HEXIWEAR

Firmware User's Guide

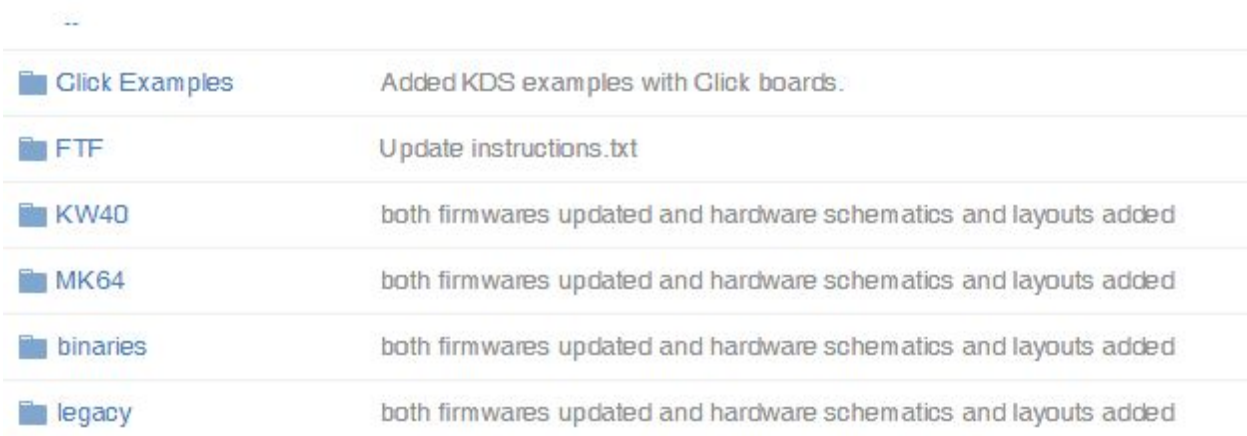
Getting Started

Hexiwear is a small, sleek, low-power device packed with sensors and wireless capabilities, being able to connect both to devices nearby and remote cloud servers.

General information about HEXIWEAR can be found at the link below:
http://docs.mikroe.com/Hexiwear

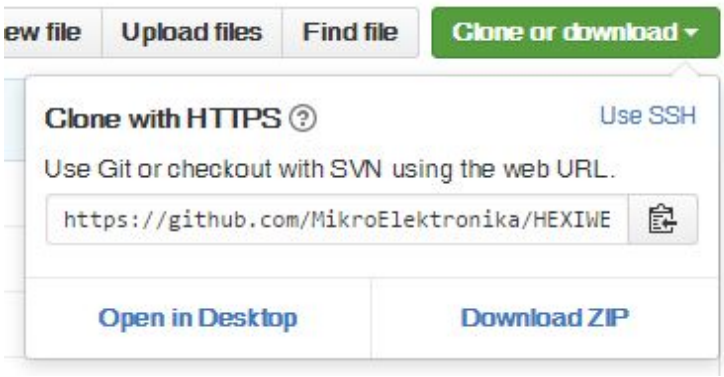
Github's HEXIWEAR public repository is located at the address below:
https://github.com/mikroe/HEXIWEAR

In the SW folder, projects are organized into subfolders according to its purpose.

