MicroEJ Platform Reference Implementation

Developer's Guide

TWRK65F180M 1.1.5

Reference:	TLT-0791-DGI-PlatformReferenceImplementation-TWRK65F180M
Version:	1.1.5
Revision:	1.1.5

Confidentiality & Intellectual Property

Revision History		
Revision 1.1.5	July 05th 2016	
Refactored the changelog section		
Revision 1.1.4	July 05th 2016	
Added changelog section		
Revision 1.1.3	June 30th 2016	
Fixed regression on display issue at reset		
Revision 1.1.2	June 28th 2016	
Fixed display issue at reset		
Revision 1.1.1	June 23th 2016	
Added a picture for the touchscreen strap instructions		
Revision 1.1.0	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 3
    2.1. Create a MicroEJ Platform 3
    2.2. Run an Example on the MicroEJ Simulator 5
        2.2.1. Create Example 5
        2.2.2. Run Example 6
    2.3. Run the Example on the TWR-K65F180M Board 7
        2.3.1. Compile MicroEJ Standalone Application 7
        2.3.2. Link and Deploy MicroEJ Standalone Application 8
3. Specification 11
    3.1. Overview 11
    3.2. MicroEJ Platform Configuration 11
    3.3. Platform Output stream 11
    3.4. RTOS Configuration 12
    3.5. Memories 12
    3.6. Multi Applications 13
    3.7. Graphical User Interface 13
        3.7.1. Display 14
        3.7.2. Inputs 14
    3.8. Network 15
    3.9. File System 15
    3.10. Serial Communications 15
        3.10.1. UART Connector 15
    3.11. HAL 16
4. Board Configuration 18
    4.1. Mandatory Connectors 18
    4.2. Communication Connectors 18
    4.3. HAL Connectors 20
    4.4. UI Setup 21
5. Freescale Kinetis Design Studio Configuration 24
    5.1. Install Freescale Kinetis Design Studio 24
        5.1.1. Download Freescale Kinetis Design Studio 24
    5.2. BSP Project Structure 24
6. Changelog 26
    6.1. Version 1.1.5 26

List of Figures

2.1. MicroEJ Platform Reference Implementation Selection 3
2.2. New MicroEJ Platform Naming 4
2.3. MicroEJ Platform Build 5
2.4. MicroEJ Standalone Application Selection 6
2.5. MicroEJ Standalone Application Naming 6
2.6. MicroEJ Standalone Application Running 7
2.7. Execution on Device 8
2.8. Deploy Configuration 8
2.9. Freescale Kinetis Design Studio Project Selection 9
2.10. Freescale Kinetis Design Studio IDE 10
4.1. Mandatory Connectors 18
4.2. Communication Connectors - Ethernet 19
4.3. Communication Connectors - Serial 20
4.4. HAL Connectors 21
4.5. Touchscreen Strap 23

List of Tables

3.1. MCU Technical Specifications 11
3.2. MicroEJ Configuration 11
3.3. FreeRTOS Tasks 12
3.4. Internal RAM: SRAM_L (64 KB) 13
3.5. Internal RAM: SRAM_U (192 KB) 13
3.6. External RAM: SDRAM (256 MB) 13
3.7. Internal flash: Program Flash (2 MB) 13
3.8. HAL GPIOs Ports and Pins 16
3.9. HAL GPIOs Declaration (port, pin) 17
3.10. HAL Analog IOs Declaration (port, pin) 17
4.1. SW1 Configuration Switches 22
4.2. SW5 Configuration Switches 22

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 TWR-K65F180M board.

Note

In this document, all the references to the TWR-K65F180M board point to a full NXP Tower system made of the following:

- 2 TWR-ELEV boards (primary and secondary)
- 1 TWR-K65F180M board
- 1 TWR-SER board
- 1 TWR-LCD board connected to the primary TWR-ELEV 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 TWR-K65F180M 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 TWR-K65F180M board.
- Run a MicroEJ standalone application on the MicroEJ simulator.
- Run a MicroEJ standalone application on the MicroEJ platform and deploy it on the TWR-K65F180M board.

1.3. Prerequisites

- PC with Windows 7 or later.
- The MicroEJ SDK environment must be installed.
- TWR-K65F180M board.
- The Segger J-Link software.
- A GNU-C development environment. The examples are packaged ready to run using the Freescale Kinetis Design Studio C IDE, which this document assumes has been successfully in-

stalled. Please visit the NXP website to obtain a version of the Freescale Kinetis Design Studio C IDE. Note, however, that developers are free to use a different CDT packaging..

Chapter 2. Create and Use Your First MicroEJ Platform

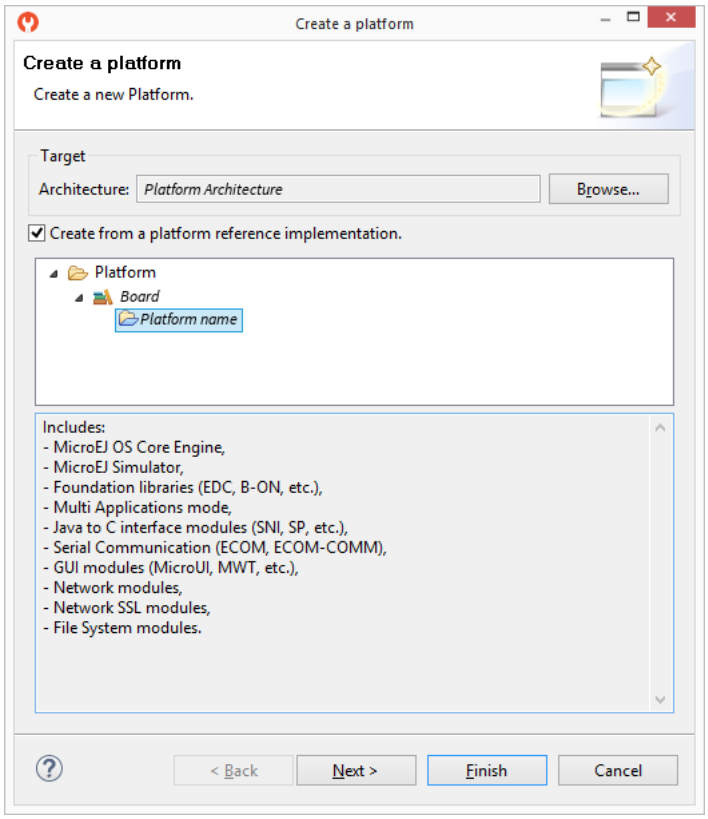
2.1. Create a MicroEJ Platform

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 TWR-K65F180M.

