Mercury System

Getting Started Guide

IoT and Connectivity Made Simple

Francesco Ficili
17/11/2018

Revision Log				
Author	**Date**	**Major**	**Minor**	**Description**
Francesco Ficili	17/11/2018	1	0	Initial Release.
Francesco Ficili	02/02/2019	1	1	Minor corrections.

SUMMARY

1. INTRODUCTION 4
2. WHAT THE MERCURY SYSTEM IS 5
3. WHAT YOU CAN DO WITH MERCURY SYSTEM 7
4. HW COMPONENTS 8
The Base Board (BB) 8
The Modem Boards (MB) 10
The Slave Boards (SB) 11
The Expansion Boards (EB) 14
The Power Boards (PB) 16
The Brain-Less Boards (BL) 17
5. THE MERCURY SYSTEM FRAMEWORK 18
The Framework 18
The Framework Functionalities 19
6. MSF INSTALLATION 20
Framework Installation 20
Other tools 22
7. CREATE A NEW PROJECT 23
8. PROGRAM THE BASE BOARD 26
Program the Base Board with the USB bootloader 26
Program the Base Board with an external programmer 28
9. EXAMPLES 30
BT Example 30
WiFi Example 31

1. Introduction

This manual provides a getting started guide for the implementation of embedded applications using Mercury System. The manual provides a first introduction to the HW and SW components of the system, hints about tools installations, projects creations and programming and some getting started examples as well.

2. What the Mercury System is

The Mercury System (MS in short) is a modular system for the development of connectivity and IoT applications. The system uses various type of electronic boards (logic unit, modems, slave boards equipped with sensors and actuators, power boards...) and a complete SW framework to allow the realization of complex applications. Scalability, ease of use and modularity are key factors and are allowed by the use of a heterogeneous set of components that allow to assemble the system like a construction made with LEGO© bricks.

The board set which composes the system is made up by the following "families":

- **Base Board (BB):** It's the "brain" of the system and contains the main logic unit as well as different communication buses and connectors to interfaces the slaves. It also contains a simple power supply system and a recharge unit for a single LiPo cell (it can satisfy the power requirements of simpler systems). It can exist in different variants, depending on the employed microcontroller unit.
- **Modem Board (MB):** it's the board that allows network connectivity. It can exist in different variant, depending on the network interface (GSM/GPRS, Wi-Fi, BT, Radio…). It's interfaced to the Base Board with a dedicated communication line.
- **Power Board (PB):** it's the board that allows to satisfy the particular power requirements of the system, when it's necessary. They can vary, depending on the particular power requirement to satisfy (high power, solar harvesting, piezo harvesting, etc.).
- **Slave Board (SB):** these are the system's peripherals, and they vary depending on the specific mounted sensor or actuator. Typical examples are SBs with relays, temperature sensors, RGB LED controllers, servo controllers, accelerometers, etc. They communicate with the BB with I2C or UART channels and a dedicated command set.
- **Expansion Board (EB):** these are the boards that allow planar connection of Mercury System. There are variants which can contains Displays, battery socket, etc.
- **Brain-Less Board (BL):** these are the controller-less boards. They in general contain really simple sensors or actuators that don't need the bus interface. There are meant as an alternative to slave boards for cost-sensitive applications.

Slave Boards and Modem Boards are provided pre-programmed with a FW which implements a dedicated command set for a high-level management of the board, while the Base Boards are provided with a SW framework which encapsulates all the low-level services (operative system, device drivers, system services, etc.), leaving to the user the development of application level logic only. Moreover, the Base Board comes with an USB bootloader, so it can be programmed without the need of an external programmer.

Figure 1 shows a typical system connection (only BB, MB and SBs):

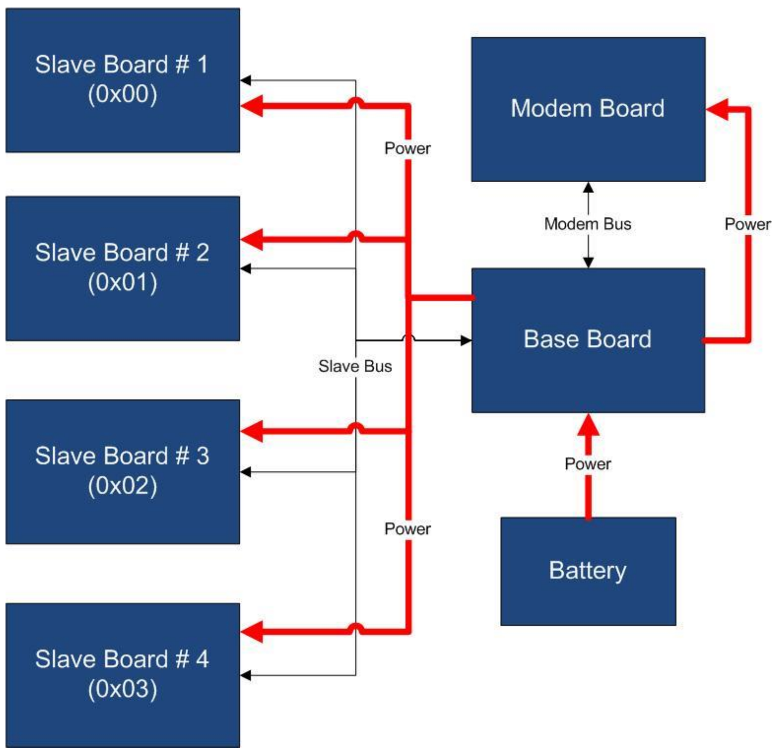


Figure 1 - Typical System Connection

Examples of application fields of MS are:

- Home automation System,
- IoT applications,
- Connectivity Applications,
- Monitoring and control Systems,
- Remote Control,
- Industrial Process control,
- Robotics applications,
- Test benches,
- Etc…

3. What you can do with Mercury System

The MS is an HW/SW development system specifically designed to simplify complex application development, in particular for those applications that deal with IoT (Internet of Things) and Connectivity contents.

The system simplifies the development of such kind of applications with four main characteristics:

1. **High Modularity:** by providing an high variety of Modem Boards and Slave Boards the system allows to incorporate several different type of sensors/actuators and to connect to an high variety of connectivity and IoT channels (WiFi, BT, GSM/GPRS, LoRa, Radio, etc.).

2. **Intelligent Slaves:** by placing the Slaves on buses the MS provides an almost infinite level of scalability (you can add as many as slaves you want, without worrying about pin collision and uC resources) and, at the same time, simplifies the interfacing of complex sensors and actuators, by providing a virtualized and uniform way to access the slave devices. This is a key factor in many applications.

3. **Modular Power Management:** with the Power Boards the MS provides a simple and robust way to address the power management problem, especially for connected and IoT devices.

4. **Development Framework:** by providing a SW Framework the MS simplifies the application SW development task and provides simple API and SW stacks for the handling of wireless connectivity channels (WiFi, BT, GSM/GPRS, etc.).

The above characteristics are key factors to simplify and speed up the development of the embedded side of your connectivity and IoT applications.

You can find some examples of developed applications on our You Tube channel:

- Bluetooth Moodlamp: https://www.youtube.com/watch?v=Co3H3hVcO5o
- GSM/GPRS Intrusion detector: https://www.youtube.com/watch?v=JjRrdgCXYrM
- IoT Meteo Station: https://www.youtube.com/watch?v=HAflxVgQlsk&feature=youtu.be
- Bluetooth Controlled Rover: https://www.youtube.com/watch?v=ULyeNoNSruA
- Irrigation System with BT interface: https://www.youtube.com/watch?v=ipqQ0NbGfG0

4. HW components

As we saw in the previous sections, the Mercury System comprises a quite heterogeneous set of HW components, divided into families; the following sections go deeper in details in the description of the MS board families.

The Base Board (BB)

The Base Board (BB) is the Mercury System's main component, as it is the user programmable element and it provides all the interfaces to the rest of the system. The purpose of the Base Board is to act as system master, interface the Modem Board and Slave Boards and runs the system user application. Moreover, the Base Board provides a simple power supply system (up to 1A) and single LiPo cell recharge unit, in order to satisfy the power requirement of small applications, even if battery powered.

