### **MicroEJ Platform**

Developer's Guide



### Renesas S7G2DK 1.0.1

Reference: Version: Revision: TLT-802-DGI-Platform-Renesas S7G2DK 1.0.1 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<sup>®</sup>. 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<sup>®</sup> and MicroEJ<sup>®</sup> and all relative logos are trademarks or registered trademarks of IS2T S.A. in France and other Countries.

Java<sup>™</sup> 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 <sup>™</sup> symbol, it includes implementations of the technology by companies other than Sun.

Java<sup>™</sup>, 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. Develop and Run Your First MicroEJ Standalone Application
2.1. Run an Example on the MicroEJ Simulator 2
2.1.1. Create Example 2
2.1.2. Run Example 3
2.2. Run the Example on the S7G2DK Board 4
2.2.1. Compile MicroEJ Standalone Application 4
2.2.2. SEGGER J-Link Programming Tool 4
3. Specification 7
3.1. Overview
3.2. MicroEJ Platform Configuration 7
3.3. Platform Output stream 7
3.4. Memories 8
3.5. Graphical User Interface 8
3.5.1. Display 8
3.5.2. Inputs
3.6. Network
3.7. File System
3.8. Serial Communications 9
3.8.1. UART Connector 9
3.9. HAL
4. Foundation Libraries 11
4.1. List 11
5. Board Configuration 12
5.1. Mandatory Connectors 12
5.2. Communication Connectors 12
5.3. HAL Connectors 13
6. Changelog 15
6.1. Version 1.0.1 15
6.2. Version 1.0 15

## List of Figures

2.1. MicroEJ Standalone Application Selection	2
2.2. MicroEJ Standalone Application Naming	3
2.3. MicroEJ Standalone Application Running	3
2.4. Execution on Device	4
2.5. MicroEJ Tool Launcher Creation	5
2.6. SEGGER J-Link MicroEJ SDK Tool Window	5
2.7. SEGGER J-Link MicroEJ SDK Tool Configuration Window	6
5.1. Mandatory Connectors 1	.2
5.2. Communication Connectors 1	.3
5.3. HAL Connectors 1	.4

### List of Tables

3.1. MCU Technical Specifications	7
3.2. MicroEJ Configuration	7
3.3. Internal RAM: SRAM (640 KB)	8
3.4. External RAM: SDRAM (32 MB)	, 8
3.5. Internal flash: Program Flash (4 KB)	. 8
3.6. HAL GPIOs Ports and Pins	10
3.7. HAL GPIOs Pins Designation Mapping	10
3.8. HAL Analog IOs Pins Designation Mapping	10
4.1. Foundation Libraries	11

# **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 Renesas S7G2DK board.
- 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

- 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.

# 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 S7G2DK 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 S7G2DK-FI7B4 from the combo box.
- Select the example Samples > Getting Started > Hello World.

Figure 2.1. MicroEJ Standalone Application Selection

	New Micro	oEJ Standalone Exampl	e Project	
Select an Ex	ample			P
Target				
Platform: M	croEJ Platform			B <u>r</u> owse
⊿ 📄 Samp ⊿ 🛋 Ge	les etting Started Hello World			
This example	prints Hello World on 1	the standard output st	ream	
This example	prints Hello World on t	the standard output st	ream	

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

#### Figure 2.2. MicroEJ Standalone Application Naming

۲	New MicroEJ Example Project	_ 🗆 🗙
Create a Mic Enter a project	roEJ Project name.	
Project Name:	MyHelloWorldSample	
?	< <u>Back</u> Next > <u>Finish</u>	Cancel

• 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

- -🛱 Package Explorer 🔀 🚺 HelloWorld.java 🔀 2⊕ \* Java.. E 3  $\nabla$ nackage com 2t.examples.edc.hello; D Dynamicski k Province Providence Provid New Go Into Open in New Window essage "Hello World !" Open Type Hierarchy F4 lloWorld { Show In Alt+Shift+W ▶ ic void main(String[] args) { Ctrl+C Ð Copy out.println("Hello World !"); Þ Copy Qualified Name Paste Ctrl+V ß × Delete Delete **Build Path** . Alt+Shift+S ▶ Source Refactor Alt+Shift+T ▶ Import... è പ്പ Export... Ś Refresh F5 Close Project **Close Unrelated Projects** Assign Working Sets... Run As 🖭 🛛 1 Java Applet Alt+Shift+X. A ۲ Debug As ۲ 2 Java Application Alt+Shift+X, J Profile As ۲ 3 MicroEJ Application Alt+Shift+X, M Validate Run Configurations... Build with EasyAnt Restore from Local History... 🖭 Front Panel Preview 🍵 History 🛷 Search 🎲 JAutodoc æ

Figure 2.3. MicroEJ Standalone Application Running

The application starts. It is executed on the MicroEJ simulator of the selected MicroEJ platform (S7G2DK-FI7B4). The result of the test is printed in the console:

Hello World !