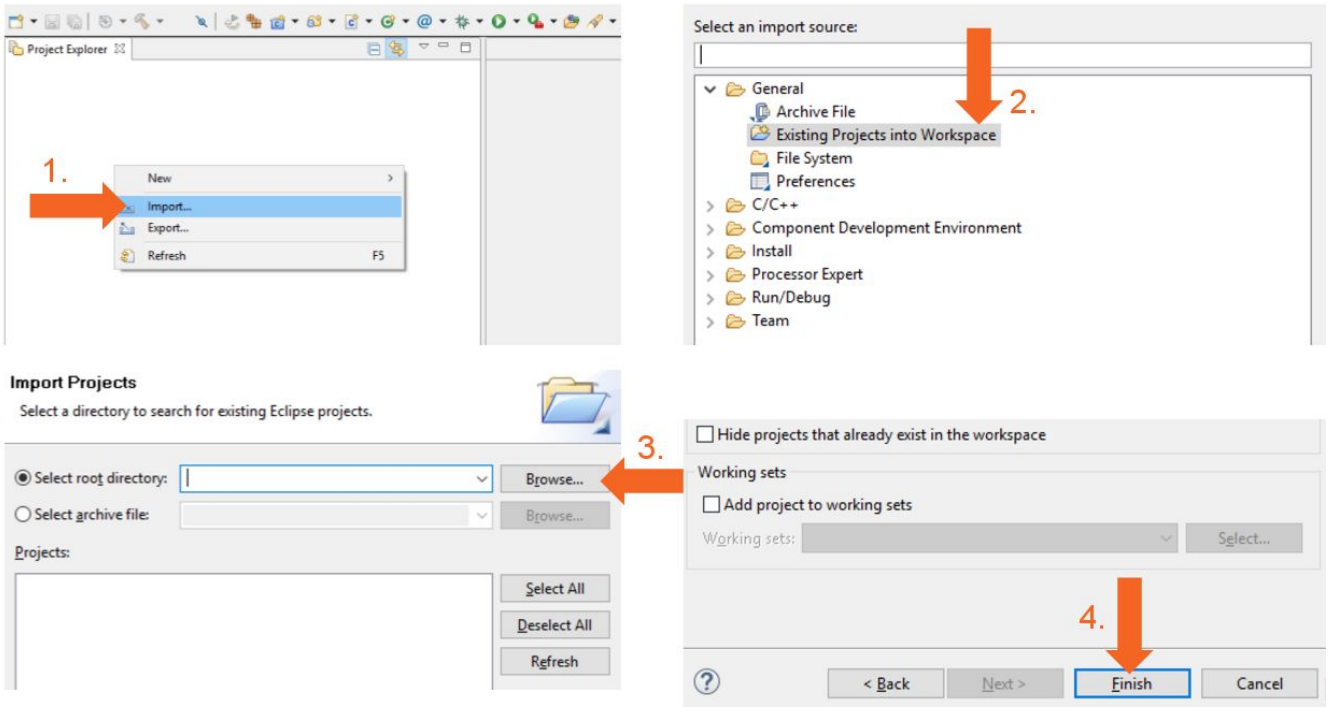


For example, binaries folder contains the binaries for both MCUs. To access the latest HEXIWEAR firmware for the main MCU, open the folder named MK64. There are two projects, one being the bootloader and the other being the main firmware project.

To download projects, you can fork the repository to your own account, clone it or download as a ZIP archive.



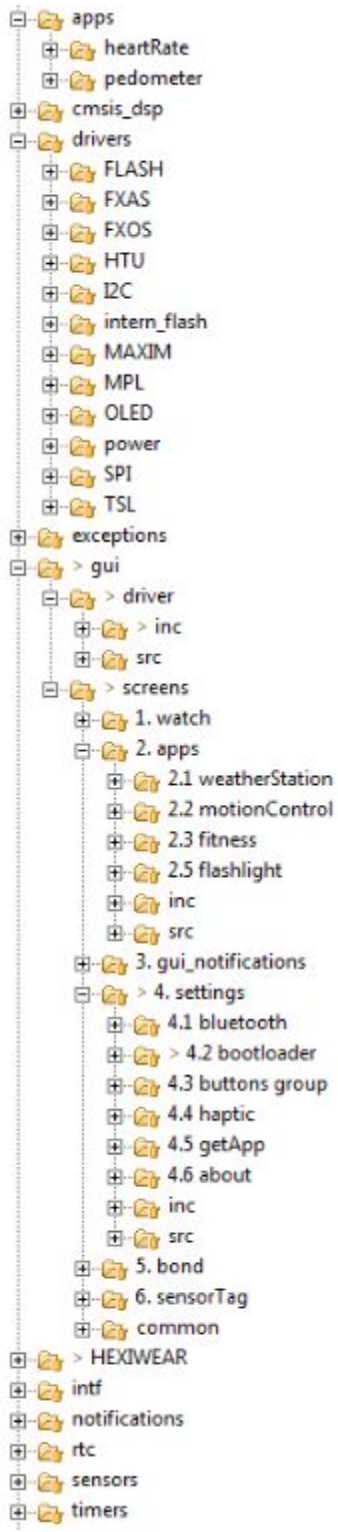
The IDE is Kinetis Design Studio (short: KDS), version 3.2 or newer is recommended. You can download it here. After successful download, import the firmware project following the steps below:



Importing library steps

The general folder structure

The general folder structure is shown here on the right.