The Base Board contains two main interface connectors:

- The Standard Mercury Connector,
- The Mercury Modem Connector.

By using these two connectors the Base Board can be interfaced with all the MS Slave Board, expansion Board, Brain-Less Board and Power Board (Mercury Connector) and to the Modem Board (Mercury Modem Connector).

The Base Board is equipped with the following main components:

- A microcontroller with internal Bootloader (USB programmable),
- A power supply circuit with LiPo battery recharge unit,
- A full-speed USB interface with USB mini connector,
- An on-board non-volatile EEPROM memory,
- An internal RTCC (Real-Time clock and calendar) fed with a low-power oscillator,
- Three on-board user LEDs and one user Button,
- A Standard Mercury Connector (with I2C, UART, externa interrupts lines and up to 11 I/Os),
- A Mercury Modem Connector,
- A main ON/OFF switch.

Figure 2 depicts an example of Base Board (BB110):

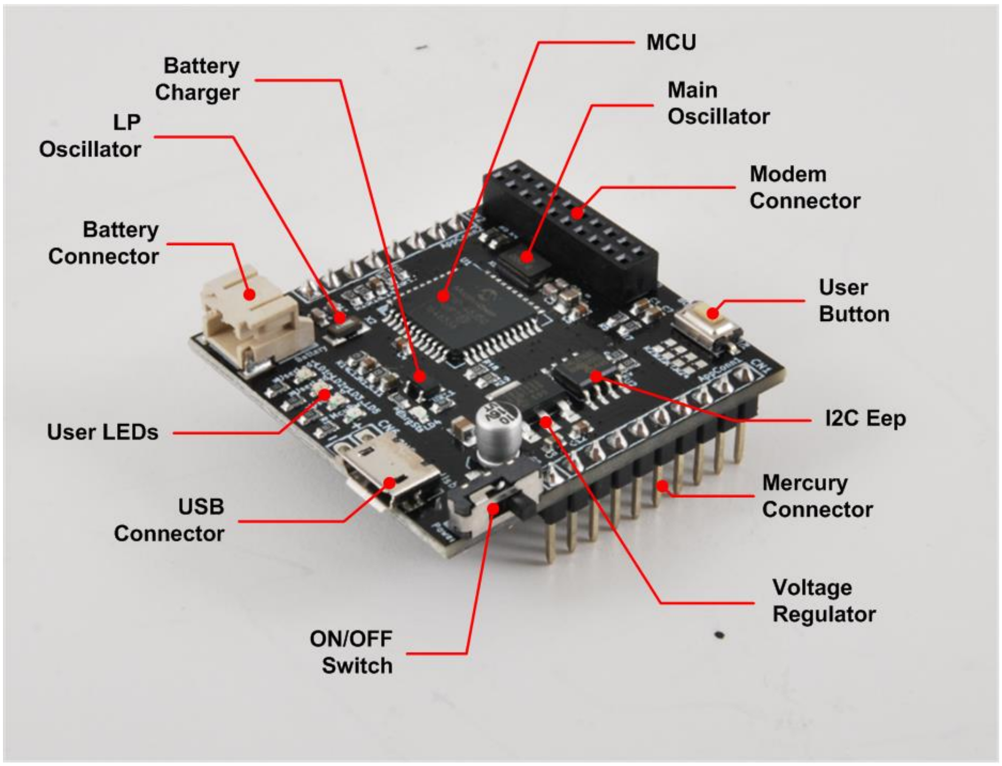


Figure 2 - MS BB110

The Modem Boards (MB)

The Mercury System foresees several Modem Board that enable communication to networks (locals or internet) and/or to control devices (Smartphones, Tablets, PCs, etc.), in order to allow the development of connected and IoT applications.

The Modem Boards are interfaced directly to the Base Board with a dedicated communication channel, and may contain several different types of interfaces (e.g. WiFi, BT, Radio, etc.). The support to emerging communication technologies can be easily added by the introduction of a dedicated Modem Board, so an high level of scalability is ensured.

Some examples of Modem Boards are:

- WiFi Modem Board,
- Bluetooth Modem Board,
- GSM/GPRS Modem Board,
- LoRa Modem Board,
- 433 MHz Radio Modem Board,
- ZigBee Modem Board,
- MiWi Modem Board,

Figure 3 depicts an example of Modem Board with BT interface (MB310):

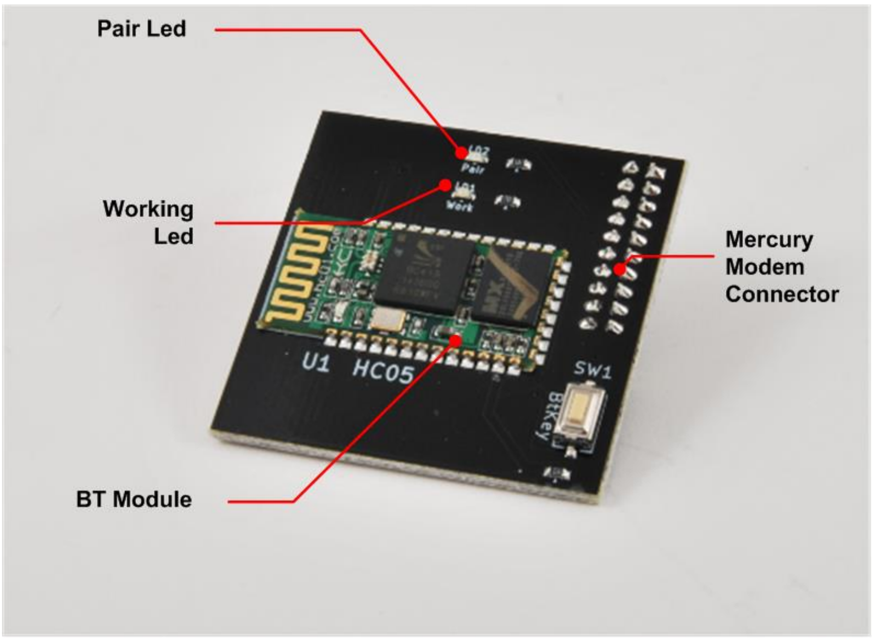


Figure 3 - MS MB310

The Slave Boards (SB)

In order to develop applications, the Mercury System provides several different Slave Board, each of them containing a specific peripheral (could be a particular sensor, an actuator, a communication device, etc.), managed by a local controller.

The MS Slave Boards layout is standardized, in order to simplify the interfacing with the Base Board and ensure and high level of modularity and scalability. Each Slave Board is provided with a I2C (Inter Integrated Circuit) communication line and a four position dip-switch to dynamically set the bus address of the Slave Board. Addresses from 0x01 to 0x0F are available for the Slaves, while address 0x00 is reserved for broadcast communication. In this way up to 15 devices can be connected to the Base Board using the dynamic addressing scheme. This number can be even increased by reprogramming the Slave with a software provided address. Moreover, two open-collector digital lines connected to the Base Board external interrupts are provided for the Slave Boards that need to provide async interrupts. Moreover, slave Boards that need an higher bandwidth and peer-to-peer communication can be interfaced using an additional UART channel.