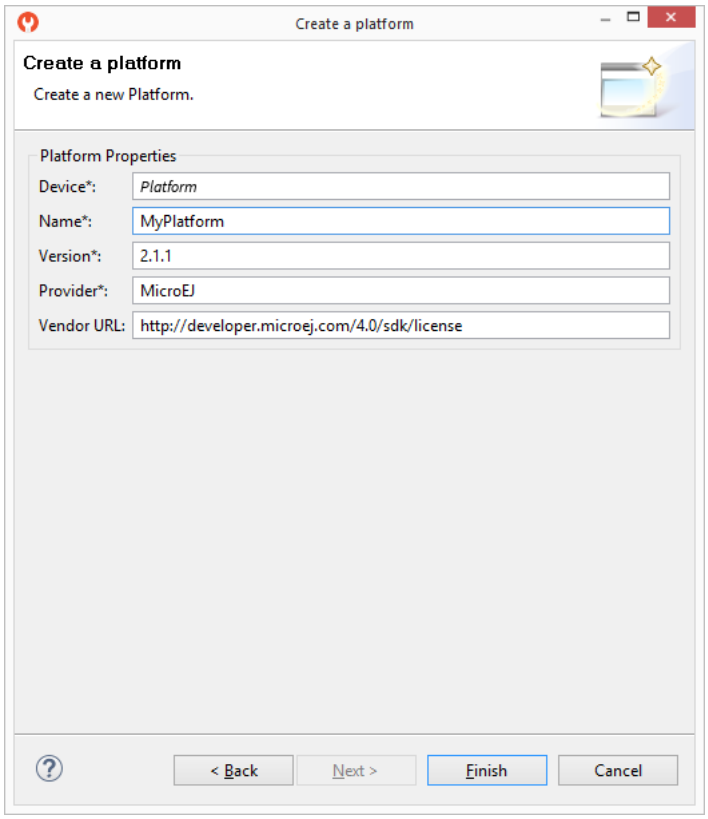
- Open MicroEJ SDK.
- Open the MicroEJ platform wizard: `File > New > Platform`.
- Select the MicroEJ architecture ARM Cortex-M4 GCC from the combo box. A MicroEJ Platform Reference Implementation is available:

Figure 2.1. MicroEJ Platform Reference Implementation Selection



- Select the MicroEJ platform Evaluation for the TWR-K65F180M 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:

  - TWRK65F180M-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`).

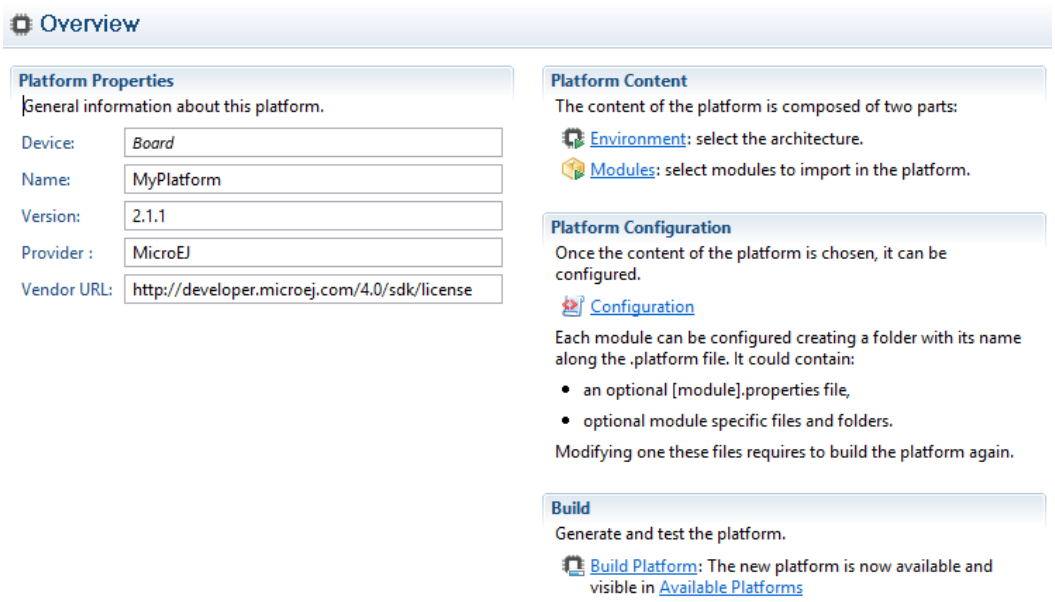
  - TWRK65F180M-MyPlatform-CM4hardfp_GCC48-bsp: Contains a ready-to-use BSP software project for the TWR-K65F180M board, including a Freescale Kinetis Design Studio project, an implementation of MicroEJ core engine (and extensions) port on FreeRTOS RTOS and the TWR-K65F180M board support package.

  - TWRK65F180M-MyPlatform-CM4hardfp_GCC48-fp: Contains the board description and images for the MicroEJ simulator. This project is updated once the platform is built. It

  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 TWR-K65F180M board and is ready to be linked into the Freescale Kinetis Design Studio project. Its name is `TWRK65F180M-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

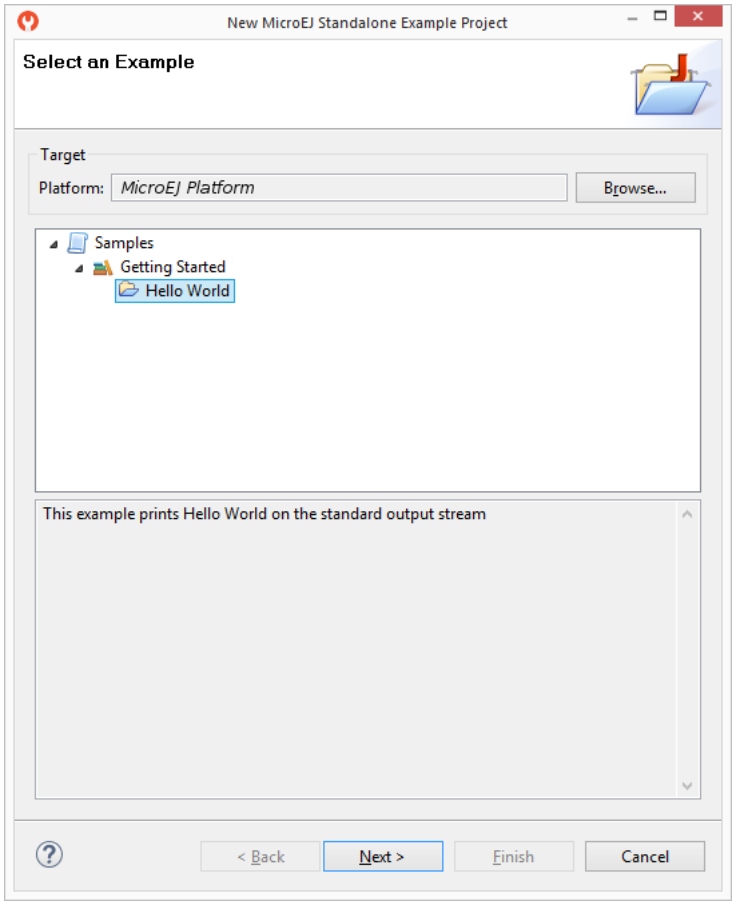
The aim of this chapter is to create a MicroEJ MicroEJ standalone application from a built-in example. This example will initially be run on the MicroEJ simulator. Then, in the next section, this application will be compiled and deployed on the TWR-K65F180M board using the MicroEJ platform.