Folder ‘apps’ contain the code implementing the logic being used for apps in the HEXIWEAR menu (currently, pedometer and heart rate apps are present).

Folder named ‘drivers’ contains drivers for sensors, external and internal FLASH storage and OLED screen.

The general structure for all the drivers is the same: source files implementing the driver functionality have a "_driver" suffix, while the structures containing sensor information and user settings are located in source files with a "_defs" suffix. Constants and other relevant information are in header files with "_info" suffix. Relevant structure types and enumerations are located in header files with suffix "_types". Drivers rely either on Kinetis SDK drivers, or on the wrappers located in their respective folders, e.g. a particular sensor’s I2C driver is either using KSDK `fsl_i2c_master_driver` or is using a I2C wrapper located in the ‘I2C’ folder.

The sensors present in HEXIWEAR are the following:
- NXP FXAS21002C 3-axis digital gyroscope
- NXP FXOS8700CQ 3-axis digital accelerometer and magnetometer
- NXP MPL3115A2 Precision Altimeter
- TAOS TSL2561 light-to-digital converter
- MEAS HTU21D digital humidity and temperature sensor
- MAXIM MAX30101 heart-rate Sensor

The general graphical interface structure is located under the folder named ‘gui’. All the menus can be found here as separate folders. The hierarchy between screens can also be seen by observing the folder structure: the main HEXIWEAR screen, watch screen, is at the same level as the ’apps’, ’settings’, ‘sensorTag’ and ‘gui_notifications’ folders. These folders represent basic HEXIWEAR menus at their highest level and its subfolders represent separate submenus, respectively.
The submenu folders may contain the appropriate driver files to implement the logic used by an app, if a logic is simple enough (retrieve particular data packets, display them on-screen etc.). For more complicated logic, drivers are located in the ‘apps’ folder and invoked here by appropriate GUI drivers, for example, in case of fitness apps.

Folder named ‘HEXIWEAR’ contains functionality which is not tied to some specific entity in the project and fits on a more global scale. For example, a first task to execute in the system, HEXIWEAR_startup(), responsible for initializing the hardware and the rest of the tasks, is located here.

Folder ‘intf’ contains the interface code for the communication between MK64 and Bluetooth-powered KW40. Interface tasks are defined in `host_mcu_interface_rx.c` and `host_mcu_interface_tx.c`, which are in charge of sending and receiving data-packets.

Folder ‘notifications’ contains the code for displaying the smartphone notifications received from KW40.

Folder ‘rtc’ contains the code for RTC clock functionality, e.g. retrieving current time, programming actions upon RTC alarm event, etc.

Folder ‘sensors’ contains the code for manipulating sensor data and for sending the data to the right targets. This is where I2C communication takes place, mainly for retrieving raw sensor data. All the sensors are using I2C communication. NXP’s sensors are always turned on, while non-NXP’s sensors have power supplies which can be turned on/off and are connected to the different I2C bus.
This functionality is mainly located in the task sensor_GetData(). The sensors are initialized via function sensor_InitModules(). Afterwards, the task is fetching the sensor's data in an infinite loop. All the sensors' data is fetched in its default raw data form and then formatted into a desired format before sending the data-packet. Data type for most of the sensor's data-packets is called mE_t type; mE_t type is an `int16_t` type containing a value with 2 decimal places multiplied by 100, e.g. if the sensor value is 101,35 kPa, it is saved as the int16_t value 10135.
Formatted data values are packed into a data-packet and its types are defined in the `host_mcu_interface.h` header file located in the ‘intf/inc’ folder.
Interrupt callback function `PacketHandler()` implements the state machine for analyzing data-packets and for determining the data-packet type, retrieving data etc. It is implemented to wait on a queue fed with data-packets from different sources.