Figure 4 provides an example of Slave Boards interface:

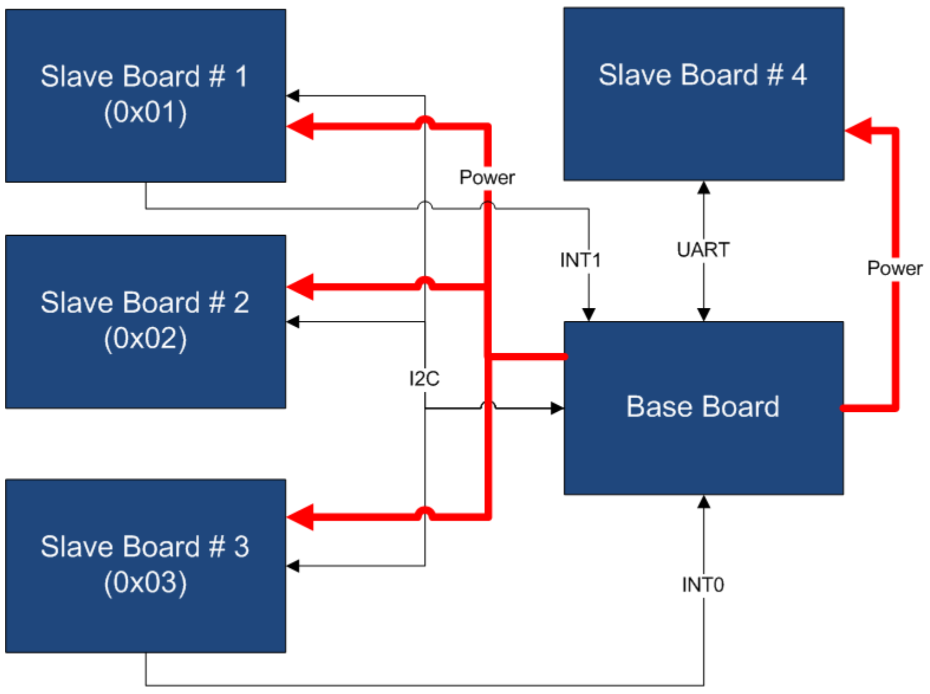


Figure 4 - Example of SBs Interfacing

There are several sub-families of Slave Boards:

- Sensor SBs,
- Actuator SBs,
- Communication SBs,
- Interface SBs,
- Special SBs.

Table 1 provides some examples for each sub-family:

Sub-Families	Examples
Sensor SBs	Ultrasonic, Infrared, Temperature&Humidity, PIR, Gas Sensor, Ari Quality, Thermocouple, Moisture, Analog Input, Accelerometer.
Actuator SBs	Relay, High-Side Driver, Low-Side Driver, Servo, DC Motor, Stepper Motor, Neopixel.
Communication SBs	RS232, RS485, CAN, LIN, Ethernet.
Interface SBs	OLED Display, Keypad, Mini-Joystick.
Special SBs	SD Card, MP3 Decoder.

Table 1 - Slave Board Sub-families

Figure 5, Figure 6 and Figure 7 provide some example of Slave Boards:

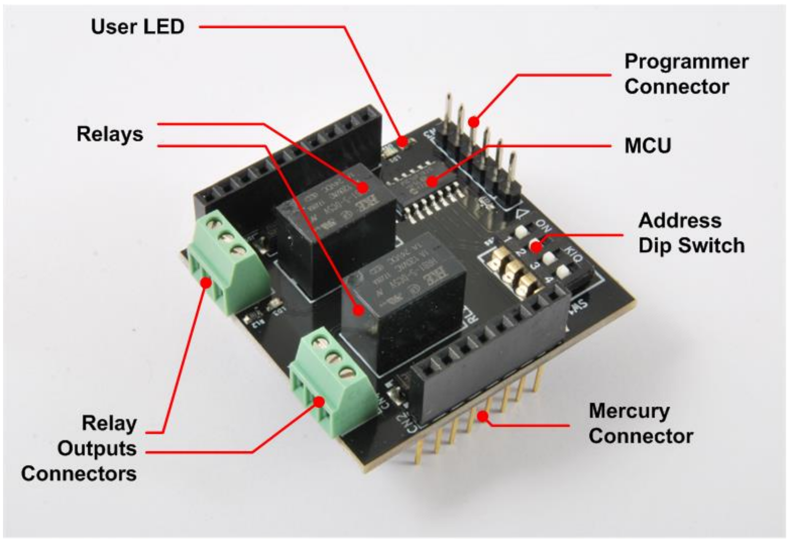


Figure 5 - SB110 - Relay Slave Board

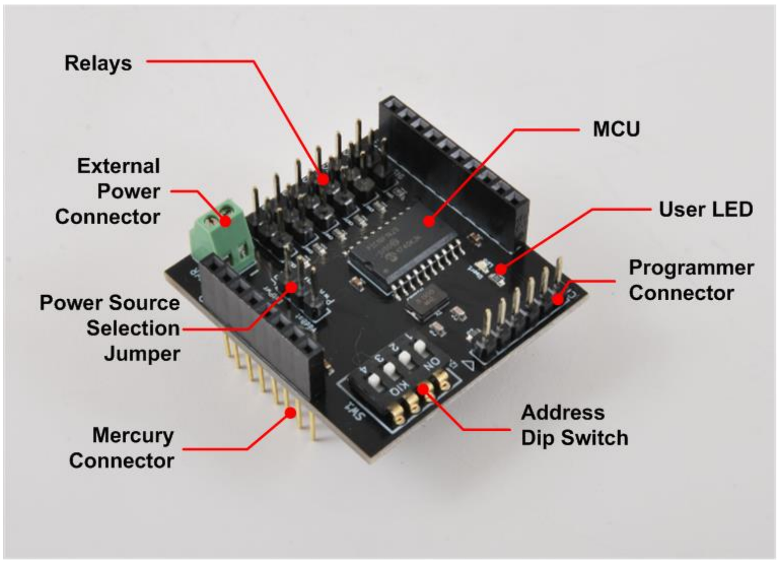


Figure 6 - SB130 - Servo Control Slave Board

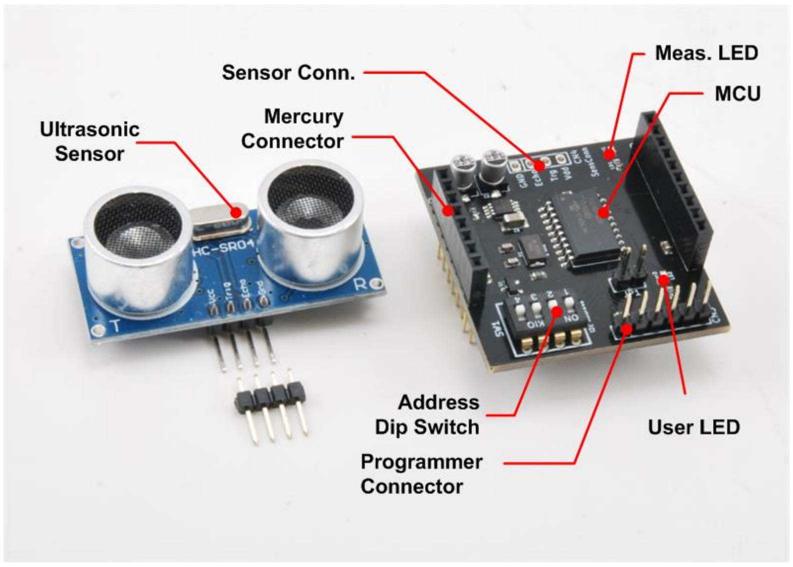


Figure 7 - SB310 - Ultrasonic Sensor Slave Board

The Expansion Boards (EB)

As an application with Mercury System can be composed by several different board, an alternative way to interconnect the various Boards, apart from the classic “stackable” Arduino-like way has been introduced. This alternative way is achieved by the usage of so-called expansion boards, that are Boards with various Mercury Connector sockets interconnected between each other, in order to allow a “planar” interconnection scheme. Moreover, these boards have additional connectors to interconnect to other EBs as well and may have additional features, like integrated displays or battery holders.

Figure 8 and Figure 9 depicts two examples of expansion board, an expansion with two slots and an expansion with two slots and an integrated alphanumeric 16x2 display.

