3.9. HAL GPIOs Ports and Pins	16
3.10. HAL GPIOs Pins Designation Mapping	17
3.11. HAL Analog IOs Pins Designation Mapping	17

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 ST STM32F746G-DISCO 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 STM32F746G-DISCO 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 STM32F746G-DISCO board.
- Run a MicroEJ standalone application on the MicroEJ simulator.
- Run a MicroEJ standalone application on the MicroEJ platform and deploy it on the STM32F746G-DISCO board.

1.3. Prerequisites

- PC with Windows 7 or later.
- The MicroEJ SDK environment must be installed.
- STM32F746G-DISCO board.
- The STM32 ST-LINK utility (minimal version 3.7.1).
- Keil MDK-ARM μ Vision 5.18.0.0 or higher. The Keil μ Vision evaluation version is 32KB code size limited. To get a Keil μ Vision evaluation license for MicroEJ SDK, please consult the chapter “Install Keil MDK-ARM”.

Chapter 2. Create and Use Your First MicroEJ Platform

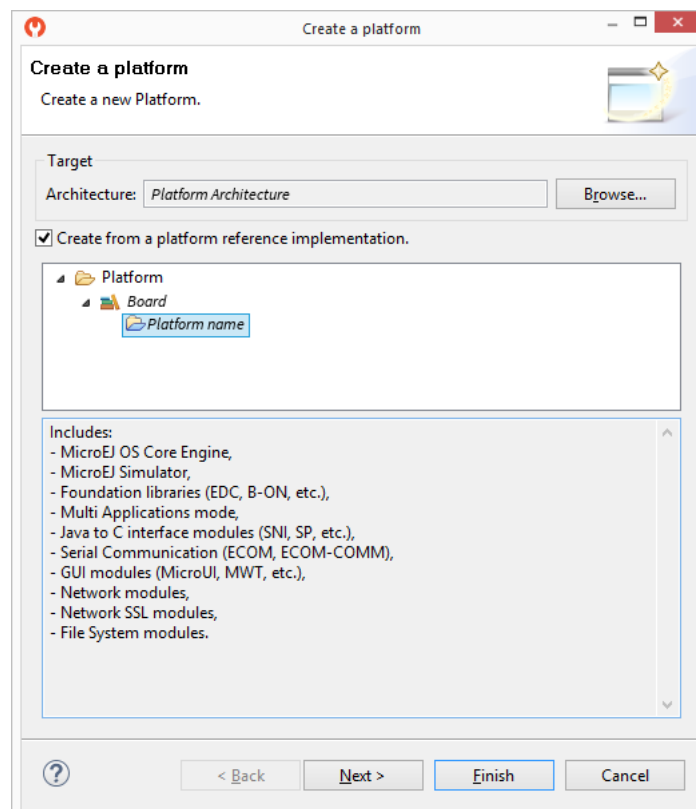
2.1. Create a MicroEJ Platform

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 STM32F746G-DISCO.

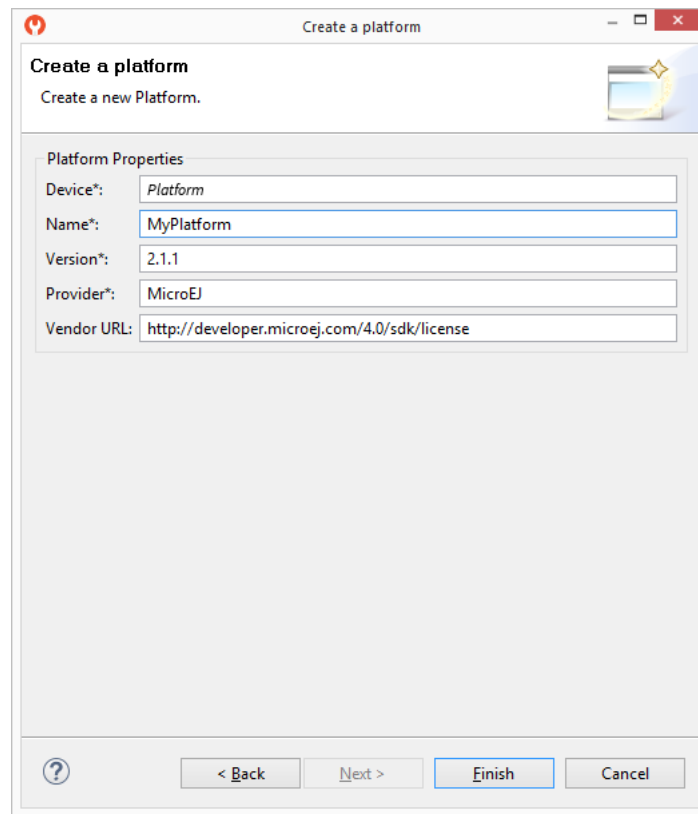
- Open MicroEJ SDK.
- Open the MicroEJ platform wizard: `File > New > Platform`.
- Select the MicroEJ architecture ARM Cortex-M7 ARMCC from the combo box. A MicroEJ Platform Reference Implementation is available:

Figure 2.1. MicroEJ Platform Reference Implementation Selection



- Select the MicroEJ platform SingleApp for the STM32F746G-DISCO 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:
 - STM32F746GDISCO-MyPlatform-CM7hardfp_ARMCC5-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).
 - STM32F746GDISCO-MyPlatform-CM7hardfp_ARMCC5-bsp: Contains a ready-to-use BSP software project for the STM32F746G-DISCO board, including a Keil uVision project, an implementation of MicroEJ core engine (and extensions) port on FreeRTOS RTOS and the STM32F746G-DISCO board support package.
 - STM32F746GDISCO-MyPlatform-CM7hardfp_ARMCC5-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 screenshot shows the 'Overview' window of the MicroEJ Platform Build tool. It is divided into four main sections: 'Platform Properties', 'Platform Content', 'Platform Configuration', and 'Build'.
1. **Platform Properties**: Contains fields for 'Device' (Board), 'Name' (MyPlatform), 'Version' (2.1.1), 'Provider' (MicroEJ), and 'Vendor URL' (http://developer.microej.com/4.0/sdk/license).
2. **Platform Content**: Explains that the platform content is composed of two parts: 'Environment' (select the architecture) and 'Modules' (select modules to import in the platform).
3. **Platform Configuration**: States that once the content is chosen, it can be configured. It includes a link to 'Configuration' and explains that each module can be configured by creating a folder with its name along the .platform file. It lists two options: an optional [module].properties file and optional module specific files and folders. It also notes that modifying these files requires rebuilding the platform.
4. **Build**: Instructs to 'Generate and test the platform.' It includes a 'Build Platform' button and states that 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 STM32F746G-DISCO board and is ready to be linked into the Keil uVision project. Its name is STM32F746GDISCO-My-Platform-CM7hardfp_ARMCC5.

