

MicroEJ Platform Reference Implementation

Developer's Guide



Renesas S7G2DK 1.0.1

Reference:	TLT-801-DGI-PlatformReferenceImplementation-Renesas S7G2DK
Version:	1.0.1
Revision:	1.0.1

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 1.0.1	December 30th 2016	BJ
Early release for MicroEJ 4.1		
Revision 1.0	August 23th 2016	BJ
Initial version		

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 S7G2DK Board	6
2.3.1. Compile MicroEJ Standalone Application	6
2.3.2. Link and Deploy MicroEJ Standalone Application	7
3. Specification	9
3.1. Overview	9
3.2. MicroEJ Platform Configuration	9
3.3. Platform Output stream	9
3.4. RTOS Configuration	10
3.5. Memories	10
3.6. Graphical User Interface	11
3.6.1. Display	11
3.6.2. Inputs	12
3.7. Network	12
3.8. File System	12
3.9. Serial Communications	13
3.9.1. UART Connector	13
3.10. HAL	13
4. Board Configuration	15
4.1. Mandatory Connectors	15
4.2. Communication Connectors	15
4.3. HAL Connectors	16
5. Renesas e2 studio	18
5.1. Install Renesas e2 studio	18
5.1.1. Download and install Renesas e2 studio	18
5.1.2. Download and install Synergy Software Package (SSP)	18
5.2. BSP Project Structure	18
6. Changelog	20
6.1. Version 1.0.1	20
6.2. Version 1.0	20

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. Renesas e2 studio Project Selection	8
2.9. Renesas e2 studio IDE	8
4.1. Mandatory Connectors	15
4.2. Communication Connectors	16
4.3. HAL Connectors	17

List of Tables

3.1. MCU Technical Specifications	9
3.2. MicroEJ Configuration	9
3.3. ThreadX Tasks	10
3.4. Internal RAM: SRAM (640 KB)	10
3.5. External RAM: SDRAM (32 MB)	11
3.6. Internal flash: Program Flash (4 KB)	11
3.7. HAL GPIOs Ports and Pins	14
3.8. HAL GPIOs Pins Designation Mapping	14
3.9. HAL Analog IOs Pins Designation Mapping	14

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 Renesas S7G2DK 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 S7G2DK 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 S7G2DK board.
- Run a MicroEJ standalone application on the MicroEJ simulator.
- Run a MicroEJ standalone application on the MicroEJ platform and deploy it on the S7G2DK board.



Note

This platform is delivered as a developer preview. It shall only be used for early development stage and not for production. This platform development, documentation and validation is not complete. Its known limitations are described in the associated errata document. There is no guaranty that the APIs or features will be maintained.

1.3. Prerequisites

- PC with Windows 7 or later.
- The MicroEJ SDK environment must be installed.
- S7G2DK board.
- The Segger J-Link software.
- A GNU-C development environment. The examples are packaged ready to run using the e2studio 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.

Chapter 2. Create and Use Your First MicroEJ Platform

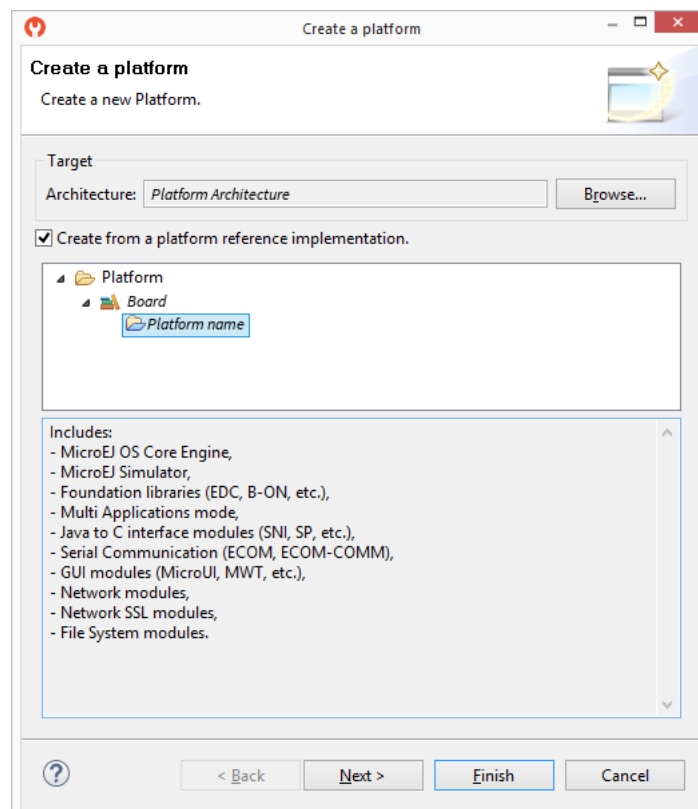
2.1. Create a MicroEJ Platform

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 S7G2DK.