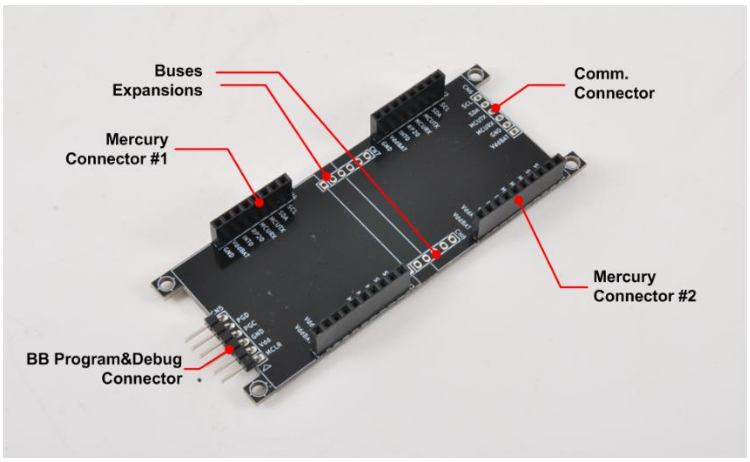


Figure 8 - EB110 - Expansion Dual Slot

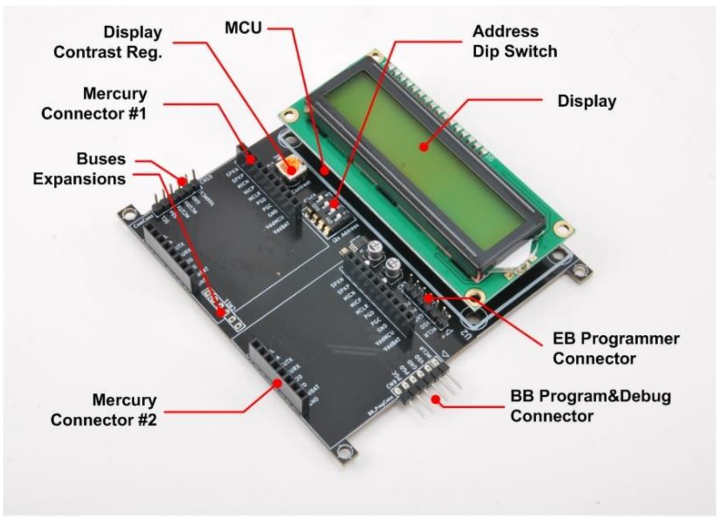


Figure 9 - EB210 - Expansion with 16x2 LCD display

Figure 8 and Figure 9 depicts two examples of boards interconnection using the EB110.

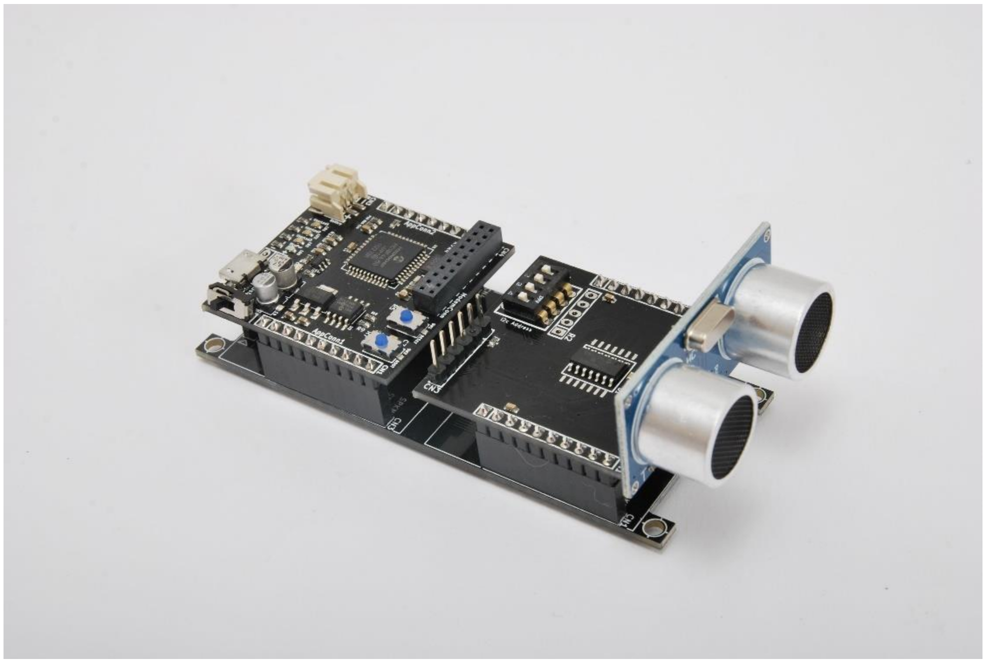


Figure 10 - Example of Expansion with BB and SB310

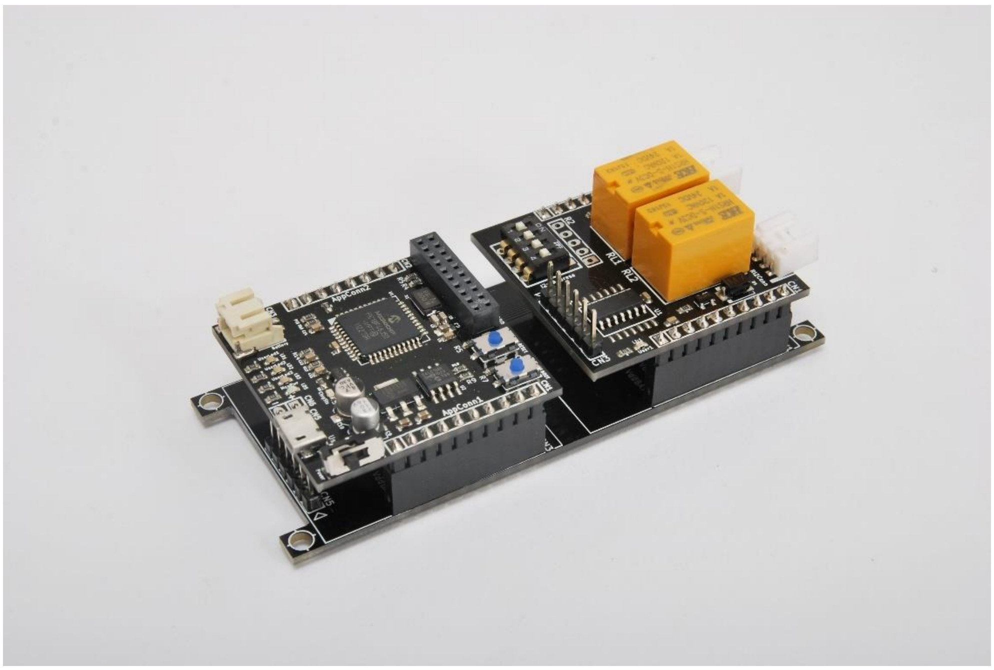


Figure 11 - Example of Expansion with BB and SB110

The Power Boards (PB)

Despite the Base Boards are equipped with a simple power supply system and a LiPo battery recharge system, this is not enough to address possible applications power requirements, thus another family of board has been introduced: the Power Boards (PB).

These boards provide the Mercury System with the possibility to have the same scalability the system has for Slaves and Modems to power.

Some examples of Power Boards are:

- Power Boards for current up to 5A,
- Power Boards for current up to 10A,
- Power Boards with Solar Energy Harvester,
- Power Boards with Piezo Energy Harvester.

The Brain-Less Boards (BL)

The Brain-Less Boards are basically Slave Boards without a local controller in it (Brain-less). They are intended to be used for simple and cost-sensitive application, where simple sensors or actuators are used.

5. The Mercury System Framework

The Framework

The Mercury System Framework (MSF) is a layered SW framework specifically designed to support application development with Mercury System. It provides to the user a complete set of base functionalities to easily interface MS Slave Boards (SB) and Modem Boards (MB) as well as some infrastructural and system services. Figure 12 shows the layered Architecture of the MSF.