The MicroEJ platform is now ready for use and available in the MicroEJ platforms list of your MicroEJ repository (Windows > Preferences > MicroEJ > Platforms in work-space).

2.2. Run an Example on the MicroEJ Simulator

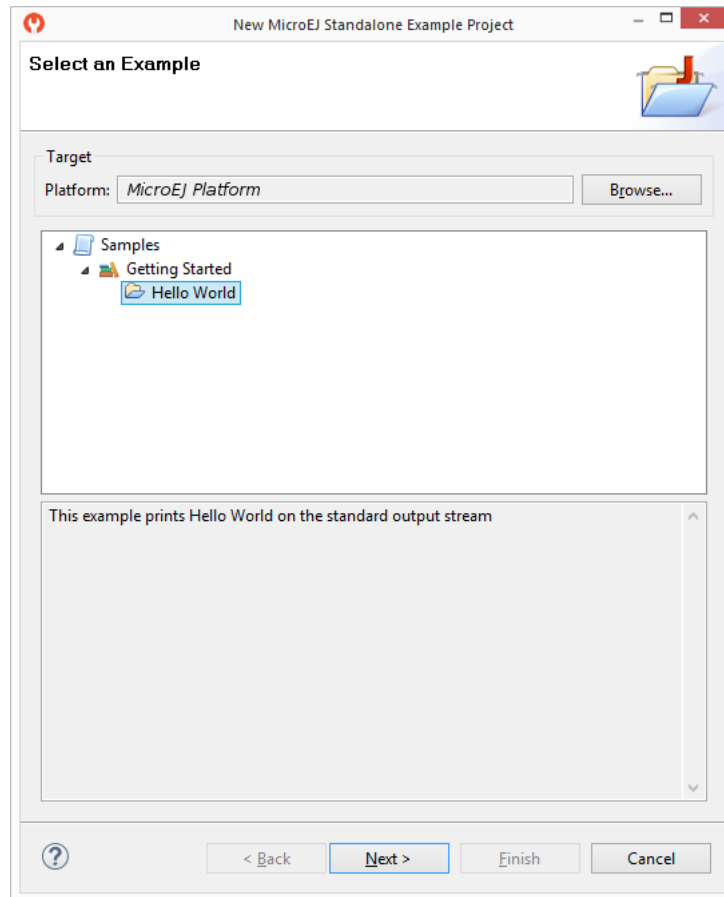
The aim of this chapter is to create a MicroEJ standalone application from a built-in example. Initially, this example will run on the MicroEJ simulator. Then, in the next section, this application will be compiled and deployed on the STM32F746G-DISCO board using the MicroEJ platform.

2.2.1. Create Example

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

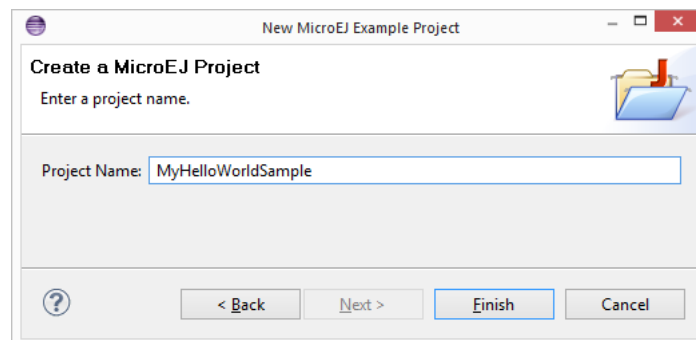
- Select the MicroEJ platform STM32F746GDISCO-MyPlatform-CM7hardfp_ARMCC5 from the combo box.
- Select the example Samples > Getting Started > Hello World.