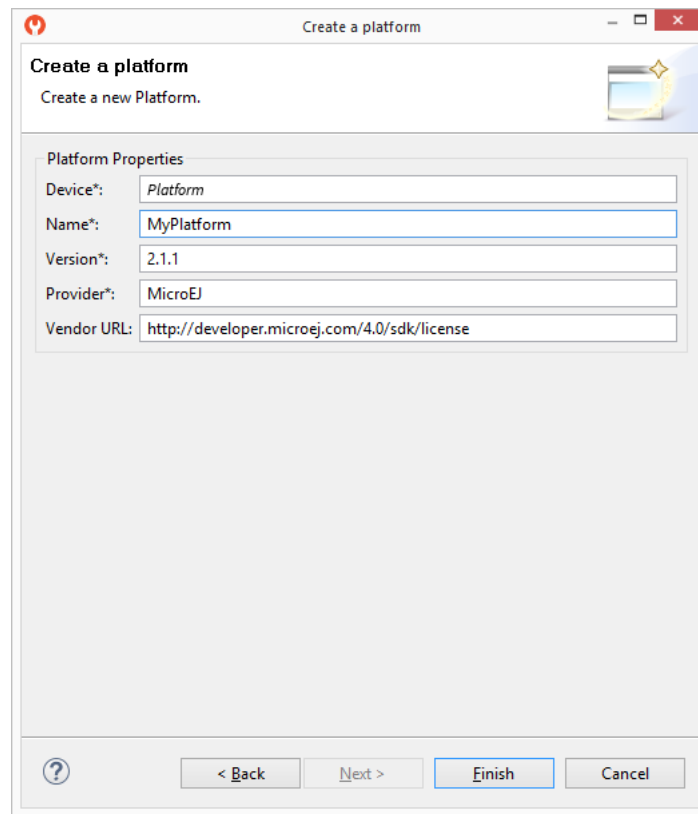
- Open MicroEJ SDK.
- Open the MicroEJ platform wizard: `File > New > Platform`.
- Select the MicroEJ architecture ARM Cortex-M4 GCC from the combo box. A MicroEJ Platform Reference Implementation is available:

Figure 2.1. MicroEJ Platform Reference Implementation Selection



- Select the MicroEJ platform SingleApp for the S7G2DK 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:
 - S7G2DK-MyPlatform-CM4hardfp_GCC48-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).
 - S7G2DK-MyPlatform-CM4hardfp_GCC48-bsp: Contains a ready-to-use BSP software project for the S7G2DK board, including a Renesas e2 studio project, an implementation of MicroEJ core engine (and extensions) port on ThreadX RTOS and the S7G2DK board support package.
 - S7G2DK-MyPlatform-CM4hardfp_GCC48-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:

Name:

Version:

Provider :

Vendor URL:

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 S7G2DK board and is ready to be linked into the Renesas e2 studio project. Its name is S7G2DK-MyPlatform-CM4hardfp_GCC48.

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

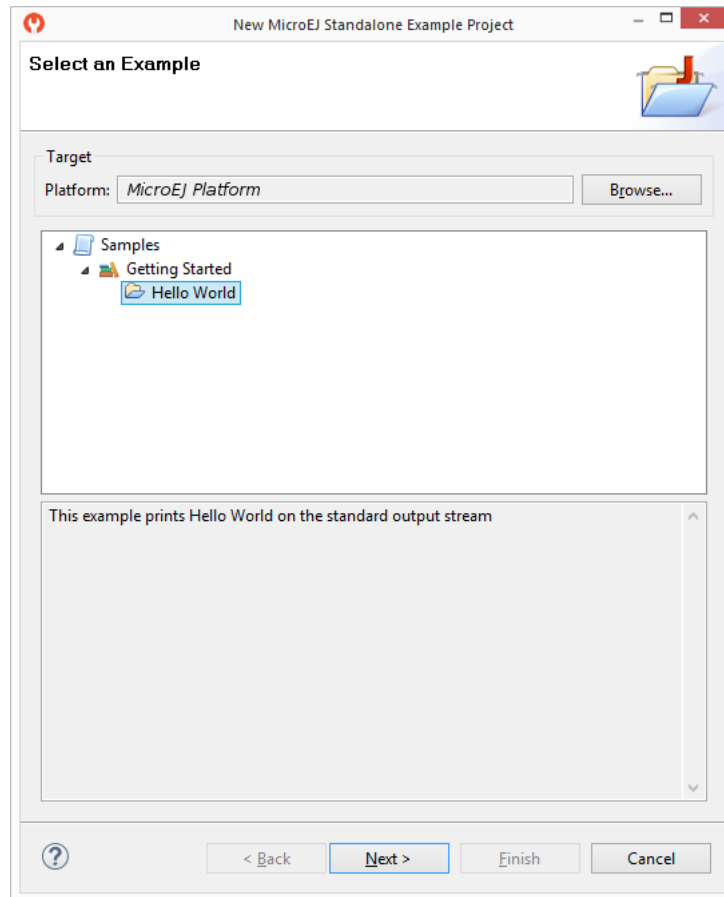
The aim of this chapter is to create a MicroEJ standalone application from a built-in example. Initially, this example will run on the MicroEJ simulator. Then, in the next section, this application will be compiled and deployed on the S7G2DK board using the MicroEJ platform.