Folder ‘timers’ contains the code implementing timers’ functionality which is used throughout the project.

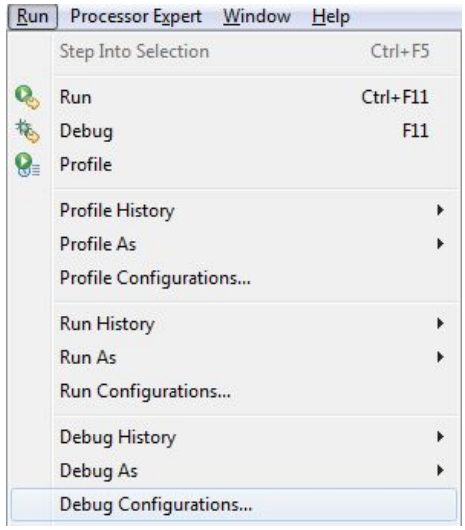
Folder ‘power’ contains the code for HEXIWEAR’s low-power functionality, which is implemented by toggling the OLED screen and sensors’ power supplies on/off, as well as putting the MCU to sleep.

Building the firmware

After importing and tweaking the project, you can rebuild the project by right-clicking on the project’s name in the Project Explorer and choosing the ‘Build Project’ option.

Downloading the firmware to HEXIWEAR

There are a couple of options for programming and debugging the firmware. In case of programming and debugging the firmware, open the “Debug Configurations…” menu and choose your preferred type of configuration, e.g. “GDB OpenOCD Debugging” configuration. Right click on it and choose “New” to create a new configuration. This will create a new configuration and fill almost all necessary fields for your debugging session. It’s usually required to manually fill in the Config options; this can vary from one debug configuration to another.
For debugging, OpenOCD, Segger J-link and PEMicro GDB debugging configurations are directly supported from within KDS.

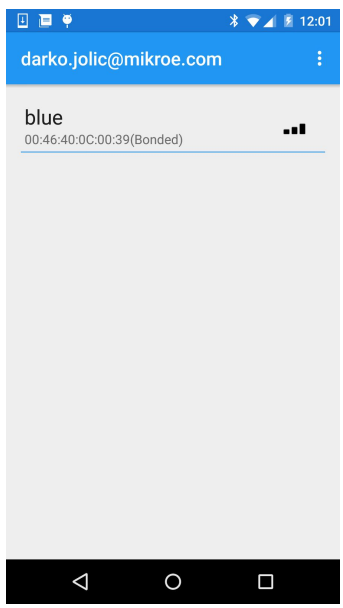


There are several options for programming HEXIWEAR.

OTAP

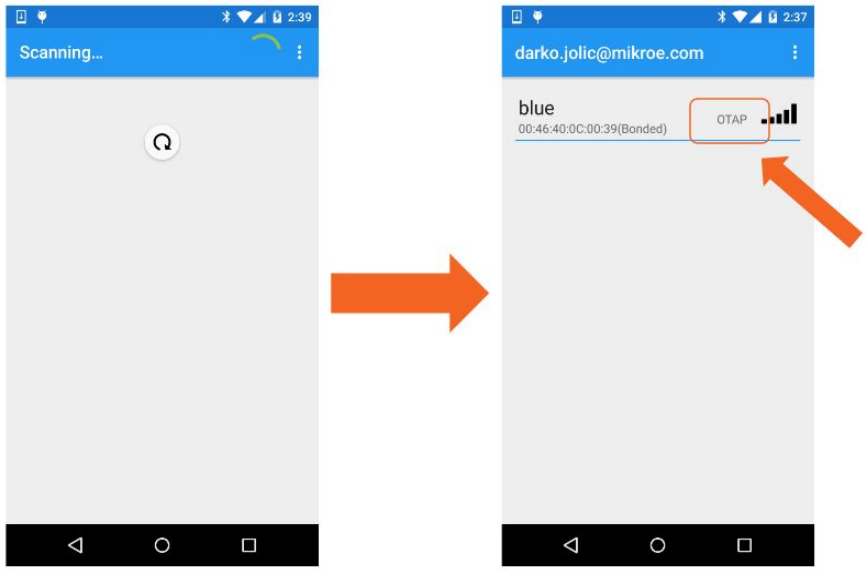
Over-The-Air-Programming is available via Android and iPhone applications; it works by transferring the firmware to HEXIWEAR via BLE protocol.
Before starting OTAP, it’s necessary for HEXIWEAR to be bonded and connected to the smart device. Be sure to update the app so it includes the latest HEXIWEAR firmware. To run OTAP, follow the next procedure.
The steps for OTAP are the following:

1. Enable Bluetooth in your device options.
2. Make sure that HEXIWEAR is connected and bonded.
3. Run HEXIWEAR app afterwards and you should see your HEXIWEAR device on-screen, but don’t select your HEXIWEAR device yet. Default name for HEXIWEAR device is HEXIWEAR before activating the device.

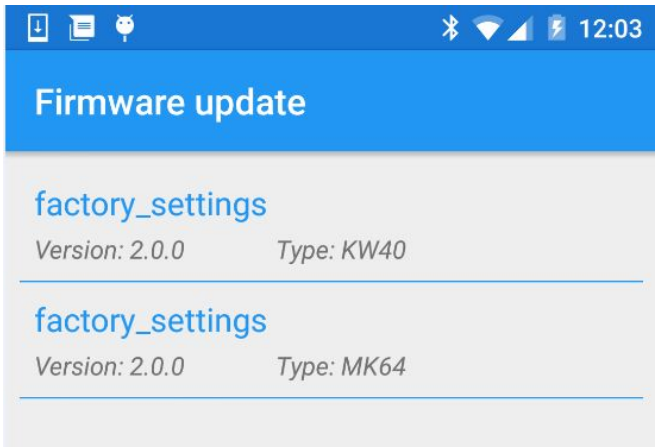


4. Activate the OTAP procedure on HEXIWEAR (image below)

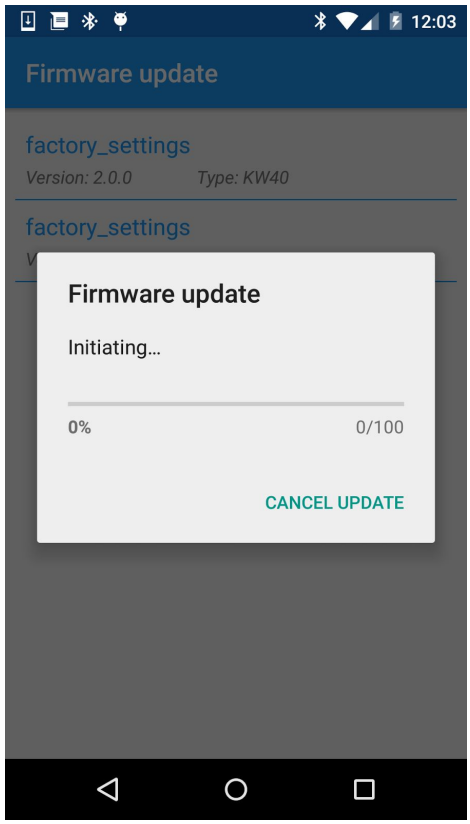
Refresh the device screen and the device name will appear with “OTAP” suffix, at which point you should select it.



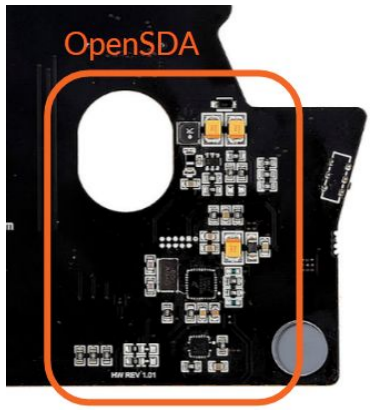
A screen will appear asking you to choose which MCU firmware you’d like to update.



5. The progress bar will appear and the transfer process will start, at which point you should wait for it to finish and for HEXIWEAR to reboot itself upon successful flashing of the new image.