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



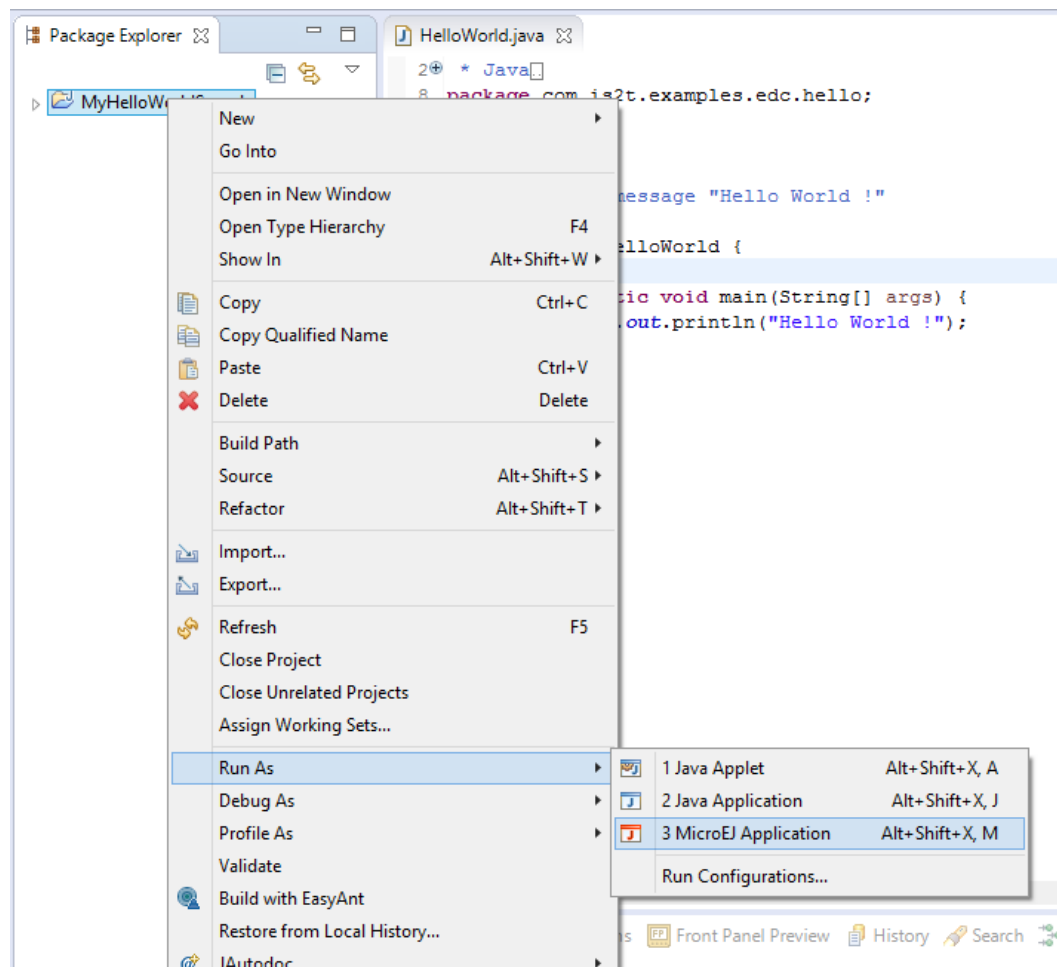
- Click on Finish. The selected example is imported into a project with the given name. The main class (the class which contains the `main()` method) is automatically opened.

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 (STM32F746GDISCO-MyPlatform-CM7hardfp_ARMCC5). The result of the test is printed in the console:

```
Hello World !
```

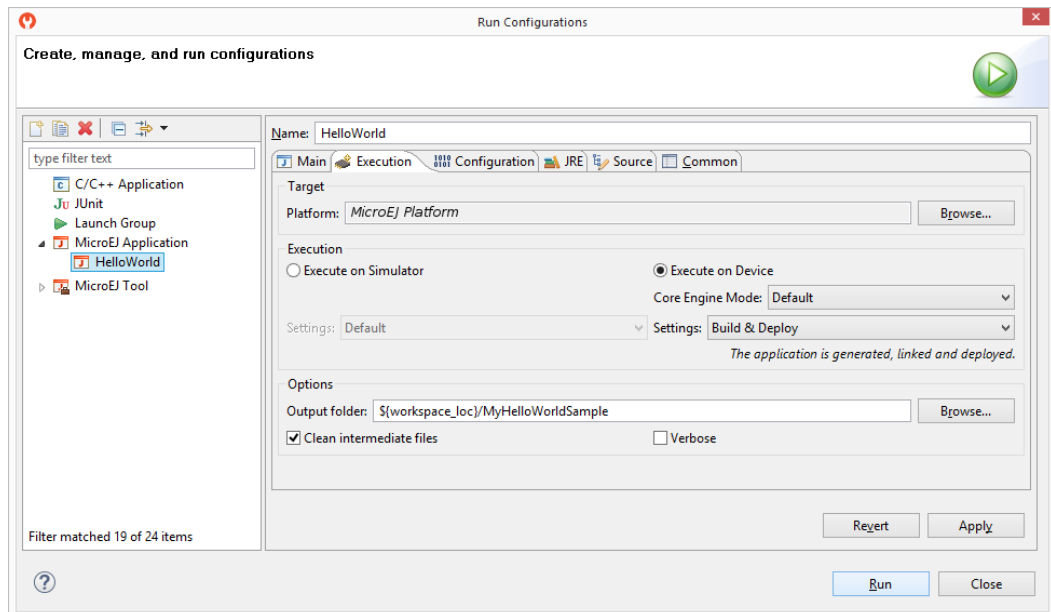
2.3. Run the Example on the STM32F746G-DISCO Board

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



- 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 Keil uVision 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 STM32F746G-DISCO board.



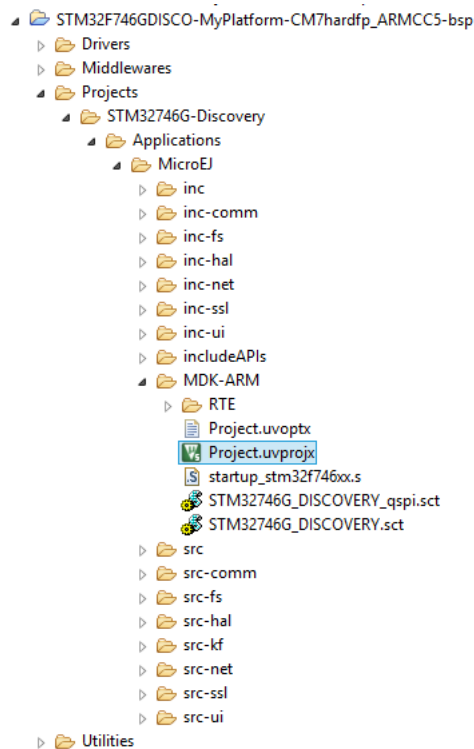
Note

This final step uses Keil uVision 5.18.0.0.

The following steps are performed within MicroEJ.

- In MicroEJ SDK, expand the project `STM32F746GDISCO-MyPlatform-CM7hardfp_ARMCC5-bsp` and the folder `Projects/STM32746G-Discovery/Applications/MicroEJ/MDK-ARM`. A Keil uVision project file (`Project.uvprojx`) is available.