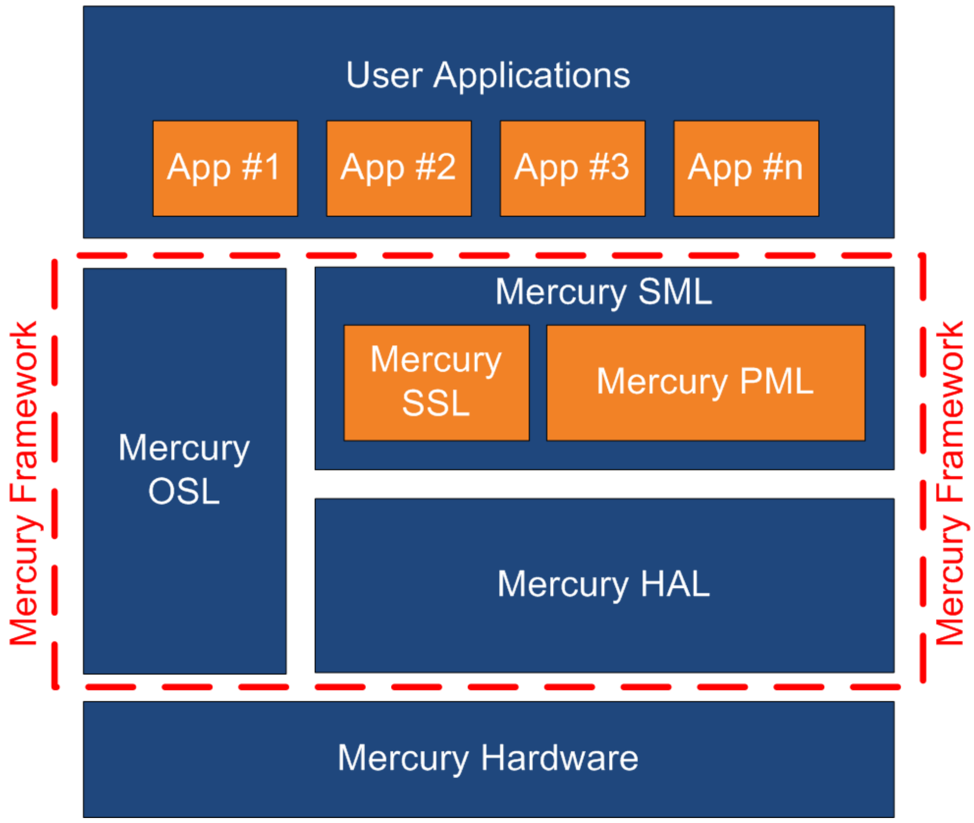


Figure 12 - Mercury System Framework Architecture

The framework is made up by the following components:

HAL (Hardware Abstraction Layer): the purpose of this layer is to abstract the HW dependencies to the upper layers.

SML (System Management Layer): the purpose of this layer is to provide services for the management of communication buses (I2C, UART) and for the management of Mercury System's Modem Board (WiFi, BT, GSM/GPRS). It also provides a set of System Services, like System Power Management, RTCC, USB terminal, etc. It's divided in two main components:

- PML: Peripheral Management Layer,
- SSL: System Services Layer.

OSL (Operative System Layer): this layer is made up by a lightweight RTOS that provides basic services to the system, like scheduling tables for the various tasks, Events, SW Timers, Alarms, etc…

The Framework Functionalities

The Mercury System Framework provide a broad set of functionalities that helps the user in the developing of applications. The management of all buses and Modem communication stacks is provided along with services for the handling of the most useful microcontroller internal peripheral (RTCC, ADC, USB, Power Management, etc.). Moreover, a simple real time OS implementation with services likes schedule tables, SW timers, alarms, etc. is provided.

The user has to implement only the high-level application logic and schedule a period function to implement his own application:

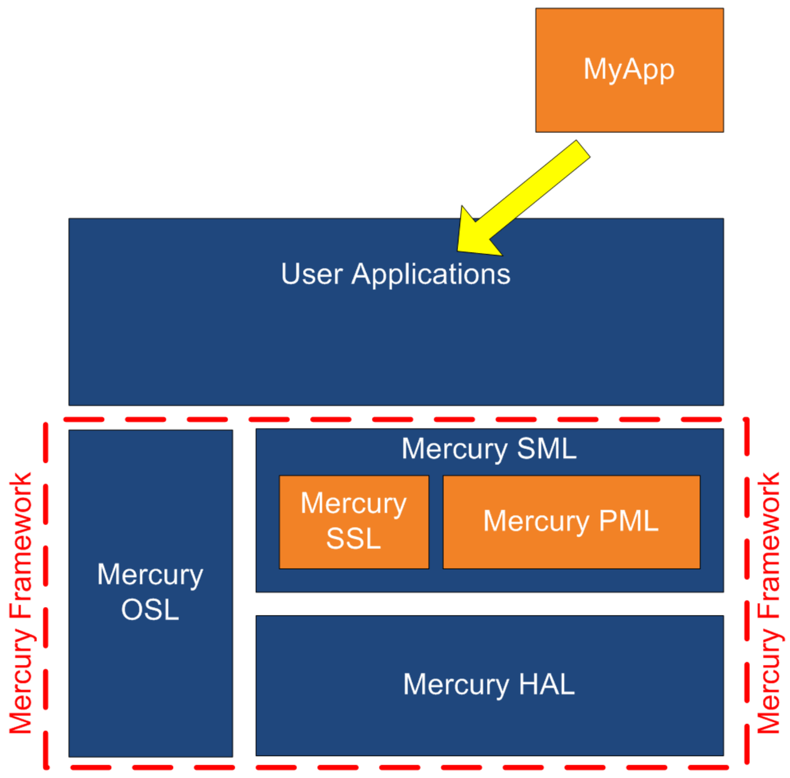
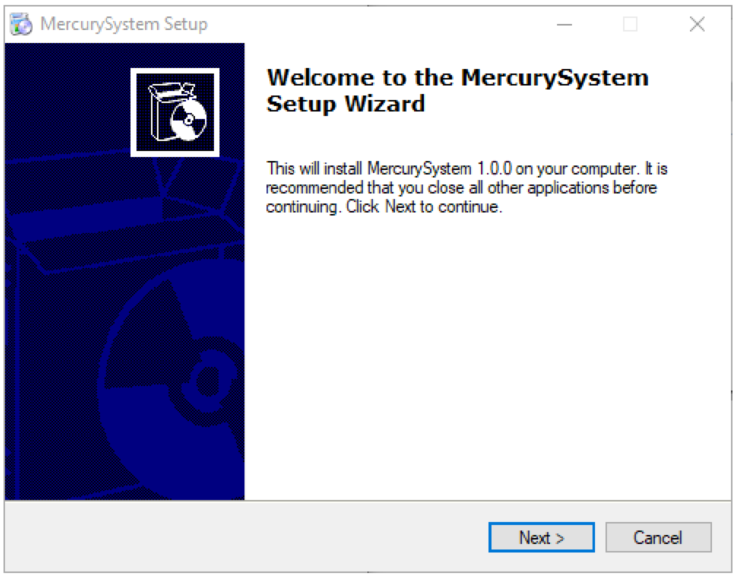


Figure 13 - Example of user application positioning inside the MSF

6. MSF Installation

Framework Installation

To install the MSF simply run the file **MSF_Setup_vX_Y_Z.exe**, where X, Y and Z is the framework version. Once the installation wizard starts you will see the welcome window depicted on Figure 14,

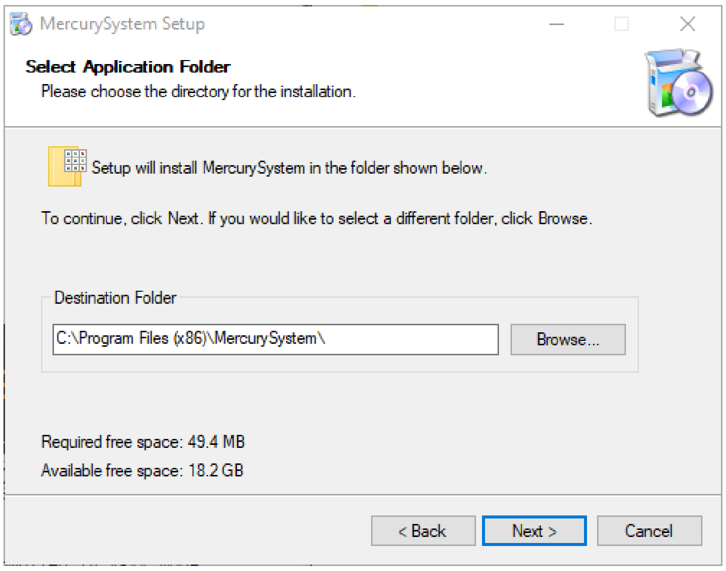

MercurySystem Setup

Welcome to the MercurySystem Setup Wizard

This will install MercurySystem 1.0.0 on your computer. It is recommended that you close all other applications before continuing. Click Next to continue.

Figure 14 - Installation Wizard Welcome Window

Click on "Next" to continue. In the next window you can choose the installation path of the framework (Figure 15).