OpenSDAv2



You can use the on-board OpenSDAv2 programmer at the docking station to download the firmware to HEXIWEAR.
To communicate with the programmer, use pyOCD (**recommended**) or OpenOCD gdb server in conjunction with gdb client within KDS.

Below are the steps for both options:

1. pyOCD (**recommended**): For pyOCD debugging session, download the pyOCD archive from HEXIWEAR's Github repository and do the following:

   a) Install Python 2.7.11. (Run python-2.7.11.amd64.installer.)

   b) Launch KDS -> go to Help menu -> Install New Software, click on the Add button and write in the following:

   Name: GNU ARM Eclipse Plug-ins

   Location: http://gnuarmeclipse.sourceforge.net/updates

   Complete the installation and restart KDS afterwards.

   c) Copy the executable pyocd-gdbserver.exe into the KDS root folder.

   d) Go to "Debug Configurations…" window and create the new PyOCD instance.

   e) Under Debugger tab, in Executable field, enter the path to the GDB-PyOCD executable (e.g. C:\nxp\KDS_3.2.0\pyocd-gdbserver.exe)

   f) Under Debugger tab, in Other options field, enter "-dinfo" (with no quotes)

   g) Press Apply and Debug, then wait for the firmware to be downloaded and for the debugging session to begin.

2. OpenOCD: For OpenOCD debugging session, do the following:

   a) create the new OpenOCD instance in "Debug Configurations…" window.

   b) Type "-f kinetis.cfg" into Config Options field under Debugger tab.

   c) Press Apply and Debug, then wait for the firmware to be downloaded and for the debugging session to begin.

   Please note that programming with OpenOCD isn't recommended for projects of sizes bigger than 512kB, such as HEXIWEAR project. For smaller projects, it should be fine.

Custom programmer

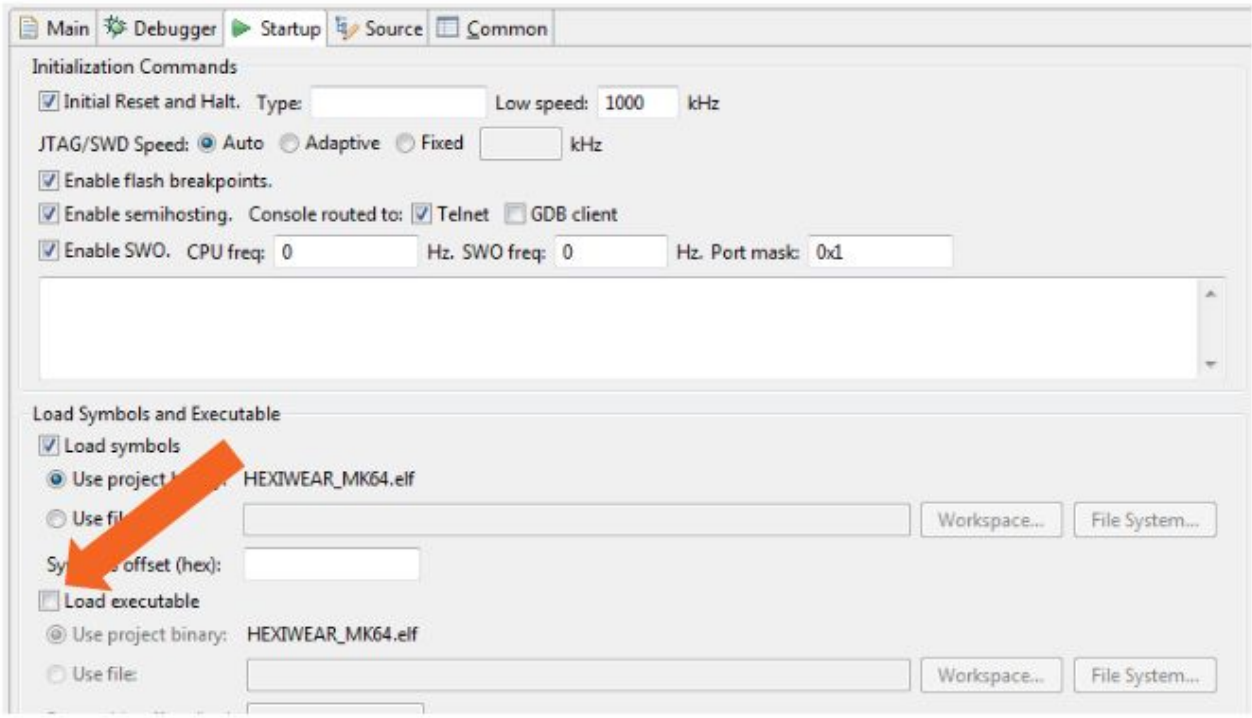
Use your own programmer to connect to HEXIWEAR by using the JTAG port on the docking station