2.2.1. Create Example

- Open MicroEJ SDK.
- Open the File > New > MicroEJ Standalone Example Project menu.
- Select the MicroEJ platform S7G2DK-MyPlatform-CM4hardfp_GCC48 from the combo box.

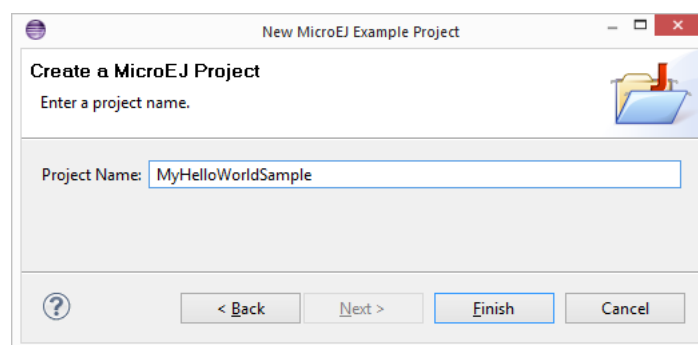
- Select the example `Samples > Getting Started > Hello World`.

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

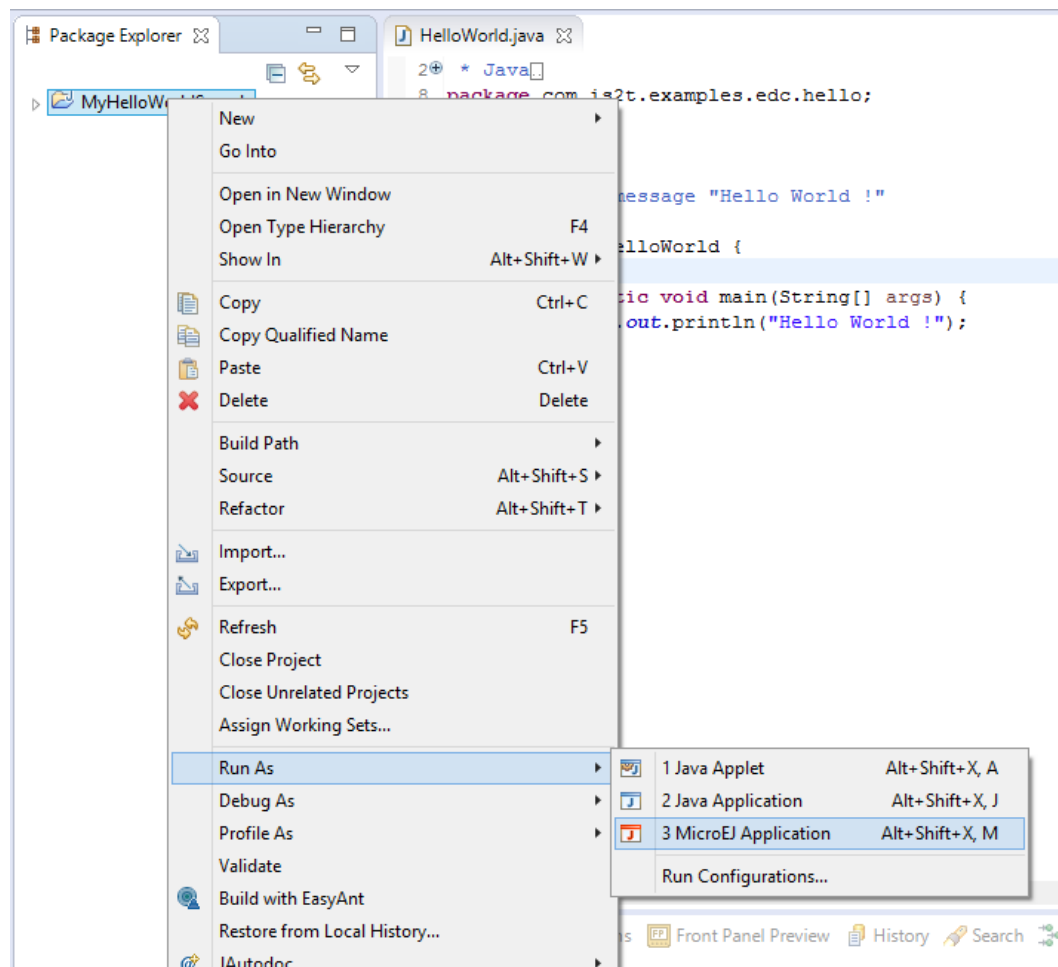


- Click on Finish. The selected example is imported into a project with the given name. The main class (the class which contains the `main()` method) is automatically opened.