MercurySystem Setup

Select Application Folder
Please choose the directory for the installation.

Setup will install MercurySystem in the folder shown below.

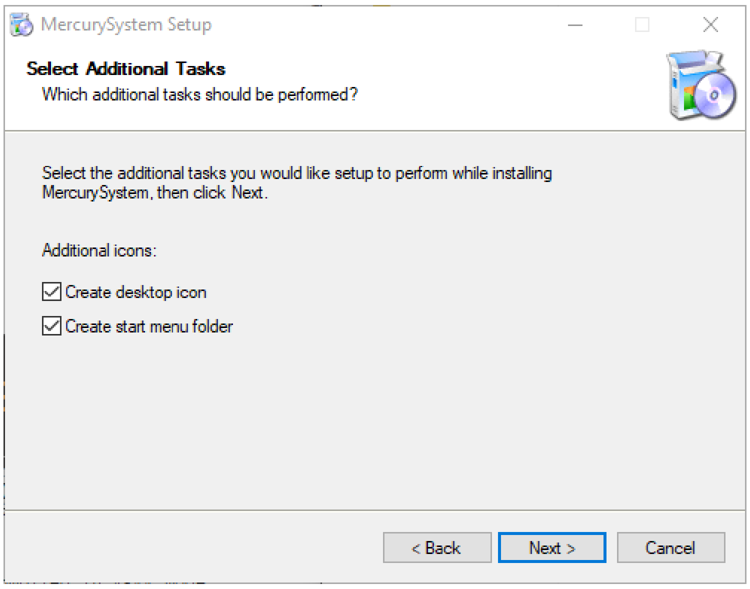
To continue, click Next. If you would like to select a different folder, click Browse.

Destination Folder
C:\Program Files (x86)\MercurySystem\

Required free space: 49.4 MB
Available free space: 18.2 GB

Figure 15 - Path selection Window

Once you are happy with path selection click “Next” to continue and the window depicted on Figure 16, that allows to choose if you want a desktop and a start menu icon, will appear.


MercurySystem Setup

Select Additional Tasks
Which additional tasks should be performed?

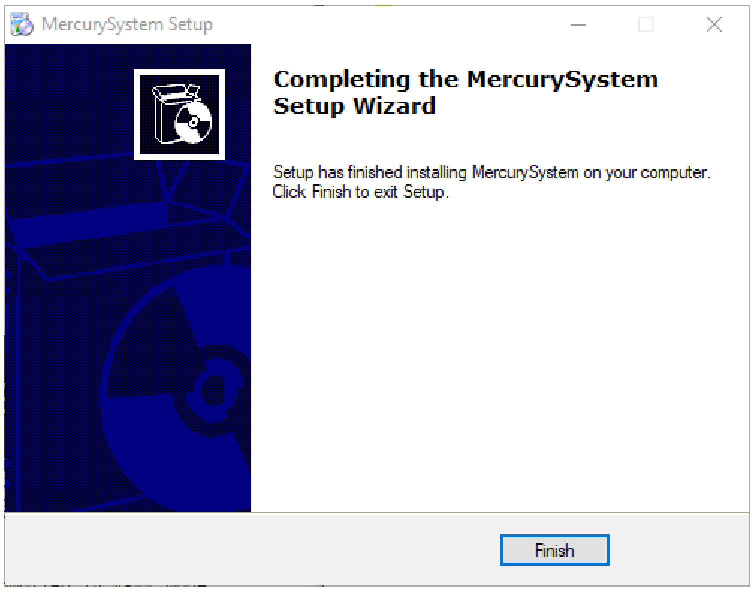
Select the additional tasks you would like setup to perform while installing MercurySystem, then click Next.

Additional icons:

☑ Create desktop icon
☑ Create start menu folder

Figure 16 - Icon creation Window

Once you click “Next” the framework installation will start and once completed you will see the window on Figure 17 that confirms the installation process is completed.


MercurySystem Setup

Completing the MercurySystem Setup Wizard

Setup has finished installing MercurySystem on your computer. Click Finish to exit Setup.

Figure 17 - Installation complete Window

If you then look at your installation directory you should see something similar to what is depicted in Figure 18:

Documentation 12/11/2018 20:59
MercuryFwk 12/11/2018 20:59
MercuryProjects 12/11/2018 20:59
Tools 12/11/2018 20:59

Figure 18 - MSF installation folder

The content of the folders is described below:

- **Documentation:** this folder contains the framework documentation as well as the available Mercury System HW datasheets.
- **MercuryFwk:** this folder contains the Framework source code.
- **MercuryProjects:** this folder is the default location of Mercury System projects made using the framework. The folder contains some demo projects as well.
- **Tools:** this folder contains useful development tools, like the USB bootloader host application.

Other tools

In order to develop application with Mercury System using the MSF the user need to install the official Microchip Technology Toolchain. The MSF has been developed for the use with the following tools:

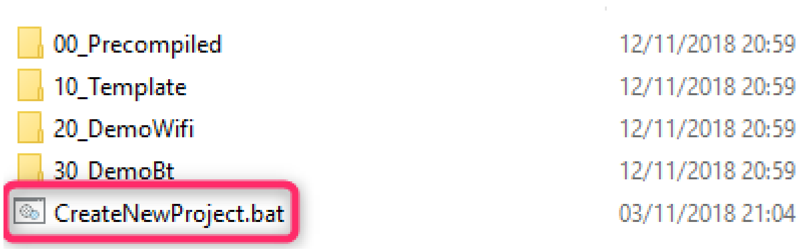
Tool Type	Tool Name
Integrated Development Environment (IDE)	MPLab X IDE
Compiler	XC8 C compiler

Table 2 - MSF toolchain

In order to know the specific tool version associated to a specific MSF release, please check the MSF Release Notes.

7. Create a new project

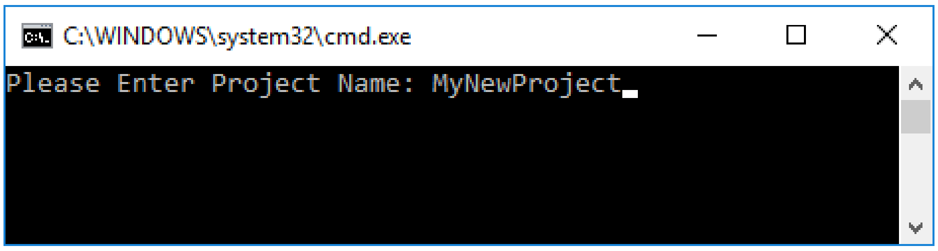
In order to create a new project, go to the following path: **MSF → MercuryProject**. Then execute the "CreateNewProject" batch script (Figure 19).



```
00_Precompiled          12/11/2018 20:59
10_Template             12/11/2018 20:59
20_DemoWifi             12/11/2018 20:59
30_DemoBt               12/11/2018 20:59
CreateNewProject.bat    03/11/2018 21:04
```

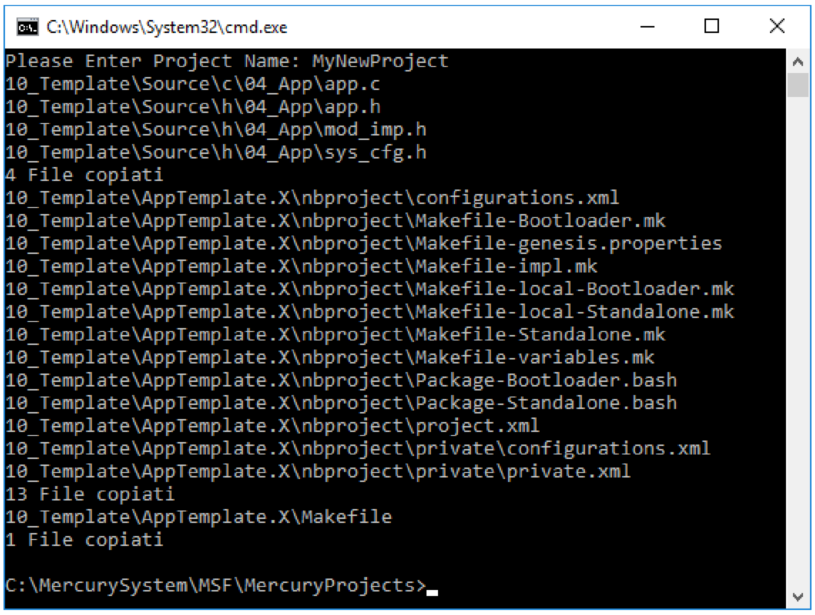
Figure 19 - Create New Project script

Once launched the script asks for a project name (Figure 20), and then create the correct folder structure and the MPLab X project with the correct links to the framework (Figure 21).



```
Please Enter Project Name: MyNewProject_
```