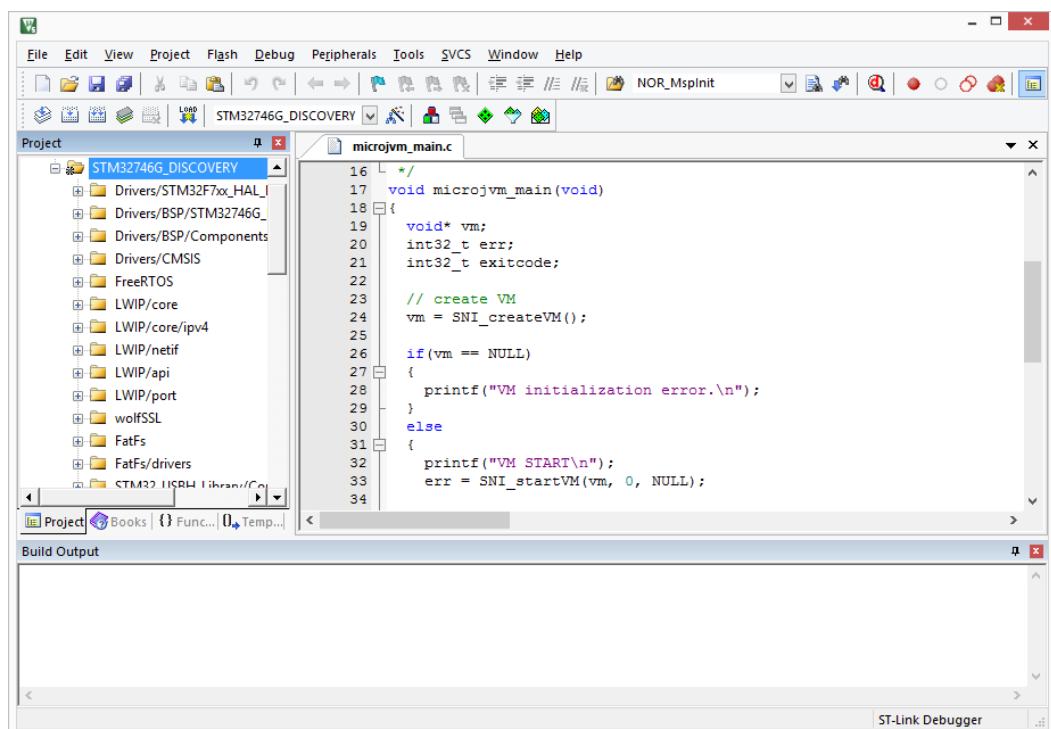
Figure 2.8. Keil uVision Project Selection



Double-click on this file to open Keil uVision.

The following steps are performed within Keil uVision.

- Figure 2.9. Keil uVision IDE



Build the Keil uVision 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 link result on the STM32F746G-DISCO board by clicking on the menu `Flash > Download`.

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 STM32F746G-DISCO is based on board support package provided by STMicroelectronics: STM32CubeF7. It includes FreeRTOS, a graphical user interface, a TCP/IP network connection, a file system on microSD card, a serial connection and some custom GPIOs. MicroEJ platform has been built against Keil μ Vision.

3.2. MicroEJ Platform Configuration

MicroEJ platform is based on MicroEJ architecture for ARM Cortex-M7.

Table 3.1. MCU Technical Specifications

MCU architecture	Cortex-M7 (STM32F746NG)
MCU Clock speed	200 MHz
Internal Flash	1 MB
Internal RAM	320 KB
External Flash	16 MB (QSPI)
External RAM	8 MB (SDRAM)

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	6.9.0
UI	9.0.2
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 ST-Link/V2 connector and is connected to the MCU USART 1.



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: 8
- Parity bits: None
- Stop bits: 1
- Flow control: None



Implementation Note

On the STM32F746G-DISCO, the following parameters can be adjusted:

- Baudrate
- Parity bits
- Stop bits

3.4. RTOS Configuration

MicroEJ platform uses FreeRTOS 8.2.1. 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
Core Engine	12 KB	11
Touch	512 bytes	12
Network Dispatch	2 KB	12
Network DHCP	512 bytes	8
Network Ethernet Link	512 bytes	9
Network Ethernet Input	350 bytes	14
LWIP TCP	1 KB	13
File System	2 KB	12
MCU Charge Calculation	512 bytes	15
Framerate Calculation	512 bytes	3

3.5. Memories

MicroEJ Platform 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: DTCM (64 KB)

Section Name	Size
Ethernet buffers	15560 B
MicroEJ standalone application stack blocks	512 * n bytes ^a
MicroEJ platform internal heap	n bytes ^b

^a n is the number of stack blocks defined in MicroEJ Application launcher options.

^b n depends on memory configuration set in MicroEJ Application launcher options.

Table 3.5. Internal RAM: SRAM1 (240 KB)

Section Name	Size
SSL buffers	65 KB
Any RW	n bytes ^a

^a n depends on MicroEJ application libraries used.

Table 3.6. External RAM: SDRAM (8 MB)

Section Name	Size
Display buffers	510 KB
MicroUI working buffer	3 MB
Multi applications working buffer	3 MB
MicroEJ standalone application heaps	1536 KB ^a

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

Table 3.7. Internal flash: AXIM interface (1 MB)

Section Name	Size
Any RO	n bytes ^a

^a n depends on MicroEJ application, MicroEJ libraries, Board support package, RTOS, drivers, etc.

Table 3.8. External flash: QSPI (16 MB)

Section Name	Size
MicroEJ standalone application resources	n bytes ^a
Pre-installed MicroEJ sandboxed application	n bytes ^b

^a n is the size of all MicroEJ standalone application resources.

^b n depends on the size defined in MicroEJ Application launcher options.

3.6. Graphical User Interface

MicroEJ platform 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 TFT 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.



Implementation Note

The display stack implementation uses the double-buffering mode: the current MicroEJ application rendering is performed in a background buffer (called back buffer) and another buffer is used by the TFT display to refresh itself (called frame buffer). When the drawing is done, a copy of pixels data from the back buffer to the frame buffer is performed (stack copy). In order to avoid flickering, this copy is synchronized on display refresh tick.

Each buffer is allocated in external RAM (SDRAM). 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: 480
- `height` is the display width in pixels: 272
- `bpp` is the number of bits-per-pixel: 16

The buffers size is $2 * 262120 = 510$ KB.