2.2.1. Create Example

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

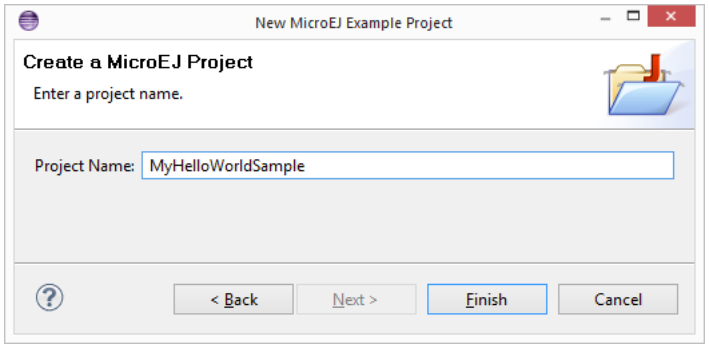
- Select the MicroEJ platform TWRK65F180M-MyPlatform-CM4hardfp_GCC48 from the combo box.
- Select the example `Samples > Getting Started > Hello World`.

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



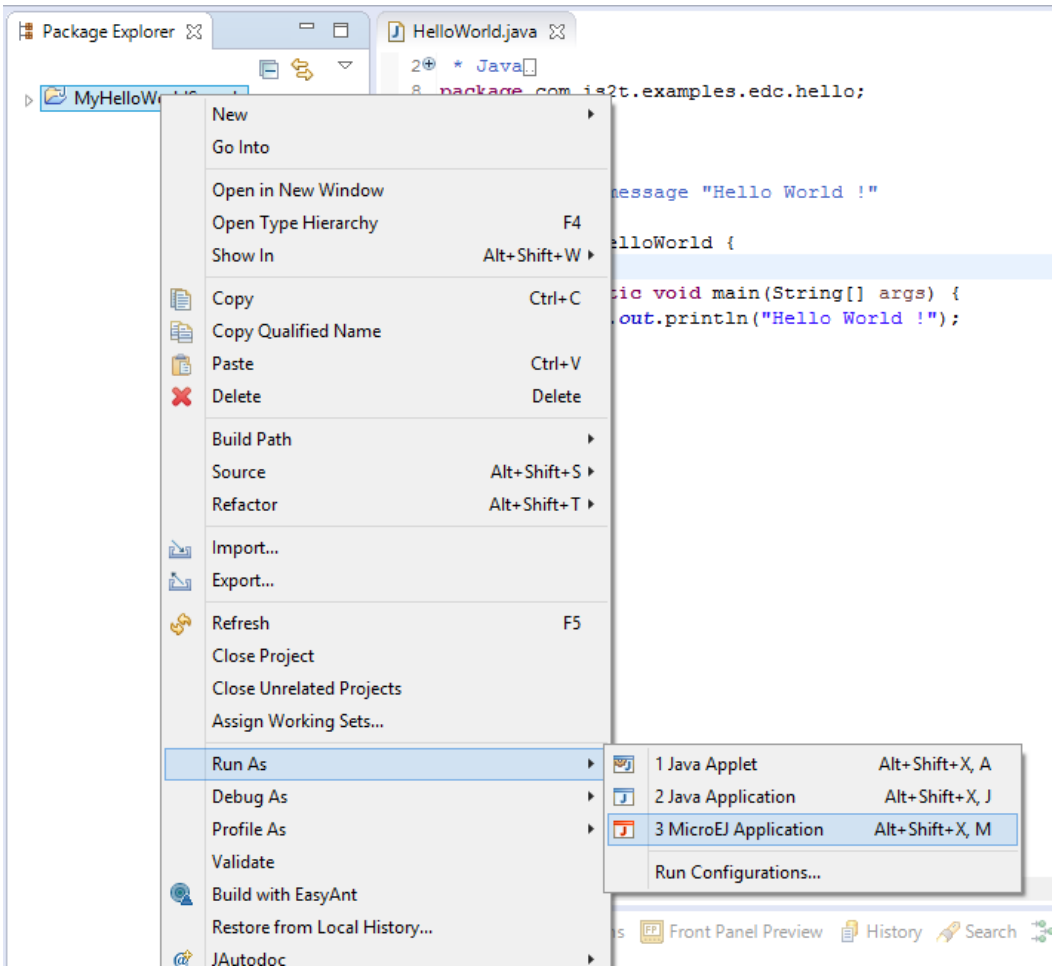
- Click on Finish. The selected example is imported into a project with the given name. The main class (the class which contains the `main()` method) is automatically opened.

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 (TWRK65F180M-MyPlatform-CM4hardfp_GCC48). The result of the test is printed in the console:

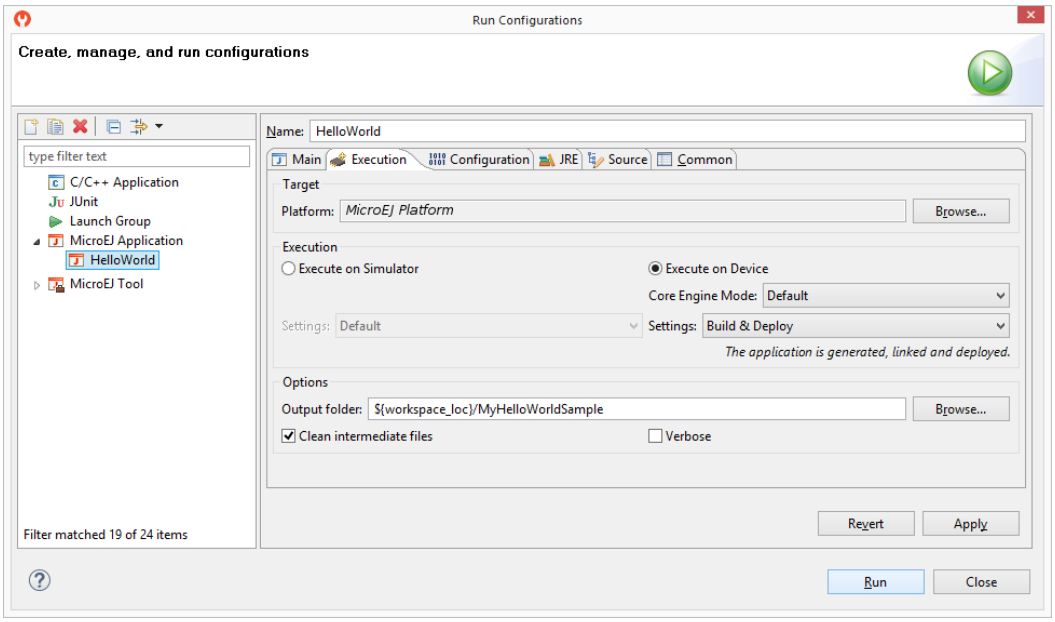
```
Hello World !
```

2.3. Run the Example on the TWR-K65F180M Board

2.3.1. Compile MicroEJ Standalone Application

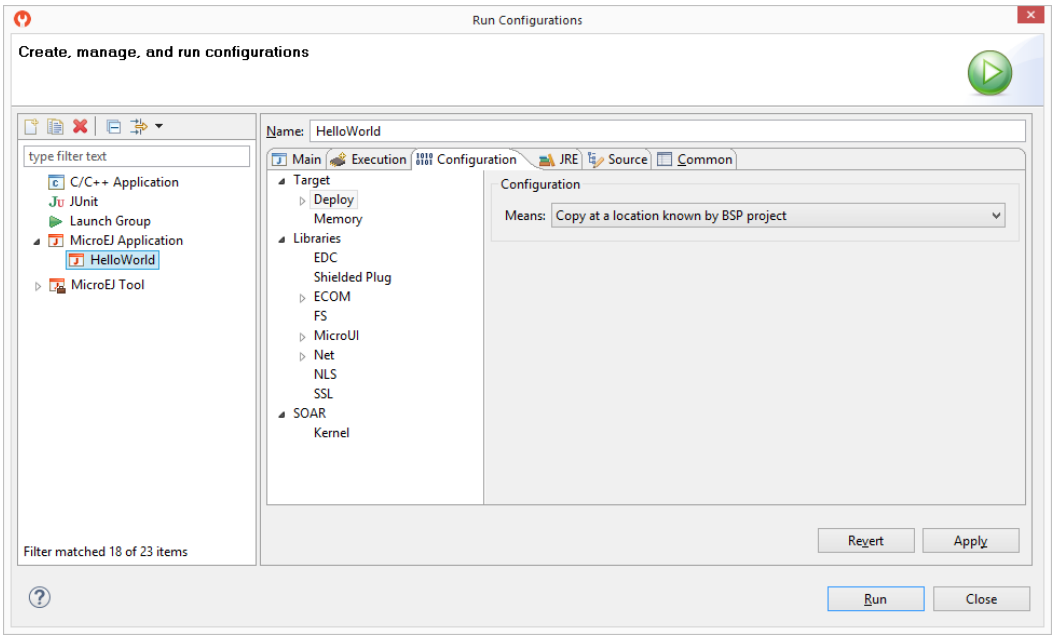
- Open the run dialog (`Run > Run configurations...`).
- Select the MicroEJ Application launcher `HelloWorld`.
- Open `Execution` tab.
- Select `Execution on Device`.

Figure 2.7. Execution on Device



- Open `Configuration` tab and sub menu `Target > Deploy`. Select `Copy at location known by BSP project`

Figure 2.8. Deploy Configuration



- Click Run: the application is compiled, and the compilation result (an ELF file) is copied into a well known location in the workspace. The Freescale Kinetis Design 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 TWR-K65F180M board.

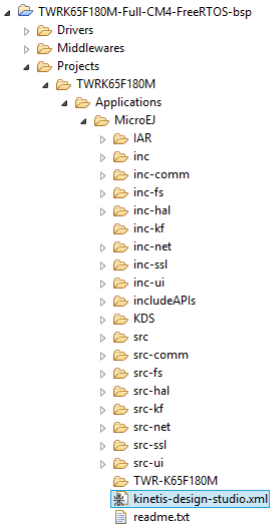
Note

This final step uses Freescale Kinetis Design Studio 3.2.0.

The following steps are performed within MicroEJ.

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

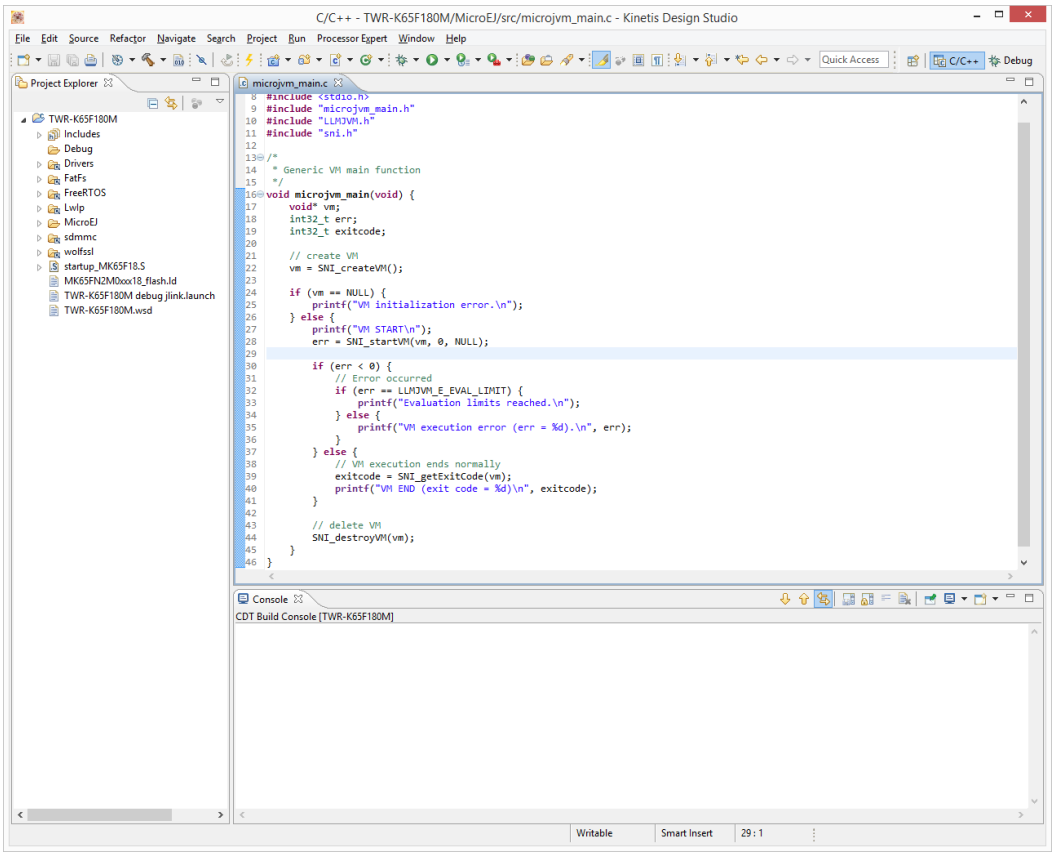
Figure 2.9. Freescale Kinetis Design Studio Project Selection



- Right-click on the file and select `Run as... > Ant build` to launch Freescale Kinetis Design Studio. If Freescale Kinetis Design 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 "kinetis-design-studio.exe" with your kinetis-design-studio.exe file location and retry.