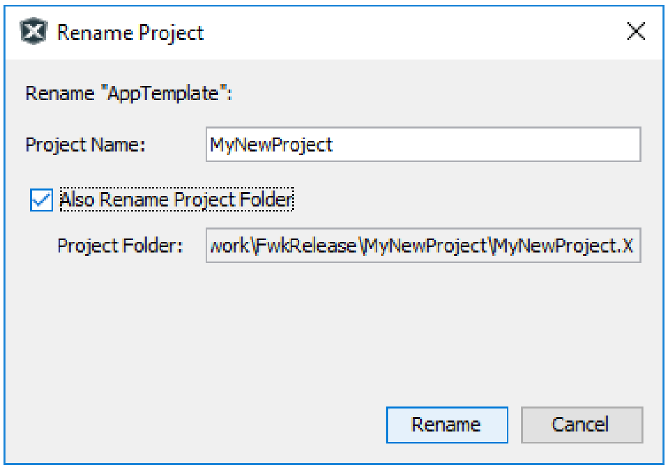
Figure 20 - New Project Creation



```
Please Enter Project Name: MyNewProject
10_Template\Source\c\04_App\app.c
10_Template\Source\h\04_App\app.h
10_Template\Source\h\04_App\mod_imp.h
10_Template\Source\h\04_App\sys_cfg.h
4 File copiati
10_Template\AppTemplate.X\nbproject\configurations.xml
10_Template\AppTemplate.X\nbproject\Makefile-Bootloader.mk
10_Template\AppTemplate.X\nbproject\Makefile-genesis.properties
10_Template\AppTemplate.X\nbproject\Makefile-impl.mk
10_Template\AppTemplate.X\nbproject\Makefile-local-Bootloader.mk
10_Template\AppTemplate.X\nbproject\Makefile-local-Standalone.mk
10_Template\AppTemplate.X\nbproject\Makefile-Standalone.mk
10_Template\AppTemplate.X\nbproject\Makefile-variables.mk
10_Template\AppTemplate.X\nbproject\Package-Bootloader.bash
10_Template\AppTemplate.X\nbproject\Package-Standalone.bash
10_Template\AppTemplate.X\nbproject\project.xml
10_Template\AppTemplate.X\nbproject\private\configurations.xml
10_Template\AppTemplate.X\nbproject\private\private.xml
13 File copiati
10_Template\AppTemplate.X\Makefile
1 File copiati

C:\MercurySystem\MSF\MercuryProjects>_
```

Figure 21 - New project created

Once the project is created you can open it with MPLab X. The MPLab X project default name is **AppTemplate**, so if you want to choose a different one you need to remane it under the IDE (just right click on the project, select the "Rename" option and provide the new name on the popup window (remember to change also the Project folder name), as depicted in Figure 22.


Rename Project

Rename "AppTemplate":

Project Name: MyNewProject

[✓] Also Rename Project Folder

Project Folder: work\FwkRelease\MyNewProject\MyNewProject.X

Figure 22 - Project renaming

At this point you will have an MPLab X project that looks like the one on Figure 23 ready for your new project implementation:

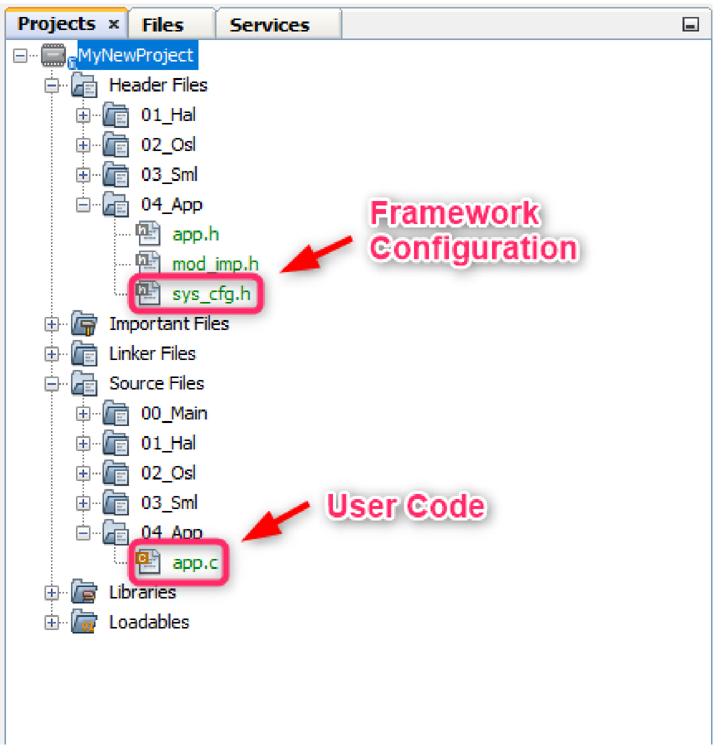


Figure 23 - Project Manager

The user has two pre-defined building configurations:

- **Standalone:** standalone applications that need a Microchip Technology programmer to be flashed on the Base Board,
- **Bootloader:** bootloader compatible applications that can be flashed on the Base Board using the USB bootloader.

The build configuration to use can be selected from the toolbar menu on the top:

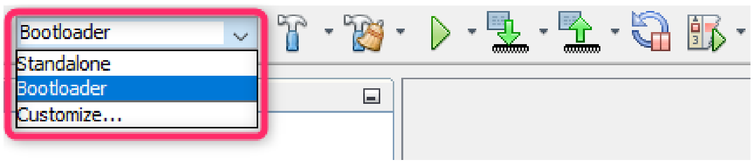


Figure 24 - Building configuration

To build your application with one option or the other use the "Build" or "Clean and Build" option of the IDE (Figure 25).

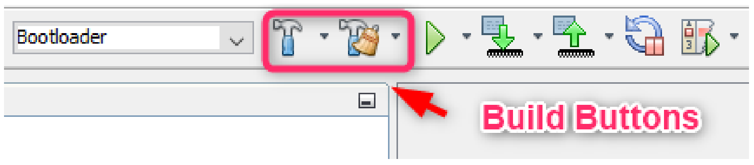
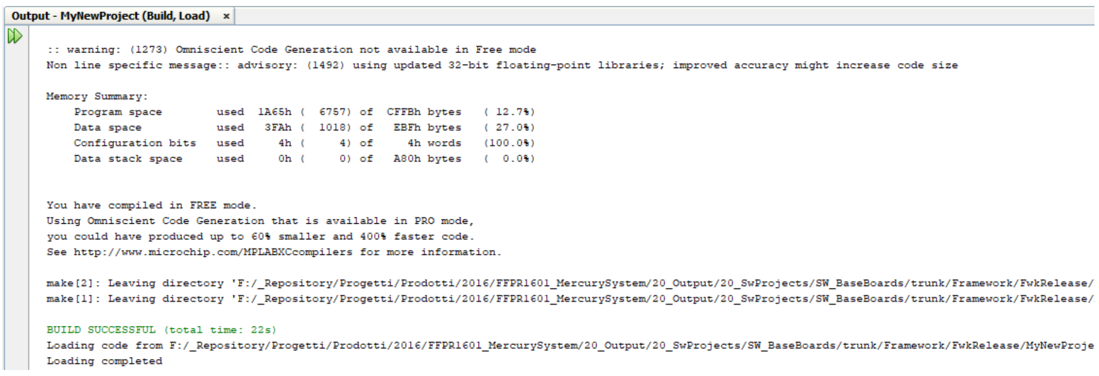