The display module uses the STM32F7 hardware acceleration to perform some drawings: the ChromArt (DMA2D). The DMA2D renders all *fill rectangles* (`GraphicsContext.fillRect()`) and performs the drawings of all images.

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 SDRAM and the reserved size is 3 MB.



Implementation Note

This buffer is called "working buffer". An image buffer size follows the same rule as the LCD buffer (see before).

3.6.2. Inputs

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



Implementation Note

A touch *press* event is detected under interrupt. It wakes up a dedicated OS task. This task is used to communicate with the touch (I2C communication). For all next *drag* events, the

touch task runs in polling mode. When a *release* is detected, the touch task goes to sleep and waiting a touch interrupt.

User button: The user button is reserved to the multi applications feature: it allows to force to kill a sandboxed application.



Implementation Note

The user button event treatment is performed under interrupt.

3.7. Network

MicroEJ plaform features a network interface. Socket limits are

- 10 sockets for TCP client connections
- 5 sockets for TCP server connections
- 6 sockets for UDP connections

A DHCP client can be activated to retrieve an IP address.



Implementation Note

MicroEJ platform uses LwIP v1.5.0 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.

3.8. 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.



Implementation Note

MicroEJ platform uses WolfSSL. WolfSSL uses a heap of 65 KB to store certificates.

3.9. File System

MicroEJ plaform features a file system interface. A microSD 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 FatFS R0.11. The FAT FS driver is the SD driver port of STM32Cube v1.3.0.

3.10. Serial Communications

3.10.1. UART Connector

MicroEJ platform provides one serial connection (ECOM COMM) on UART6 port. UART6 pins are (RTS/CTS mode is not used):

- TX: PC6; available on connector CN4 D1
- RX: PC7; available on connector CN4 D0



Implementation Note

This implementation uses interrupts and relies on the MicroEJ `LLCOMM_BUFFERED_CONNECTION` API. This API is FIFO oriented. It requires two distinct 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.

3.10.2. USB Connector

MicroEJ platform features a serial connection (ECOM COMM) on USB port USBH: USB CDC. USBH connector is CN12.



Implementation Note

This implementation uses interrupts and relies on the MicroEJ `LLCOMM_BUFFERED_CONNECTION` API. This API is FIFO oriented. It requires two distinct software buffers for reception and transmission: reception buffer uses 1024 bytes and transmission buffer uses 5 bytes. Additionally the USBH_CDC drivers requires two USB buffers of 64 bytes each. 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 (CN4 to CN7). Digital pins are implemented by a GPIO access.

Analog input pins (ADC) are driven by ADC channels of ADC 3 and analog output pins (DAC) drive PWM channels of timers 1, 3, 5 and 12.

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: PA0 has the ID 0, PA1, the ID 1, PA15 the ID 15, PB0 the ID 16 and so on. For instance pin *PF6* is accessible by (0 , 86). This designation is useful to target all MCU pins using only one virtual port.
- The standard MCU designation: PortA has the ID 1, PortB the ID 2 etc. Each pin of each port is a value between 0 (PortN-0) to 15 (PortN-15). For instance pin *PF6* is accessible by (6 , 6). 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 *PF6* is accessible on connector ARDUINO analog, pin A4: (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: CN4, CN5 and CN7 (CN6 is not connected to the MCU), with respectively these IDs: 64, 65 and 67. For instance pin *PF6* is accessible on connector CN5, pin5: (65 , 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 3.9. HAL GPIOs Ports and Pins

Port name	HAL port ID	Pins range
Global MCU virtual port	0	0 to 143
MCU port A	1	0 to 15
MCU port B	2	0 to 15
MCU port F	6	0 to 15
MCU port G	7	0 to 15
MCU port H	8	0 to 15
MCU port I	9	0 to 15
Board virtual port "ARDUINO digital"	30	0 to 15
Board virtual port "ARDUINO analog"	31	0 to 7
Board physical port "CN4"	64	1 to 8
Board physical port "CN5"	65	1 to 6
Board physical port "CN7"	67	1 to 10

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 3.10. HAL GPIOs Pins Designation Mapping

Port / Pin	MCU virtual port (1)	MCU port (2)	Board virtual port (3)	Board physical port (4)
PA0	0, 0	1, 0	31, 0	65, 1
PA8	0, 8	1, 8	30, 10	67, 3
PA15	0, 15	1, 15	30, 9	67, 2
PB4	0, 20	2, 4	30, 3	64, 4
PB14	0, 30	2, 14	30, 12	67, 5
PB15	0, 31	2, 15	30, 11	67, 4
PF6	0, 86	6, 6	31, 5	65, 6
PF7	0, 87	6, 7	31, 4	65, 5
PF8	0, 88	6, 8	31, 3	65, 4
PF9	0, 89	6, 9	31, 2	65, 3
PF10	0, 90	6, 10	31, 1	65, 2
PG6	0, 102	7, 6	30, 2	64, 3
PG7	0, 103	7, 7	30, 4	64, 5
PH6	0, 118	8, 6	30, 6	64, 7
PI0	0, 128	9, 0	30, 5	64, 6
PI1	0, 129	9, 1	30, 13	67, 6
PI2	0, 130	9, 2	30, 8	67, 1
PI3	0, 131	9, 3	30, 7	64, 8

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

Table 3.11. HAL Analog IOs Pins Designation Mapping

Port / Pin	ADC 3 channel	PWM timer / channel
PA0	0	-
PA8	-	1 / 1
PB4	-	3 / 1
PB15	-	12 / 2
PF6	4	-
PF7	5	-
PF8	6	-
PF9	7	-
PF10	8	-
PH6	-	12 / 1
PI0	-	5 / 4

Chapter 4. Board Configuration

STM32F746G-DISCO provides several connectors, each connector is used by the MicroEJ Core Engine itself or by a foundation library.