### 2.2. Run the Example on the S7G2DK 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

9	Run Conf	igurations	
Create, manage, and run cor	nfigurations		
C 🗎 🗙 🕒 🔆 🕶	Name: HelloWorld		
type filter text	🗇 Main 💣 Execution 📲 Configuration	n 🛋 JRE 🥪 Source 🔲 <u>C</u> ommon	
C/C++ Application	Target		
Ju JUnit 🕨 Launch Group	Platform: MicroEJ Platform		Browse
MicroEJ Application HelloWorld	Execution Execute on Simulator	Execute on Device	
MicroEJ Tool		Core Engine Mode: Default	¥
	Settings: Default	Settings: Build & Deploy	~
		The application is generated,	linked and deployed.
	Options		
	Output folder: \${workspace_loc}/MyHello	WorldSample	B <u>r</u> owse
	Clean intermediate files	Verbose	
Filter matched 19 of 24 items		Revert	Apply
(?)		Run	Close

• 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 S7G2DK 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:





• A new window appears. Give a name to the launcher and set the MicroEJ platform field to S7G2DK-FI7B4 and the Settings field to Program with Segger J-Link

Ś	Run Configurations
Create, manage, and run config (Configuration): Segger JLink > The	urations Intry "Executable file" must be specified.
Image: Second system     type filter text     Image: C/C++ Application     Ju JUnit     Image: Launch Group     Image: MicroEl Application     Image: MicroEl Tool     Image: New_configuration	Name   Flash My Board     Image:   Execution     Target   Platform:     Execution   Browse     Execution   Settings:     Program a binary file on board using the Segger JLink probe.   V     Options   Output folder:   S(microej_loc/MyHelloWorldSample     Image:   Image:   Image:     Image:   Image:   Verbose
Filter matched 6 of 14 items	Revert Apply
0	<u>R</u> un Close

Figure 2.6. SEGGER J-Link MicroEJ SDK Tool Window

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

	Run Configurations
nfigurations	
Name: Flash My Board	on RE Common MicroEJ Application Definition Executable File: mple\com.is2t.examples.edc.hello.HelloWorld\application.out Browse JLink Settings JLink Installation Directory: C:\Program Files (x86)\SEGGER\ULink_V512e Browse
	nfigurations

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. Congratulations, you have deployed a MicroEJ standalone application on a MicroEJ platform.

# **Chapter 3. Specification**

### 3.1. Overview

MicroEJ platform on S7G2DK includes a graphical user interface, a TCP/IP network connection, a file system on SD card, COM connections (via UART) 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
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 architecure 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.

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: SRAM (640 KB)

Section Name	Size
MicroEJ standalone application heaps	16384 bytes <sup>a</sup>
MicroEJ standalone application stack blocks	512 * <i>n</i> bytes <sup>b</sup>
MicroEJ platform internal heap	<i>n</i> bytes <sup>c</sup>
SSL buffers	128 KB

<sup>a</sup> Maximum size of the addition of MicroEJ heap size and MicroEJ immortal heap size. These sizes are defined in MicroEJ Application launcher options.

<sup>b</sup> *n* is the number of stack blocks defined in MicroEJ Application launcher options.

<sup>c</sup> *n* depends on memory configuration set in MicroEJ Application launcher options.

#### Table 3.4. 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	<i>n</i> bytes <sup>a</sup>

<sup>a</sup> *n* depends on MicroEJ application libraries used.

#### Table 3.5. Internal flash: Program Flash (4 KB)

Section Name	Size
Any RO	<i>n</i> bytes <sup>a</sup>

<sup>a</sup> *n* depends on MicroEJ application, MicroEJ libraries, Board support package, RTOS, drivers, etc.

### 3.5. Graphical User Interface

This MicroEJ plaform features a graphical user interface. It includes a display, a touch panel, three user buttons and a runtime PNG decoder.

### 3.5.1. 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.

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.

### 3.5.2. Inputs

Touch panel: All touch panel events are sent to the MicroEJ application using 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.

### 3.6. Network

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

### 3.7. 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.

### 3.8. Serial Communications

### 3.8.1. 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

## 3.9. 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:

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 "PMODA"	64	1 to 12
Board physical port "PMODB"	65	1 to 12

#### Table 3.6. HAL GPIOs Ports and Pins

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):

1,9

5,0

10,11

10,12

10,13

12, 2

12,3

Board physical port (3) 64, 7

65,1

65,0

64,8

64,9

64,10

64,1

64,4

Port / Pin	MCU virtual port (1)	MCU port (2)
P004	0,4	1,4

0,9

0,64

0,155

0,156

0,157

0,172

0,173

#### Table 3.7. HAL GPIOs Pins Designation Mapping

P009

P400

P911

P912

P913

PB02

PB03

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

#### Table 3.8. 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. Foundation Libraries

## 4.1. List

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

Name	Version
EDC	1.2
BON	1.2
ECOM	1.1
ECOM-COMM	1.1
NLS	2.0
SNI	1.2
SP	2.0
MicroUI	2.0
MWT	2.1
NET	1.1
FS	2.0

#### Table 4.1. Foundation Libraries

# Chapter 5. Board Configuration

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

## 5.1. 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 5.1. Mandatory Connectors

## 5.2. Communication Connectors

S7G2DK provides several communication ports:

- Ethernet
- Serial communication

#### Figure 5.2. Communication Connectors



## 5.3. HAL Connectors

S7G2DK provides several HAL GPIOs on connectors PMODA and PMODB

Figure 5.3. HAL Connectors



# 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 LLLEDS\_IMPL\_getIntensity for S7G2DK

### 6.2. Version 1.0

Initial release of the platform.