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 (S7G2DK-MyPlatform-CM4hardfp_GCC48). The result of the test is printed in the console:

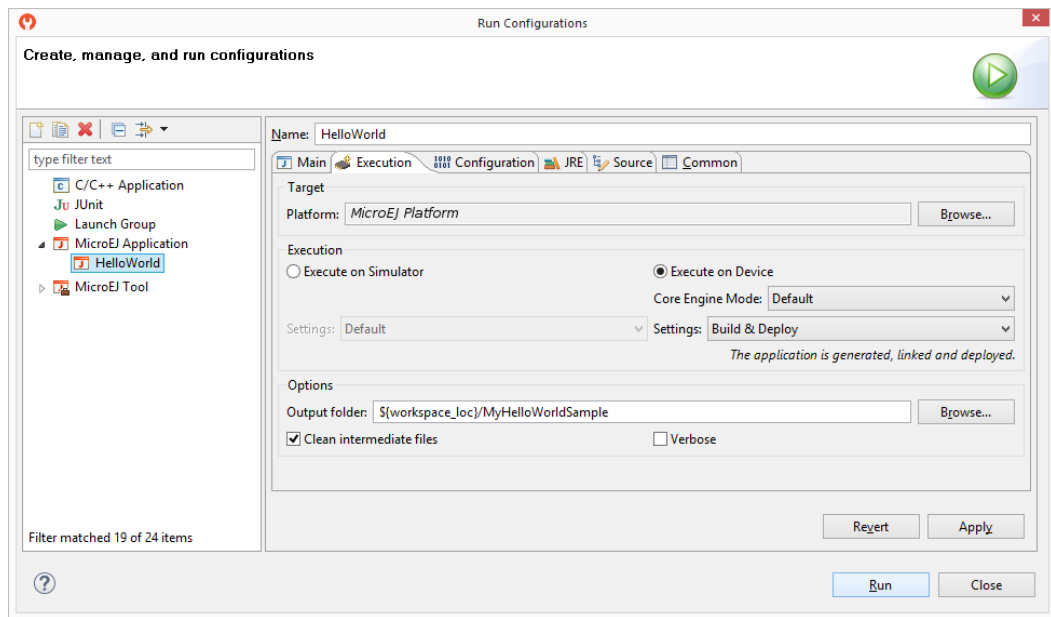
```
Hello World !
```

2.3. Run the Example on the S7G2DK 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 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 S7G2DK board.



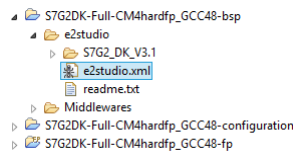
Note

This final step uses Renesas e2 studio 5.0.0.043.

The following steps are performed within MicroEJ.

- In MicroEJ SDK, expand the project S7G2DK-MyPlatform-CM4hardfp_GCC48-bsp and the folder `Projects/TWRK65F180M/Applications/MicroEJ`. A Ant script (`e2studio.xml`) is available.

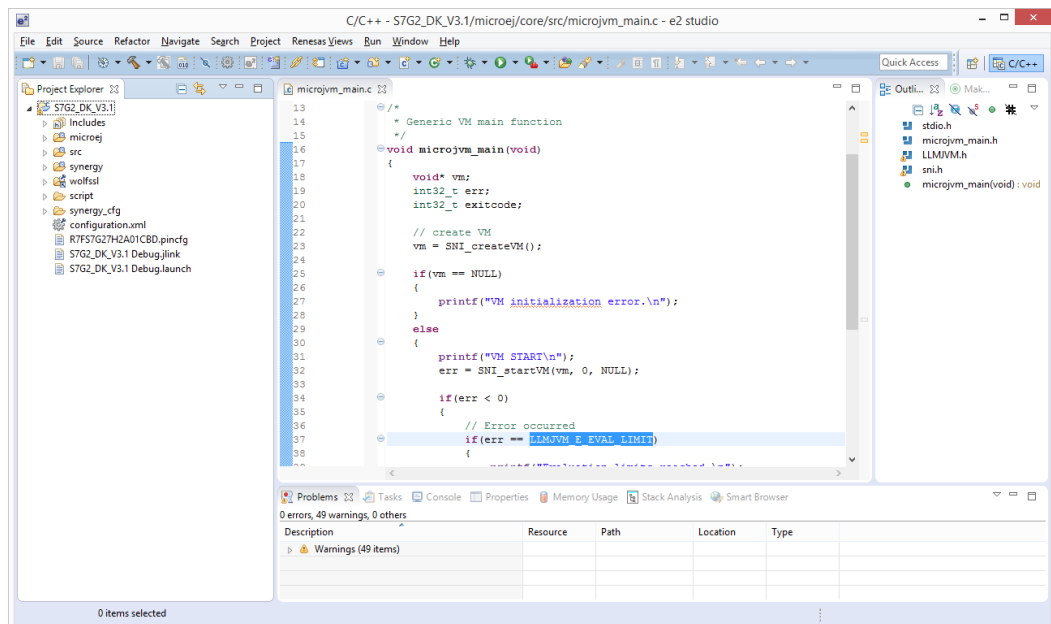
Figure 2.8. Renesas e2 studio Project Selection