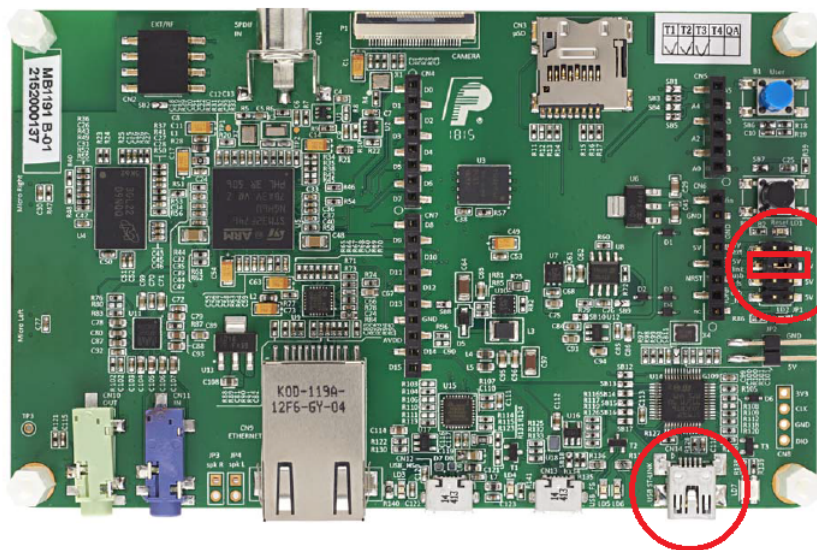
4.1. Mandatory Connectors

STM32F746G-DISCO provides a multi function USB port used as:

- Power supply connector
- Probe connector
- Virtual COM port

First of all, ensure the *Power Supply jumper* is fit to the second option: *5V Link*. 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 the MicroEJ standalone application `System.out.print` traces.

Figure 4.1. Mandatory Connectors



Power supply jumper :
select "5V link"

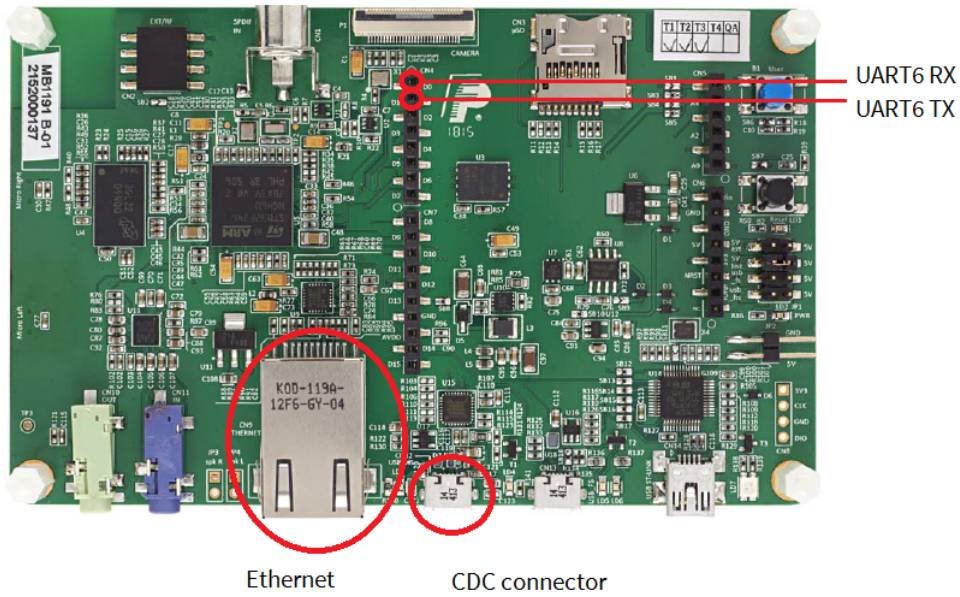
Power supply / ST-LINK/V2 / Virtual COM Port

4.2. Communication Connectors

STM32F746G-DISCO provides several communication ports:

- Ethernet
- Serial communication
- CDC communication

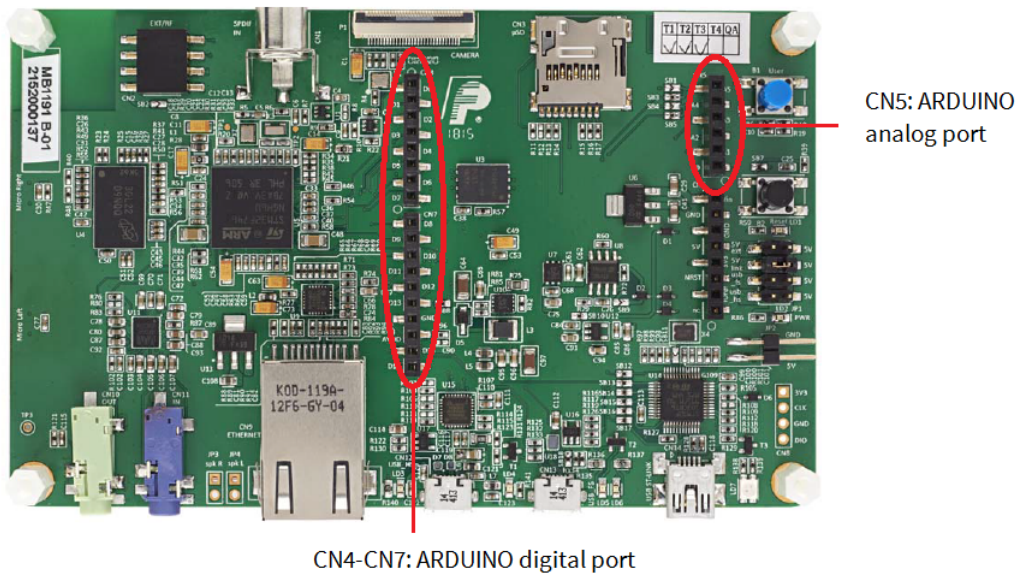
Figure 4.2. Communication Connectors



4.3. HAL Connectors

STM32F746G-DISCO provides several HAL GPIOs on connector CN4 to CN7

Figure 4.3. HAL Connectors



Chapter 5. Keil MDK-ARM Configuration

5.1. Install Keil MDK-ARM

This section describes how to install a Keil MDK-ARM Development Kit, how to activate the MDK-ARM license required for evaluating MicroEJ SDK and how to download the Keil pack related to the target microcontroller.