To program HEXIWEAR, use above-mentioned debug interfaces within KDS or use the tool provided by programmer's manufacturer. For example, in case of a J-link debug probe, you can use J-flash application to program both MCUs on HEXIWEAR.

To debug HEXIWEAR, use some of the available debug interfaces, For example, in case of a J-link debug probe:

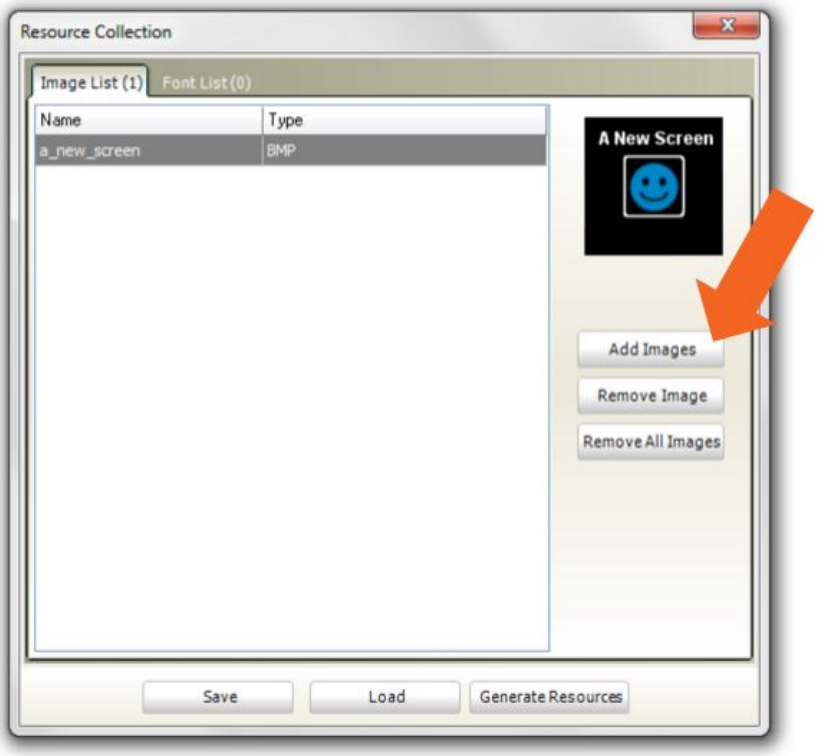
1. Create the new J-Link instance in the "Debug Configurations…" window.
2. Set the Device Name field to be "MK64FN1M0xxx12" (without quotes) under Debugger tab
3. Press Apply and Debug, then wait for the firmware to be downloaded and for the debugging session to begin.

In either case (pyOCD or OpenOCD), if you'd like to skip programming and jump right to debugging, deselect the checkbox "Load executable" under Startup tab.



Resource Collection Tool

Resource Collection window is designed as a helpful tool that will let users to easily add and organize used images and fonts. Using this tool, users can simply include desired images and fonts and use them in their projects by generating bitmap images in a suitable format.



Currently, only 16-bit bitmap images are supported with image size up to 96 x 96.

To insert an image to the current project, users should click on the Add Images button. After the image has been added, it will be populated in the images list, as on the picture below. If the user wishes to remove certain or all images that were previously added to the list, he should click on the Remove Image or Remove All Images button, respectively.

Once you have added necessary images and fonts, generate the code from the Visual TFT and the inserted image and fonts will be contained in the appropriate resource file.