- 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 S7G2DK board by clicking on the menu `Run > Debug Configurations... > Renesas GDB Hardware Debug > S7G2DK Debug`. 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. Congratulations, you have deployed a MicroEJ standalone application on a MicroEJ platform.

Chapter 3. Specification

3.1. Overview

MicroEJ platform on S7G2DK is based on board support package provided by Renesas: SSP. It includes ThreadX, 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 e2 studio and tested with SSP v1.1.3.

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
MCU Clock speed	240 MHz
Internal RAM	640 KB
External RAM	32 MB
Internal Flash	4 MB
External Flash	32 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	6.6.0
UI	8.1.0
Network	6.1.0
File System	2.2.1
HAL	1.0.3

3.3. Platform Output stream

MicroEJ platform uses a COM port as output print stream. The COM port is available on the connector J112 and is connected to the MCU UART 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

3.4. RTOS Configuration

MicroEJ platform uses ThreadX 5.7. RTOS uses a heap to allocate all its objects: tasks stacks, task monitors, semaphores etc. The heap size is: 16 KB and is allocated in internal RAM. The following table illustrates the available tasks and their stack size.

Table 3.3. ThreadX Tasks

Task name	Size	Priority
Core Engine	16 KB	6
Touch	4 KB	3
File System	3 KB	4
NET	2 KB	4
MCU Charge Calculation	1 KB	4
Framerate Calculation	1 KB	4

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: SRAM (640 KB)

Section Name	Size
MicroEJ standalone application heaps	16384 bytes ^a
MicroEJ standalone application stack blocks	512 * <i>n</i> bytes ^b
MicroEJ platform internal heap	<i>n</i> 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 RAM: SDRAM (32 MB)

Section Name	Size
Display buffers	255 KB
MicroUI working buffer	4 MB
Multi applications working buffer	3 MB
Any RW	n bytes ^a

^a n depends on MicroEJ application libraries used.

Table 3.6. Internal flash: Program Flash (4 KB)

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 platform features a graphical user interface. It includes a display, a touch panel, three user buttons 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.



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, the both buffers are swapped and then a copy of pixels data from the new frame buffer to the new back buffer is performed (stack *switch*). In order to avoid flickering, this switch 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.

MicroUI requires a RAM buffer to store the dynamic images data. A dynamic image is an image decoded at runtime (PNG image) or an image created by the MicroEJ application using the `Image.create(width, height)` API. 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 using a `Pointer` event generator.



Implementation Note

A touch *press* event is detected under interrupt. It wakes up a dedicated OS task. For all next *drag* events, the IO expander task runs in polling mode. When a *release* is detected, the task goes to sleep and waiting a touch interrupt.

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



Implementation Note

The user buttons events treatments are performed under interrupt.

3.7. Network

MicroEJ plaform features a network interface. Sockets are limited to 32. A DHCP client can be activated to retrieve an IP address.



Implementation Note

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 NetX and the DNS features use a MicroEJ software implementation.

3.8. 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 FileX.

3.9. Serial Communications

3.9.1. UART Connector

MicroEJ platform provides one serial connection (ECOM COMM) on UART1 port. UART1 pins are (RTS/CTS mode is not used):

- TX: P709; available on connector J112
- RX: P708; available on connector J112



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.

3.10. HAL

MicroEJ platform provides several GPIOs programmable via the HAL foundation library. All GPIOs are available on external connectors (PMODA and PMODB). Digital pins are implemented by a GPIO access.

Analog input pins (ADC) are driven by ADC channels of ADC 1 and analog output pins (DAC) drive PWM channels via DAC A (channels 6 and 8) and DAC B (channel 8).