A right to activate a MDK-ARM evaluation license is granted with the MicroEJ SDK evaluation version. This evaluation license is a one month time limited that allows to compile and link up to 256KB of code.

5.1.1. Download Keil MDK-ARM

- Go to <http://www.keil.com/demo/eval/arm.htm>.
- Fulfill the registration form and press `Submit` button.
- Download the executable file (e.g. `MDKxyz.exe`).
- Run executable file and follow installation steps. Install additional software and drivers if proposed. A new application named `Keil uVision5` shall have been created.

5.1.2. Activate the Evaluation License

- Start `Keil uVision5` application. If you are using Windows 7 or higher, you should run the application with administrator privileges (right-click on `Keil uVision5` shortcut > `Run as administrator`).
- Select `File > License Management...` and copy the computer ID (CID).

Figure 5.1. Keil MDK-ARM Computer ID

License Management

Single-User License | Floating License | Floating License Administrator | FlexLM License

Customer Information

Name:

Company:

Email:

Computer ID
CID:

Get LIC via Internet...

Product	License ID Code...	Support Period
MDK-Lite	Evaluation Version	

New License ID Code (LIC):

Add LIC Uninstall...

Evaluate MDK Professional Close Help

- Go to <https://www.keil.com/license/install.htm>.
- Fulfill the registration form with the computer ID and the following Product Serial Number (PSN):

0YK7R-4V1PQ-R3KM9

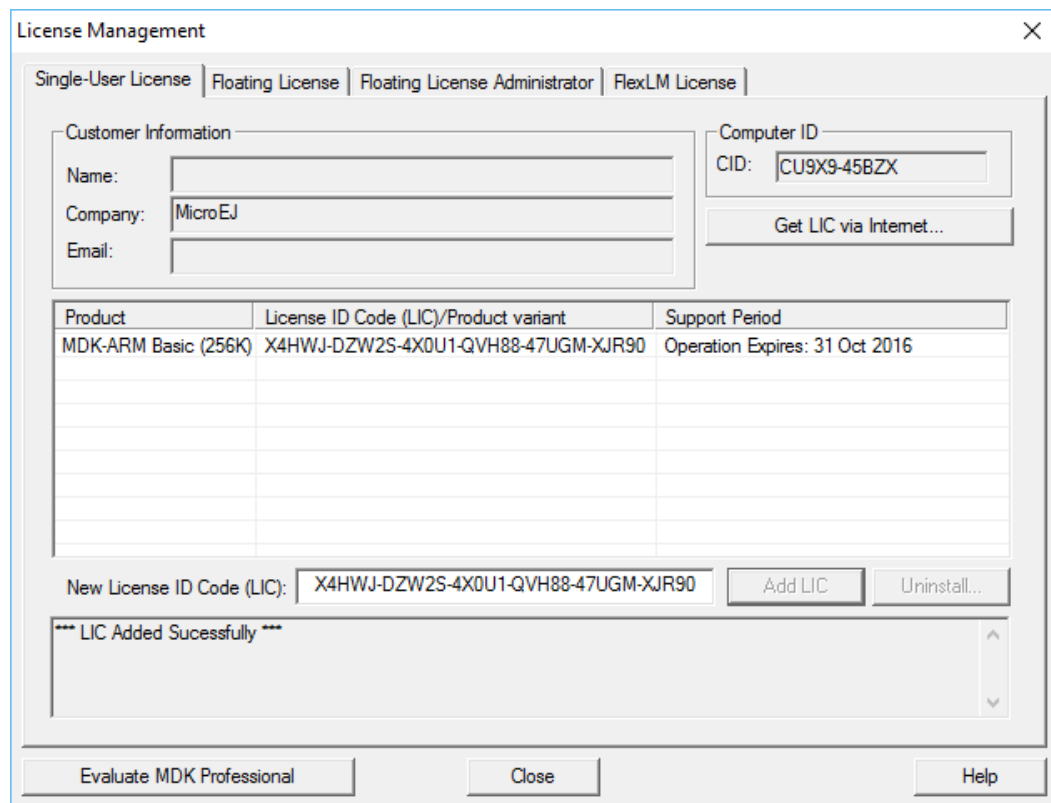


Warning

This Product Serial Number is reserved to activate a Keil MDK-ARM product license solely for a usage through this MicroEJ SDK evaluation.

- Press **Submit** button.
- You should have received an email with an activation license.
- Copy/Paste the license ID code (LIC) included in the email and press on **Add LIC** button.
- License is now installed and should display the information as shown below.

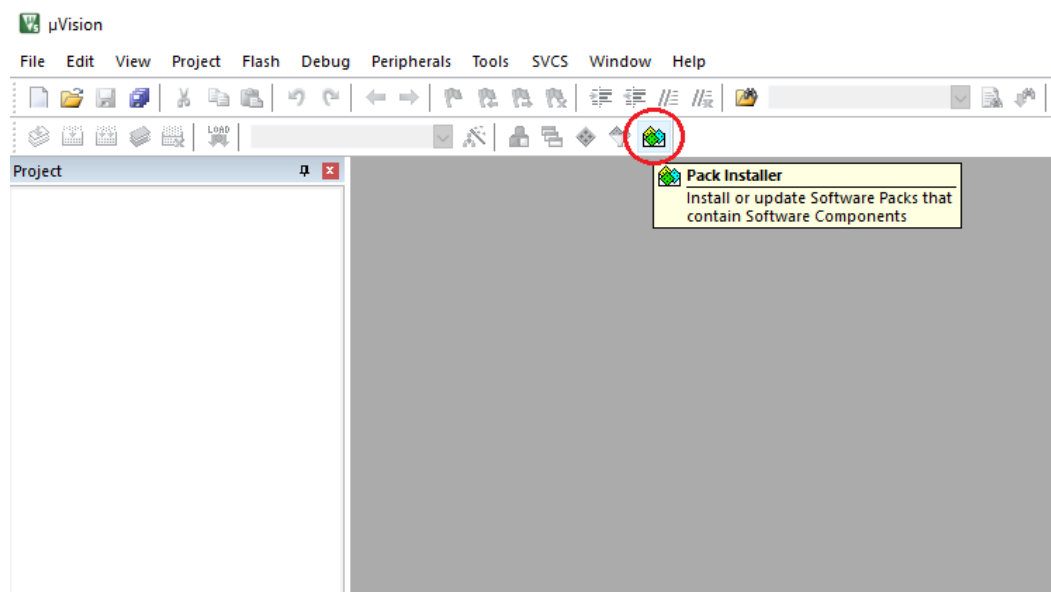
Figure 5.2. Keil MDK-ARM License Installation