The following steps are performed within Freescale Kinetis Design Studio.

- Figure 2.10. Freescale Kinetis Design Studio IDE



  Build the Freescale Kinetis Design Studio project by clicking on the menu `Project > Build Project`. The project is compiled and linked.

- Deploy the link result on the TWR-K65F180M board by clicking on the menu `Run > Flash from file...`. 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 TWR-K65F180M is based on board support package provided by NXP: KSDK. It includes FreeRTOS, a graphical user interface, a TCP/IP network connection, a file system on SD-Card, a serial connection and some custom GPIOs. MicroEJ platform has been built against Kinetis Design Studio.

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 (MK65FN2M0VMI18)
MCU Clock speed	120 MHz
Internal Flash	2 MB
Internal RAM	256 KB
External RAM	256 MB (SDRAM)

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	6.1.1
UI	7.2.1
Network	5.0.0
File System	2.1.1
HAL	1.0.2

3.3. Platform Output stream

MicroEJ platform uses a COM port as output print stream. The COM port is available via the OpenSDA connector and it is connected to the MCU UART2.

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

Implementation Note

On the TWR-K65F180M, the following parameters can be adjusted:

- Baudrate
- Parity bits
- Stop bits

3.4. RTOS Configuration

MicroEJ platform uses FreeRTOS 8.2.3. 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
LCD	128 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 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: SRAM_L (64 KB)

Section Name	Size
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: SRAM_U (192 KB)

Section Name	Size
Ethernet buffers	15560 KB
Any RW	*n* bytes [a]

[a] *n* depends on MicroEJ application libraries used.

Table 3.6. External RAM: SDRAM (256 MB)

Section Name	Size
Display buffers	300 KB
MicroUI working buffer	3 MB
Multi applications working buffer	3 MB
SSL buffers	128 KB
MicroEJ standalone application heaps	2048 bytes [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: Program Flash (2 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. Multi Applications

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

3.7. Graphical User Interface

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

3.7.1. Display

The display module drives a 320 x 240 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, a copy of pixels data from the back buffer to the frame buffer is performed (stack `copy`).

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: 320
- `height` is the display width in pixels: 240
- `bpp` is the number of bits-per-pixel: 16

The buffers size is `2 * 153600 = 300 KB`.

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 than the LCD buffer (see before).

3.7.2. Inputs

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

Implementation Note

The touch events (*press*, *drag*, *release*) are detected by polling.

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.

3.8. Network

MicroEJ plaform features a network interface. A limited number of 10 sockets could be used for TCP connections, 5 for TCP listening (server) connections and 6 for UDP connections. A DHCP client could be activated to retrieve IP address. All DNS requests could be handled by a MicroEJ software implementation or a native one.

Implementation Note

MicroEJ platform uses LwIP v1.5.0 fetched from git repository of the project. This implementation need a 50 KB internal heap to work. The TCP MSS is 1460 bytes.

The network portage use 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 wait for completion or timeout.

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

3.9. File System

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

Implementation Note

MicroEJ platform uses FatFS R0.11a. The FAT FS driver is the SD driver port of KSDK v2.0.

3.10. Serial Communications

3.10.1. UART Connector

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

- RX: PTA1; available on connector J8, plot 41
- TX: PTA2; available on connector J8, plot 42

Implementation Note

This implementation uses interrupt and relies on the MicroEJ `LLCOMM_BUFFERED_CONNECTION` API. This API is FIFO oriented. It requires two distincts software buffer 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 to connect HAL foundation library. All GPIOs are available either on the elevators connectors (J5 and J6 on the secondary elevator) or on the intern soldering plots (on the primary elevator). Digital pins are implemented by a GPIO access, analog input pin (ADC) is driven by ADC channels of ADC 1 and analog output pins (DAC) use DAC0 and DAC1.

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

1. Using the global MCU declaration: all pins of all ports are grouped under only one virtual port (port 0) and have consecutive values: PTA0 has the ID 0, PTA1, the ID 1, PTA15 the ID 15, PTB0 the ID 32 and so on. For instance pin *PTD11* is accessible by `(0, 139)`. This declaration is useful to target all MCU pins using only one virtual port.

2. Using the standard MCU declaration: PORTA has the ID 1, PORTB the ID 2 etc. Each pin of each port is a value between 0 (PORT*N*-0) to 31 (PORT*N*-31). For instance pin *PTD11* is accessible by `(4, 11)`. This declaration is useful to target a specific MCU pin.

3. Using the physical board connectors. The elevators has 4 connectors: J8, J9, J5 and J6, with respectively these IDs: 64, 65, 66 and 67. For instance pin *PTD11* is accessible on connector J5, pin46: `(65, 46)`. This declaration is useful to target a physical connector pin without knowing which MCU pin it is.

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

Table 3.8. HAL GPIOs Ports and Pins

Port name	HAL port ID	Pins range
Global MCU virtual port	0	0 to 159
MCU port A	1	0 to 31
MCU port B	2	0 to 31
MCU port D	4	0 to 31
MCU port E	5	0 to 31
Board physical port "J6"	67	1 to 80
Board physical port "J8"	64	1 to 80
Board physical port "J9"	65	1 to 80

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

Table 3.9. HAL GPIOs Declaration (port, pin)

Port / Pin	MCU virtual port (1)	MCU port (2)	Board physical port (3)
PTA0	0, 0	1, 0	-
PTA3	0, 3	1, 3	-
PTA9	0, 9	1, 9	64, 11
PTA11	0, 11	1, 11	-
PTA18	0, 18	1, 18	-
PTA19	0, 19	1, 19	-
PTA28	0, 28	1, 28	65, 35
PTA29	0, 29	1, 29	64, 35
PTB19	0, 51	2, 19	65, 72
PTD0	0, 128	4, 0	65, 63
PTD10	0, 138	4, 10	65, 23
PTD11	0, 139	4, 11	65, 46
PTD12	0, 140	4, 12	65, 48
PTD13	0, 141	4, 13	65, 45
PTD14	0, 142	4, 14	65, 44
PTE6	0, 262	5, 6	64, 21
PTE7	0, 263	5, 7	64, 24
PTE8	0, 264	5, 8	64, 59 / 64, 60
PTE9	0, 265	5, 9	64, 58 / 64, 61
PTE10	0, 266	5, 10	64, 25
PTE11	0, 267	5, 11	64, 23
PTE12	0, 268	5, 12	64, 22
PTE18	0, 274	5, 18	64, 8
PTE19	0, 275	5, 19	64, 7
PTE27	0, 283	5, 27	65, 55

Table 3.10. HAL Analog IOs Declaration (port, pin)

Port / Pin	Board physical port	ADC channel	DAC output
DAC0	64, 32	-	0
DAC1	65, 32	-	1
ADC1_SE16	67, 30	16	-

Chapter 4. Board Configuration

TWR-K65F180M provides several connectors, each connector is used by the MicroEJ Core Engine itself or by a foundation library.