Each GPIO port / pin value is accessible using either:

- The global MCU declaration designation: all pins of all ports are grouped under only one virtual port (port 0) and have consecutive values: P000 has the ID 0, P001, the ID 1, P015 the ID 15, P100 the ID 16 and so on. For instance pin *P400* is accessible by `(0, 64)`. This designation is useful to target all MCU pins using only one virtual port.
- The standard MCU declaration designation: Port 0 has the ID 1, Port 1 the ID 2 etc. Each pin of each port is a value between 0 (PortN0) to 15 (PortN15). For instance pin *P400* is accessible by `(4, 0)`. This designation is useful to target a specific MCU pin.
- The physical board connectors designation. Board has 2 connectors: PMODA and PMODB, with respectively these IDs: 64, 65. For instance pin *P400* is accessible on connector PMODB, pin 4: `(65, 4)`. 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 pins IDs:

Table 3.7. HAL GPIOs Ports and Pins

Port name	HAL port ID	Pins range
Global MCU virtual port	0	0 to 173
MCU port 0	1	0 to 15
MCU port 4	2	0 to 15
MCU port 9	4	0 to 15
MCU port B	6	0 to 15
Board physical port "PMDA"	64	1 to 12
Board physical port "PMODB"	65	1 to 12

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 3.8. HAL GPIOs Pins Designation Mapping

Port / Pin	MCU virtual port (1)	MCU port (2)	Board physical port (3)
P004	0, 4	1, 4	64, 7
P009	0, 9	1, 9	65, 1
P400	0, 64	5, 0	65, 0
P911	0, 155	10, 11	64, 8
P912	0, 156	10, 12	64, 9
P913	0, 157	10, 13	64, 10
PB02	0, 172	12, 2	64, 1
PB03	0, 173	12, 3	64, 4

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

Table 3.9. HAL Analog IOs Pins Designation Mapping

Port / Pin	ADC id / channel	PWM timer / channel
P004	1 / 0	-
P400	-	A / 6
P911	-	B / 8
P912	-	A / 8

Chapter 4. Board Configuration

S7G2DK provides several connectors, each connector is used by the MicroEJ Core Engine itself or by a foundation library.

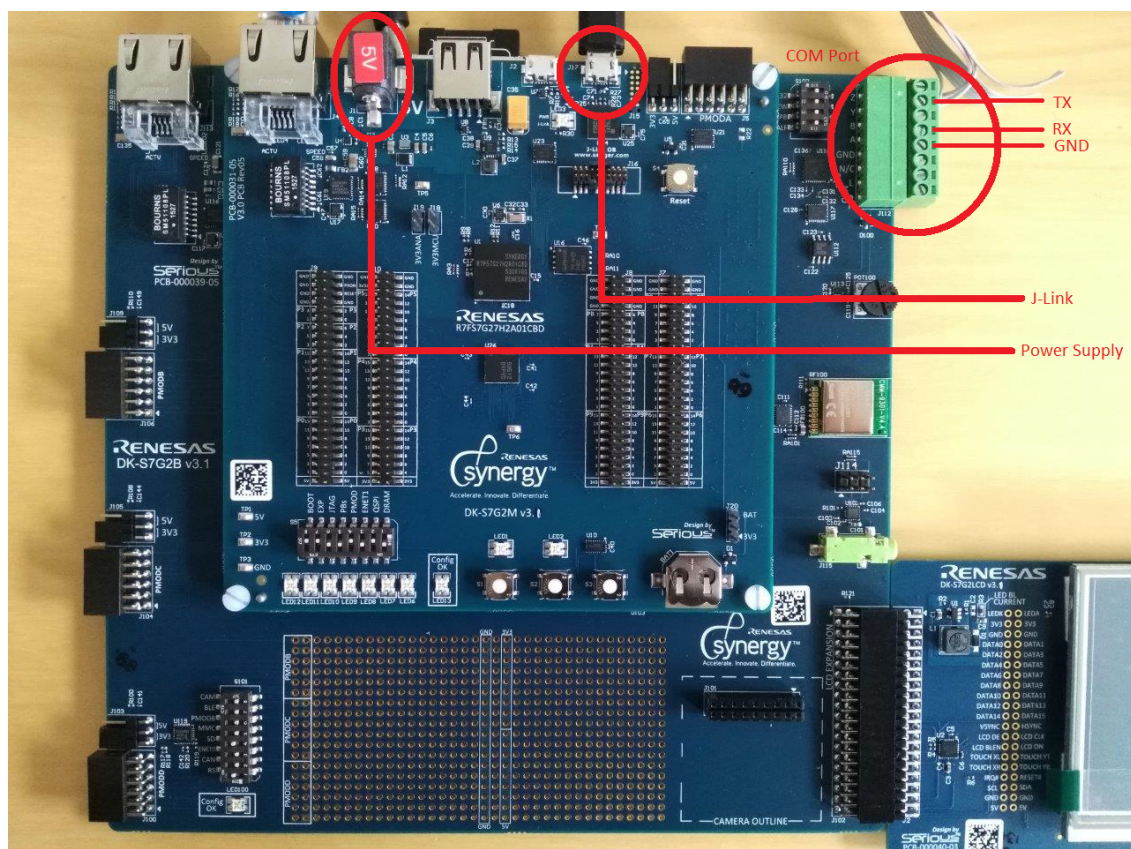
4.1. Mandatory Connectors

S7G2DK provides three connectors used as:

- Power supply connector
- Probe connector
- COM port

Plug a serial cable to a computer to see the MicroEJ standalone application `System.out.print` traces.