5.1.3. Install Microcontroller Specific Pack

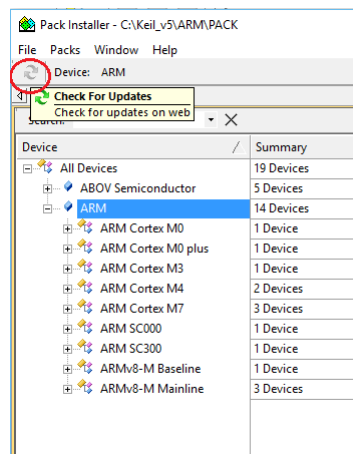
- In Keil `uVision5`, click on the pack installer icon.

Figure 5.3. Keil MDK-ARM Microcontroller Pack



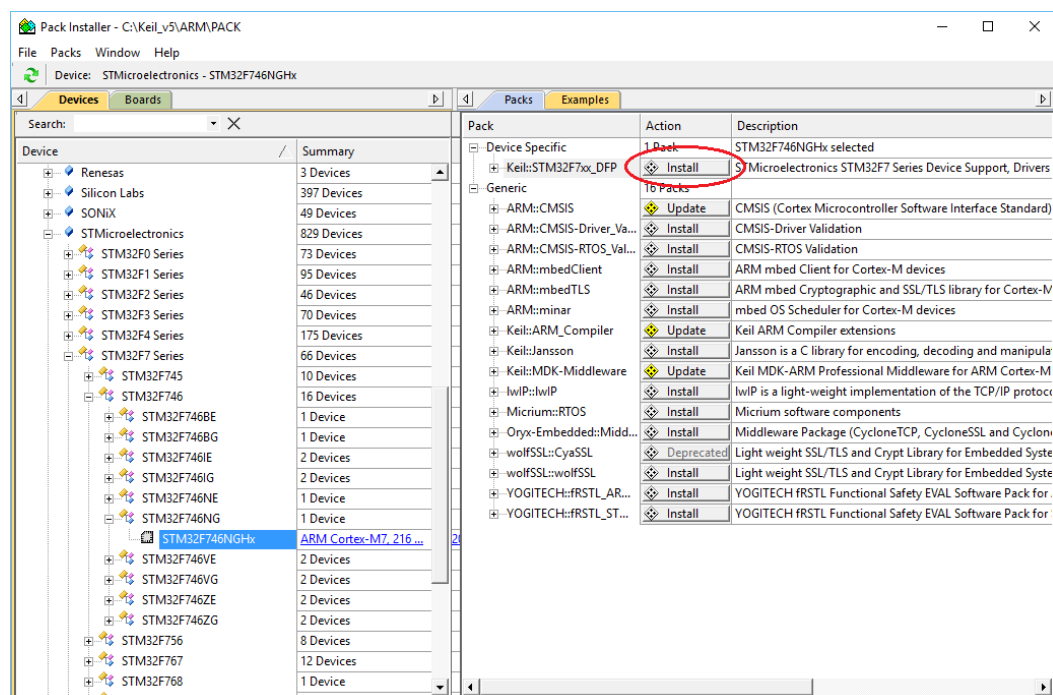
- Click on the refresh icon `Check for Updates` and wait until all available pack description are downloaded.

Figure 5.4. Microcontroller Pack Update



- In Devices tab, select the appropriate microcontroller part number All Devices > STMicroElectronics > STM32F7 Series > STM32F746 > STM32F746NG > STM32F746NGHx. In Packs Tab, select Device Specific > Keil::STM32F7xxx_DFP and click on Install.

Figure 5.5. Microcontroller Pack Selection



- The pack is downloaded and installed. Close the Pack Installer window.

5.2. BSP Project Structure

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

- Drivers: all MCU drivers, board drivers and CMSIS drivers

- Middlewares: all 3rd-party files: USB host library, FatFs, FreeRTOS, LWIP
- Projects: the MicroEJ platform project itself
- Utilities: miscellaneous C files required by drivers

The MDK-ARM BSP project file is `Projects/STM32746G-Discovery/Applications/MicroEJ/MDK-ARM/Project.uvprojx`. This MDK-ARM BSP project has been written for μ Vision V5.18.0.0. The project follows the files structure of STM32CubeF7 projects:

- Drivers/*: all MCU drivers, board drivers and CMSIS drivers
- FreeRTOS: FreeRTOS files
- LWIP/*: network LWIP files
- FatFS/*: file system files
- STM32_USBH/*: STMicroelectronics USBH drivers
- MicroEJ/*: all MicroEJ platform implementation files

The MicroEJ platform implementation files are grouped by MicroEJ features:

- MicroEJ/Core: Core Engine implementation over STM32CubeF7 and FreeRTOS (always required)
- MicroEJ/Comm: ECOM COMM implementation over UART and CDC
- MicroEJ/FS: File system implementation over FatFS
- MicroEJ/KF: Multi applications implementation over STM32CubeF7
- MicroEJ/HAL: HAL implementation over STM32CubeF7
- MicroEJ/Libs: MicroEJ platform C libraries
- MicroEJ/Net: Network implementation over LWIP
- MicroEJ/Net/SSL: SSL implementation over WolfSSL
- MicroEJ/UI: UI implementation over STM32CubeF7

Chapter 6. Changelog

6.1. Version 3.1.5

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

6.2. Version 3.1.4

MicroEJ 4.1 alignment (MWT moved as add-on library)

6.3. Version 2.2.0

Initial release of the platform.