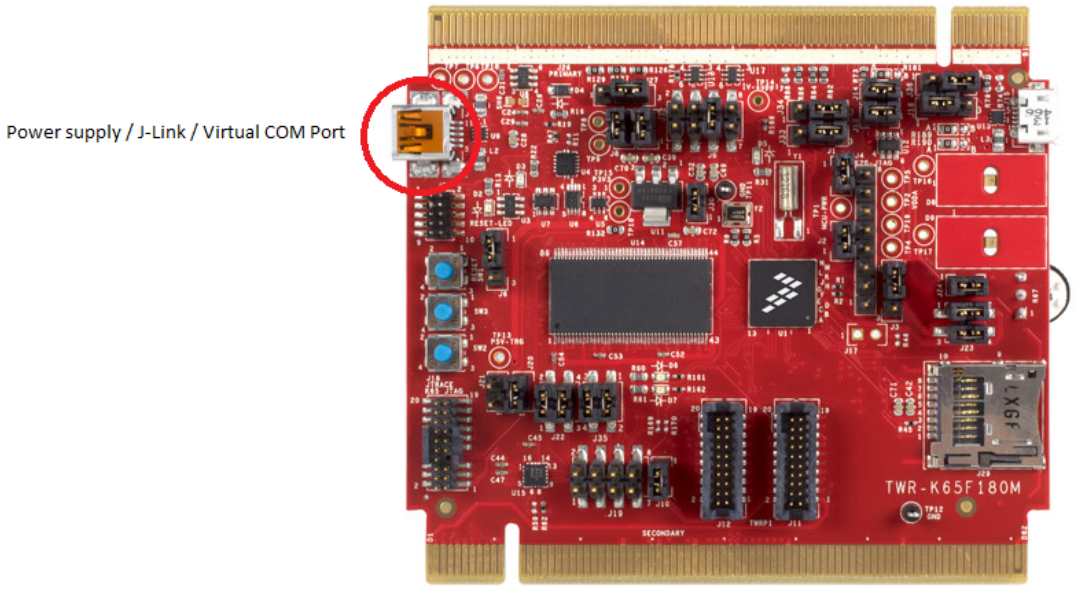
4.1. Mandatory Connectors

TWR-K65F180M provides a multi function USB port used as:

- Power supply connector
- Probe connector
- Virtual COM port

First of all, ensure the jumpers are set to their factory settings. 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 stand-alone application `System.out.print` traces.

Figure 4.1. Mandatory Connectors



4.2. Communication Connectors

TWR-K65F180M provides several communication ports:

- Ethernet
- Serial communication

To enable the Ethernet communications, make sure to have the following jumpers settings on the TWR-SER board:

- *J2 (CLK_SEL)* to position 3-4
- *J3 (CLKIN_SEL)* to position 2-3
- *J12 (ETH_CONFIG)* to position 9-10

Figure 4.2. Communication Connectors - Ethernet

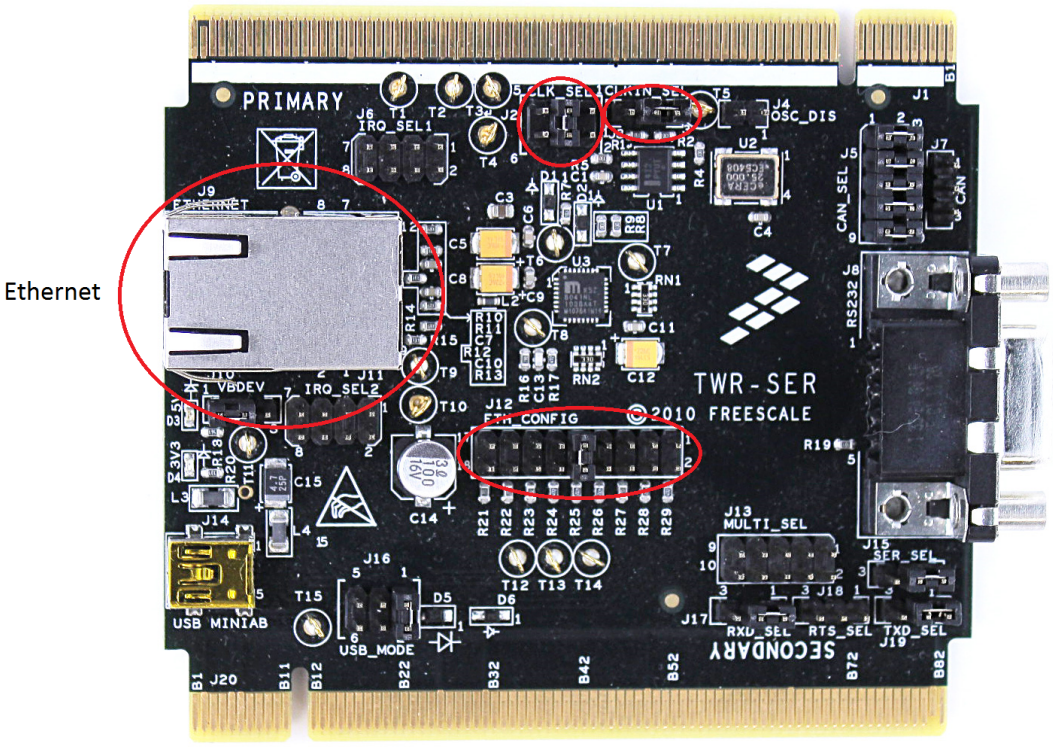
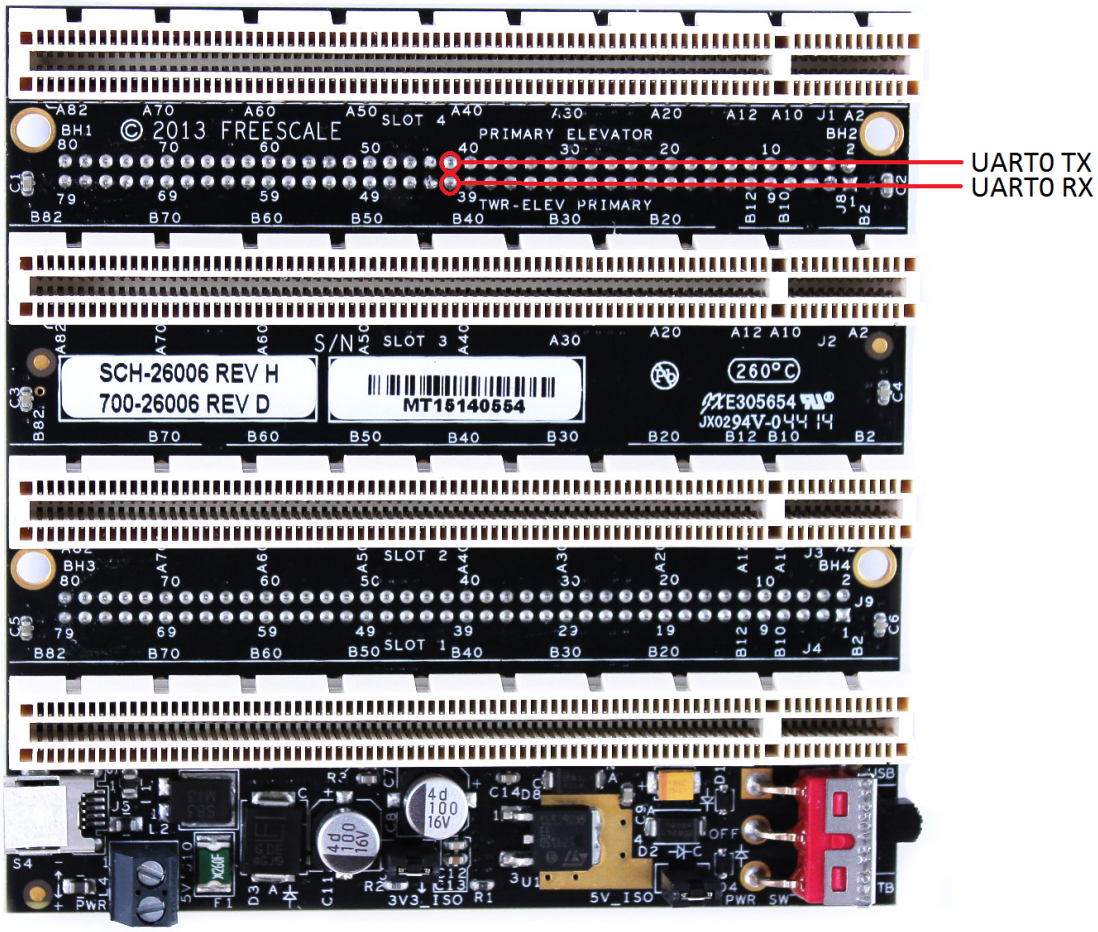