Figure 4.1. Mandatory Connectors

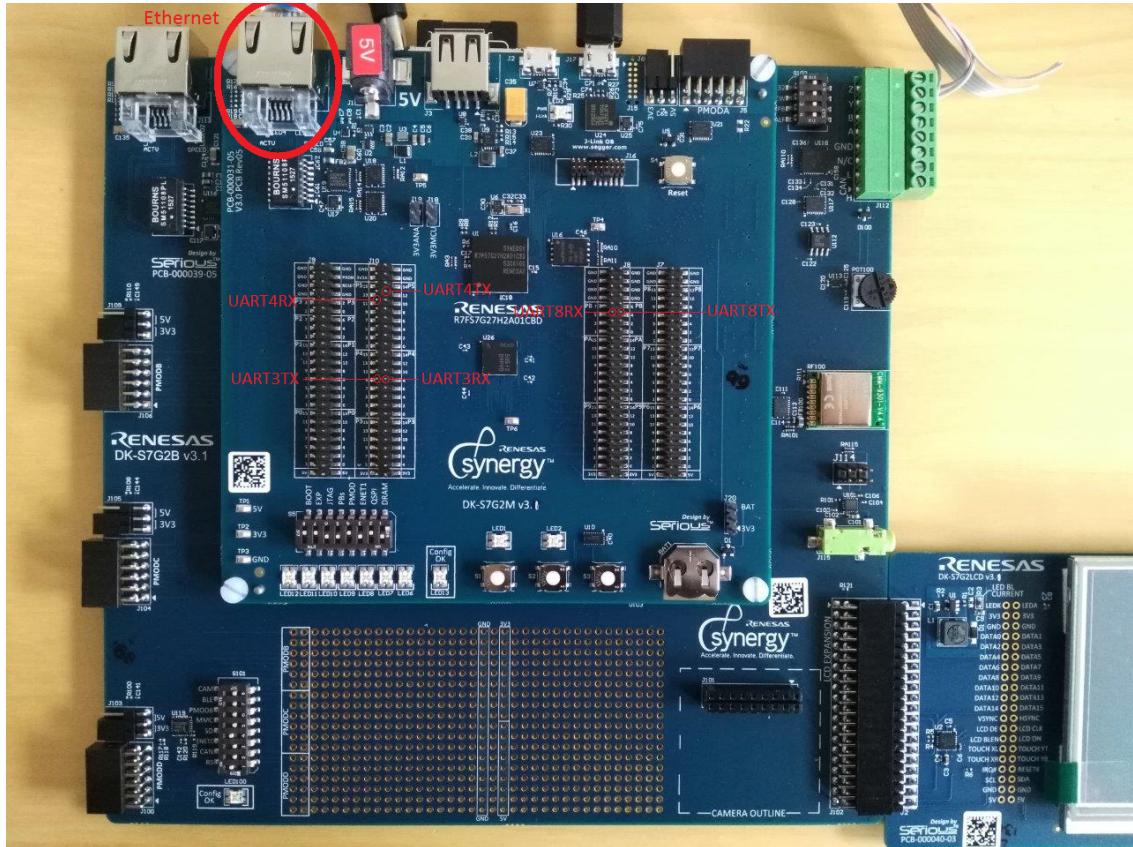


4.2. Communication Connectors

S7G2DK provides several communication ports:

- Ethernet
- Serial communication

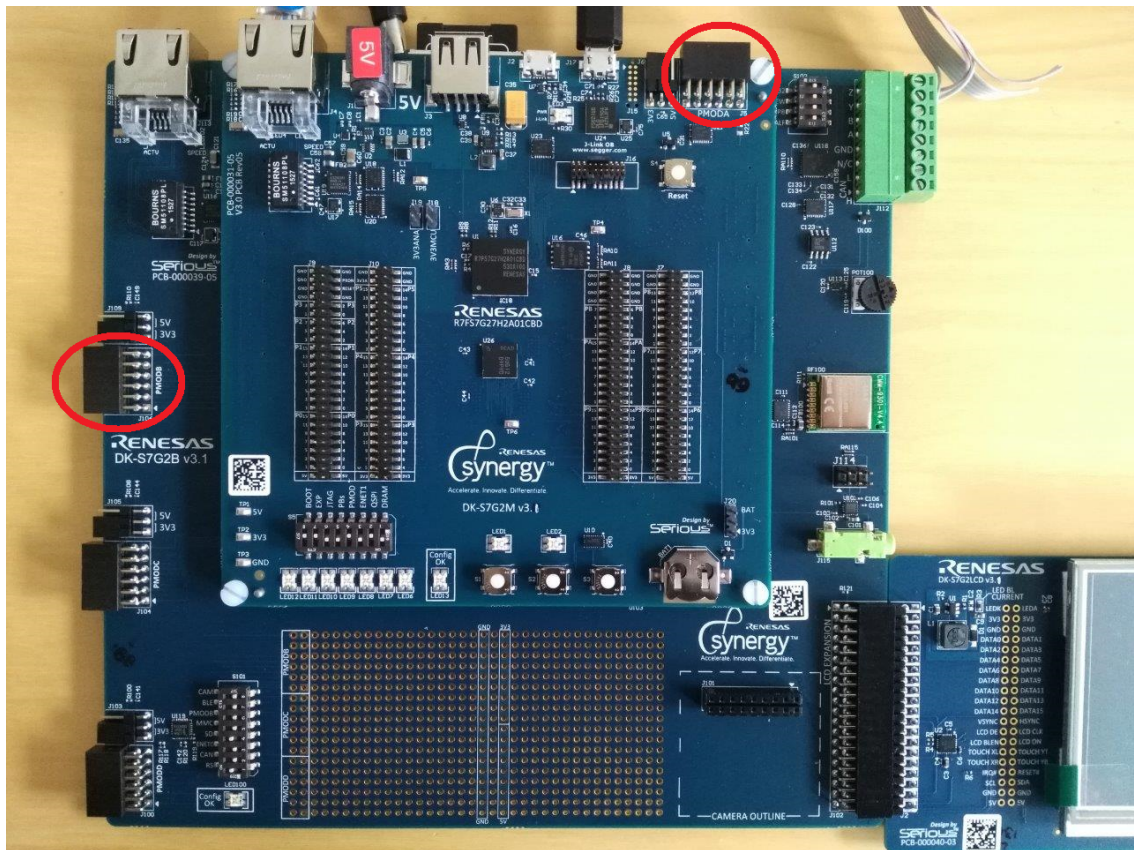
Figure 4.2. Communication Connectors



4.3. HAL Connectors

S7G2DK provides several HAL GPIOs on connectors PMODA and PMODB

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 Synergy Software Package (SSP)

- Download the version 1.1.3 of SSP from Renesas (you will need an account).
- Run executable file and follow the installation steps. Install additional software and drivers if proposed.

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
- `Middlewares`: all 3rd-party files (e.g. `wolfSSL`)

The e2 studio BSP project file is `e2studio/S7G2_DK_V3.1/.project`. This e2 studio BSP project has been written for e2 studio v5.0.0.043. The project has the following file structure:

- `microej/*`: all MicroEJ platform implementation files
- `src/*`: all the ThreadX related files, including SSP-generated code
- `synergy/*`: SSP files

The MicroEJ platform implementation files are grouped by MicroEJ features:

- microej/comm/inc: ECOM COMM implementation over UART include files
- microej/comm/src: ECOM COMM implementation over UART source files
- microej/core/inc: Core Engine implementation over SSP and ThreadX include files (always required)
- microej/core/src: Core Engine implementation over SSP and ThreadX source files (always required)
- microej/fs/inc: File system implementation over SSP include files
- microej/fs/src: File system implementation over SSP source files
- microej/hal/inc: HAL implementation over SSP include files
- microej/hal/src: HAL implementation over SSP source files
- microej/kf/inc: Multi applications implementation over SSP include files
- microej/kf/src: Multi applications implementation over SSP source files
- microej/net/inc: Network implementation over SSP include files
- microej/net/src: Network implementation over SSP source files
- microej/ssl/inc: SSL implementation over WolfSSL include files
- microej/ssl/src: SSL implementation over WolfSSL source files
- microej/ui/inc: UI implementation over SSP include files
- microej/ui/src: UI implementation over SSP source files

Chapter 6. Changelog

6.1. Version 1.0.1

- WI 12372: invalid getcurrenttime returned by the LLMJVM_getcurrenttime
- WI 17713: improve the number of display stacks to build
- WI 17890: SDRAM or cache configuration problem
- WI 18363: NET 1.1 port for S7G2DK
- WI 18504: Missing implementation of LLEDS_IMPL_getIntensity for S7G2DK

6.2. Version 1.0

Initial release of the platform.