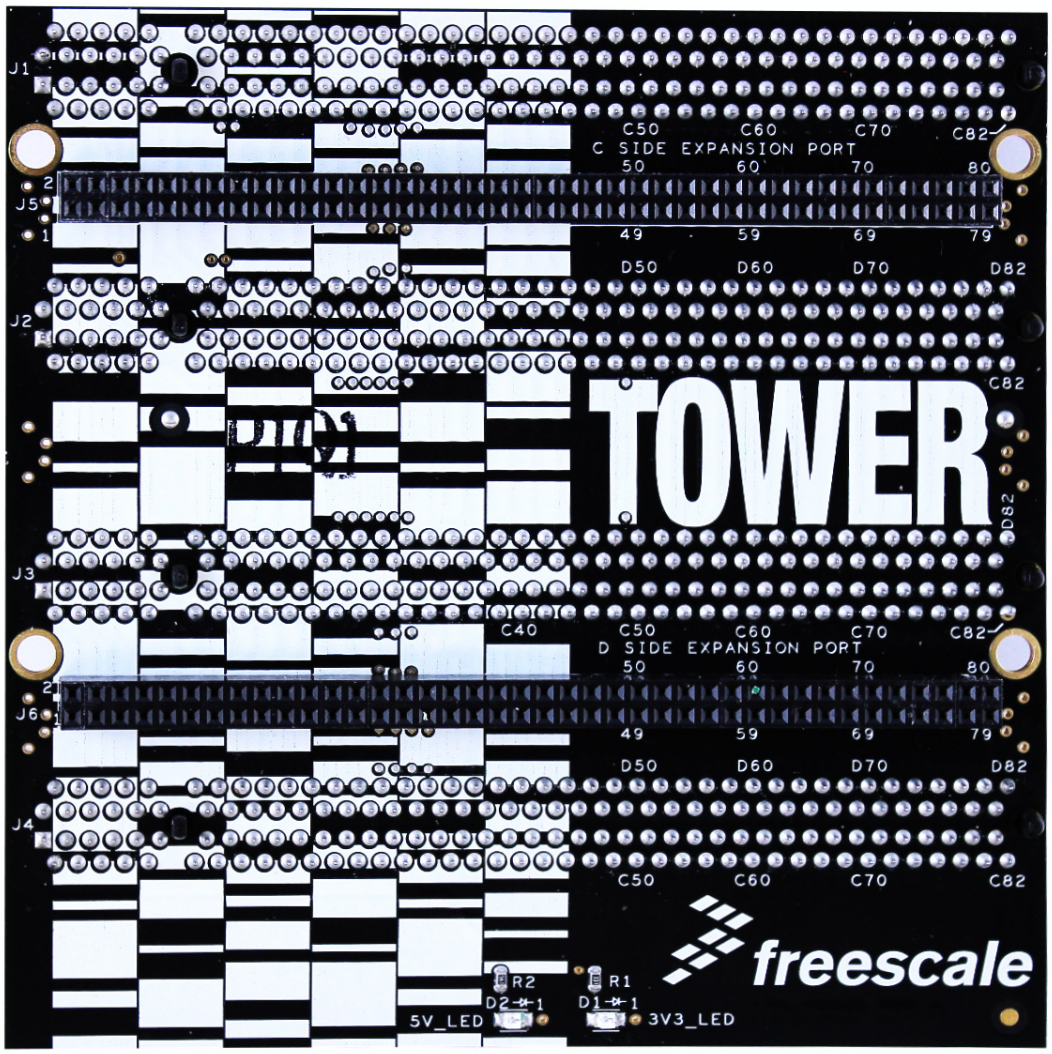
Figure 4.3. Communication Connectors - Serial



4.3. HAL Connectors

TWR-K65F180M provides several HAL GPIOs on connectors J5, J6, J8 and J9 of the TWR-ELEV boards. Since the TWR-LCD board is plugged in the connectors J8 and J9, all the GPIOs normally available through them should be accessed from the inner side of the TWR-ELEV board via the soldering points.

Figure 4.4. HAL Connectors



4.4. UI Setup

TWR-K65F180M provides a LCD display via its TWR-LCD board. This board must be setup to correctly display the MicroEJ standalone application and to relay the touch informations.

First of all, the TWR-LCD board must be flashed with a neutral firmware. This board embeds a MCU to be able to independently drive the LCD screen and use the touch interface and we need to be sure that it won't interfere with the UI operations of MicroEJ. You can get this firmware from the DVD included in the TWR-LCD box:

- Insert the DVD in the disc drive of your computer.
- In a file browser, look for the `TWR_LCD_Demo_Projects.exe` binary in the `fscommand` folder and run it.
- Follow the installation steps and when you are prompted to browse the installed files, click the OK button.

The firmware to use is the file called *JM128_Bootloader.S19*.

To update the firmware of the TWR-LCD board, follow this steps:

- Connect the mini-USB connector to a computer to power the TWR-LCD board. It must be separated from the other components of the TWR-K65F180M. Leaving it attached to its TWR-ELEV board is OK but no other board must be attached to it. If so, make sure to use the USB connector of the TWR-LCD board, not the TWR-ELEV one.
- Hold the *SW4 (BTLD)* button (bottom-right) and press the *SW3 (JMRST)* button (bottom-left). The board should beep and be enumerated as a Mass Storage Device on the host computer. Release the *SW4 (BTLD)* button.
- Copy the firmware file (e.g. *JM128_Bootloader.S19*) to the BOOTLOADER drive. The board will load the .S19 file, beep, and restart. Once restarted, the board is identified and a file named *SUCCESS.TXT* should be present.

The next step is to setup the configuration switches. There is 2 of them: SW1 (bottom-left) and SW5 (top of the reverse side).

Table 4.1. SW1 Configuration Switches

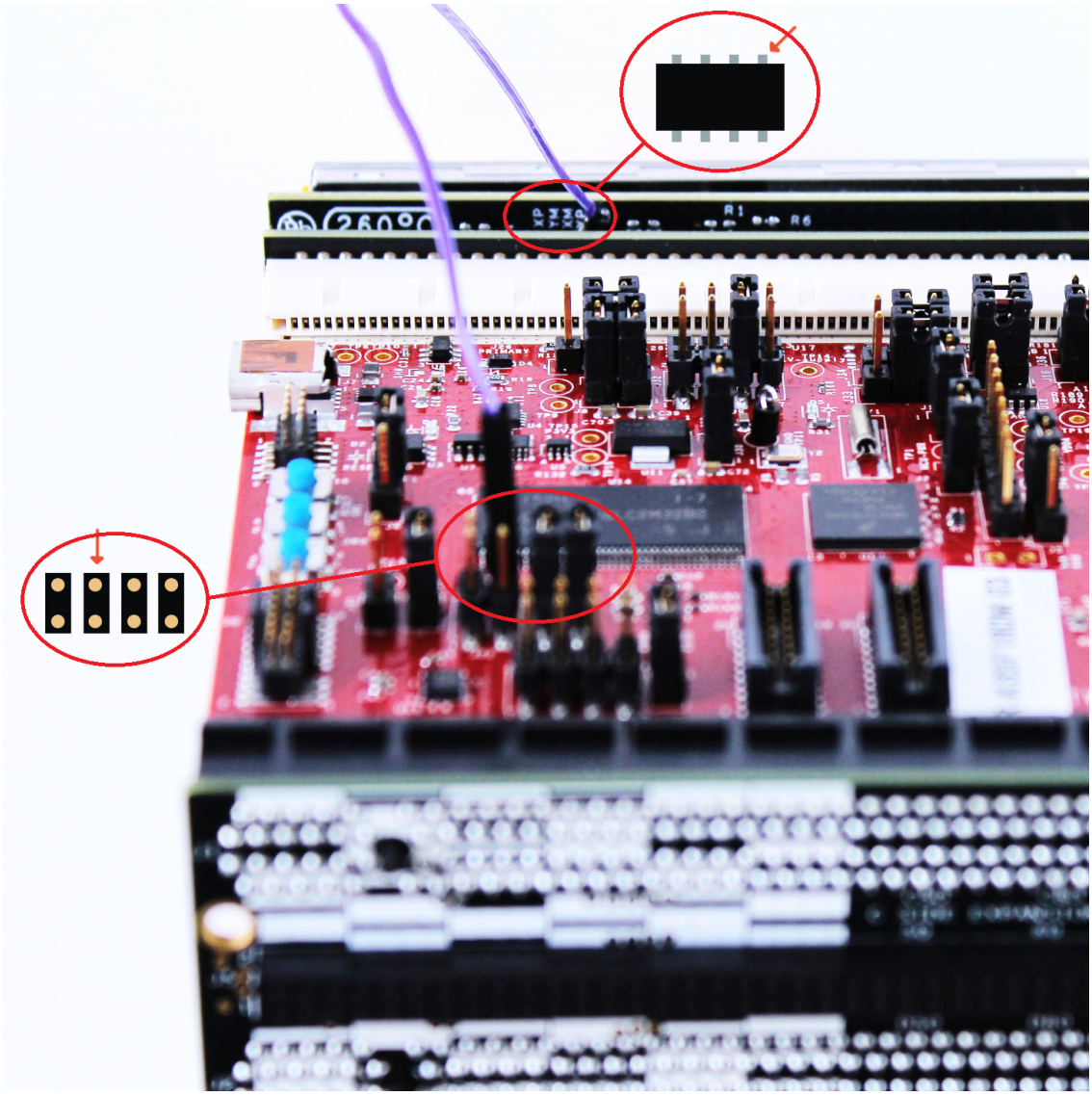
Switch #	Value
1	ON
2	OFF
3	ON
4	OFF
5	ON
6	ON
7	ON
8	OFF

Table 4.2. SW5 Configuration Switches

Switch #	Value
1	ON
2	ON
3	ON
4	OFF

Finally, the *YP* switch (number 4) of the *SW5* jumper needs to be redirected to the pin 4 of the *J22* jumper of the TWR-K65F180M board.

Figure 4.5. Touchscreen Strap



Chapter 5. Freescale Kinetis Design Studio Configuration

5.1. Install Freescale Kinetis Design Studio

This section describes how to install a Kinetis Design Studio from Freescale.

5.1.1. Download Freescale Kinetis Design Studio

- Go to `http://www.nxp.com/products/software-and-tools/run-time-software/kinetis-software-and-tools/ides-for-kinetis-mcus/kinetis-design-studio-integrated-development-environment-ide:KDS_IDE#`.
- Press the `Download` button in the `Downloads` tab. If you're not yet registered, you'll have to do so to continue.
- Download the executable file (e.g. `Kinetis Design Studio Installer for Microsoft Windows 3.2.0.exe`).
- Run executable file and follow installation steps. Install additional software and drivers if proposed. A new application named `Kinetis Design Studio 3 IDE` shall have been created.

5.2. BSP Project Structure

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

- Drivers: all MCU drivers, board drivers and CMSIS drivers
- Middlewares: all 3rd-party files: USB host library, FatFs, FreeRTOS, LwIP, SD/MMC
- Projects: the MicroEJ platform project itself

The KDS BSP project file is `Projects/TWRK65F180M/Applications/MicroEJ/KDS/.project`. This KDS BSP project has been written for Kinetis Design Studio v3.2.0. The project has the following file structure:

- Drivers/*: all MCU drivers, board drivers and CMSIS drivers
- FatFS/*: file system files
- FreeRTOS/*: FreeRTOS files
- LWIP/*: network LWIP files
- MicroEJ/*: all MicroEJ platform implementation files
- sdmmc/*: SD/MMC card driver files

The MicroEJ platform implementation files are grouped by MicroEJ features:

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

Chapter 6. Changelog

6.1. Version 1.1.5

Initial release of the platform.