Figure 25 - Build and Clean and Build toolbar buttons

Once the building process is completed the output window will show the final result of the operation () and inside the "dist" folder of your project you will find the .hex file that contains the code to be flashed inside the microcontroller FLASH memory.



```
Output - MyNewProject (Build, Load) ×

:: warning: (1273) Omniscient Code Generation not available in Free mode
Non line specific message:: advisory: (1492) using updated 32-bit floating-point libraries; improved accuracy might increase code size

Memory Summary:
    Program space        used  1A65h (  6757) of  CFFBh bytes   ( 12.7%)
    Data space           used   3FAh (  1018) of   EBFh bytes   ( 27.0%)
    Configuration bits   used     4h (     4) of     4h words   (100.0%)
    Data stack space     used     0h (     0) of   A80h bytes   (  0.0%)


You have compiled in FREE mode.
Using Omniscient Code Generation that is available in PRO mode,
you could have produced up to 60% smaller and 400% faster code.
See http://www.microchip.com/MPLABXCcompilers for more information.

make[2]: Leaving directory 'F:/_Repository/Progetti/Prodotti/2016/FFPR1601_MercurySystem/20_Output/20_SwProjects/SW_BaseBoards/trunk/Framework/FwkRelease/
make[1]: Leaving directory 'F:/_Repository/Progetti/Prodotti/2016/FFPR1601_MercurySystem/20_Output/20_SwProjects/SW_BaseBoards/trunk/Framework/FwkRelease/

BUILD SUCCESSFUL (total time: 22s)
Loading code from F:/_Repository/Progetti/Prodotti/2016/FFPR1601_MercurySystem/20_Output/20_SwProjects/SW_BaseBoards/trunk/Framework/FwkRelease/MyNewProje
Loading completed
```

Figure 26 - Output window after the building process is completed

8. Program the Base Board

As already stated before, there are two options to flash an application developed using the MSF on a Base Board:

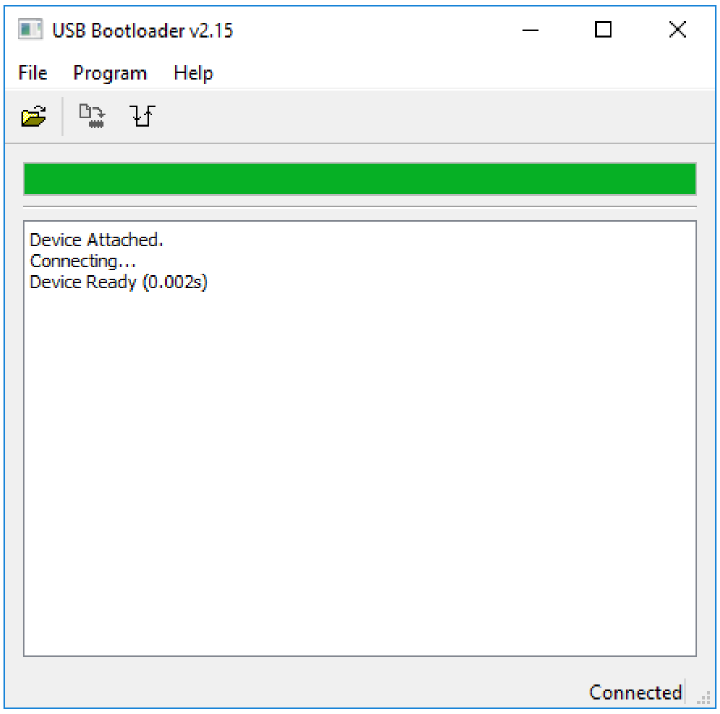
- Use the USB Bootloader,
- Use an external programmer.

The following two sections explain both these options.

Program the Base Board with the USB bootloader

In order to use the Mercury USB bootloader, you need to build your application with the "Bootloader" build configuration (don't use "Standalone"). Before to flash the application using the bootloader, the Base Board need to be put in "Bootloader" mode. To do that connect the Base Board using an USB cable to your PC, press and hold pressed the Base Board user button and turn the board on by moving the main switch to "ON" position (the user LED 1 will start flashing quickly).

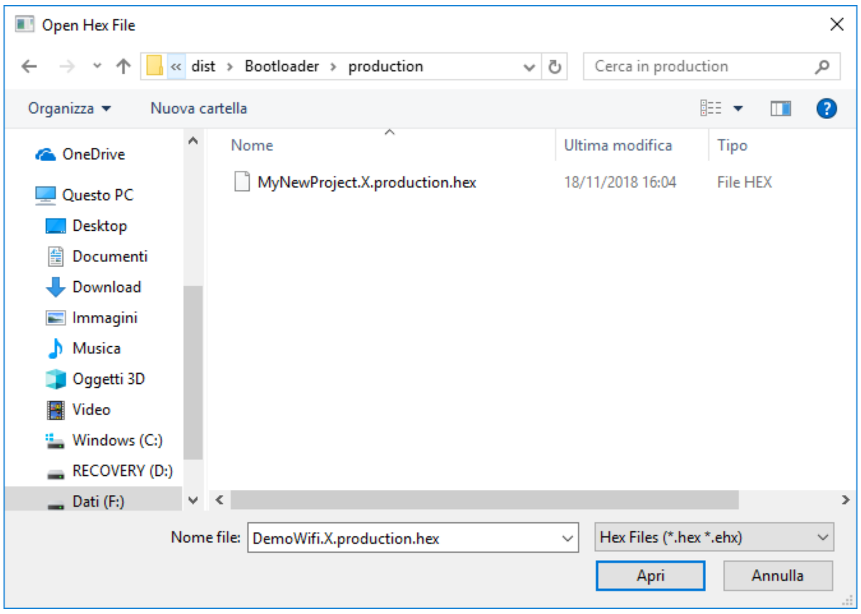
At this point you can launch the bootloader application (you will have a shortcut on you desktop called "Mercury Boot"). You should see something similar to Figure 27.



```
Device Attached.
Connecting...
Device Ready (0.002s)
```

Figure 27 - Host application connected to Bootloader

Now you need to select the .hex file to load using the "import firmware image" option (first from the left on the top).

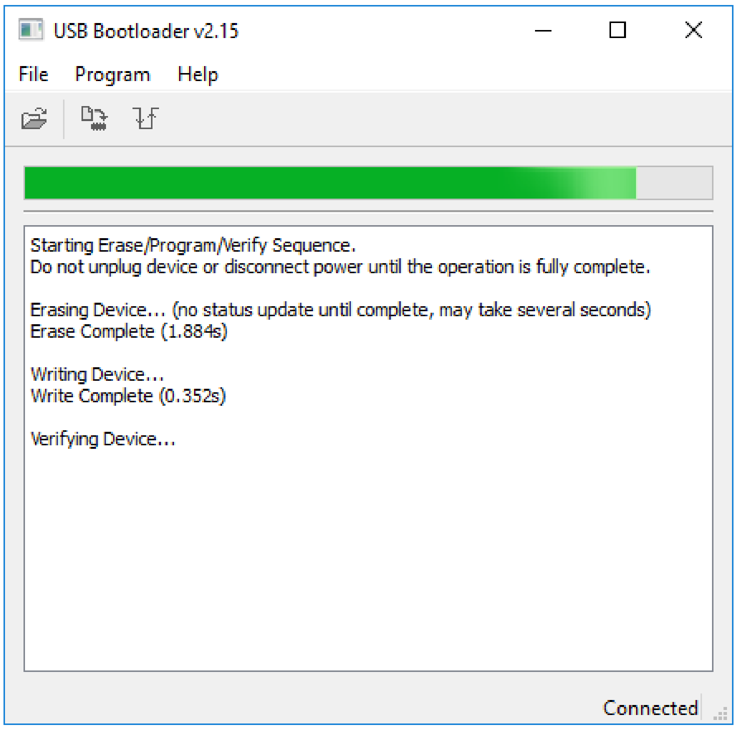


Nome	Ultima modifica	Tipo
MyNewProject.X.production.hex	18/11/2018 16:04	File HEX

Nome file: DemoWifi.X.production.hex
Hex Files (*.hex *.ehx)

Figure 28 - Load hex file

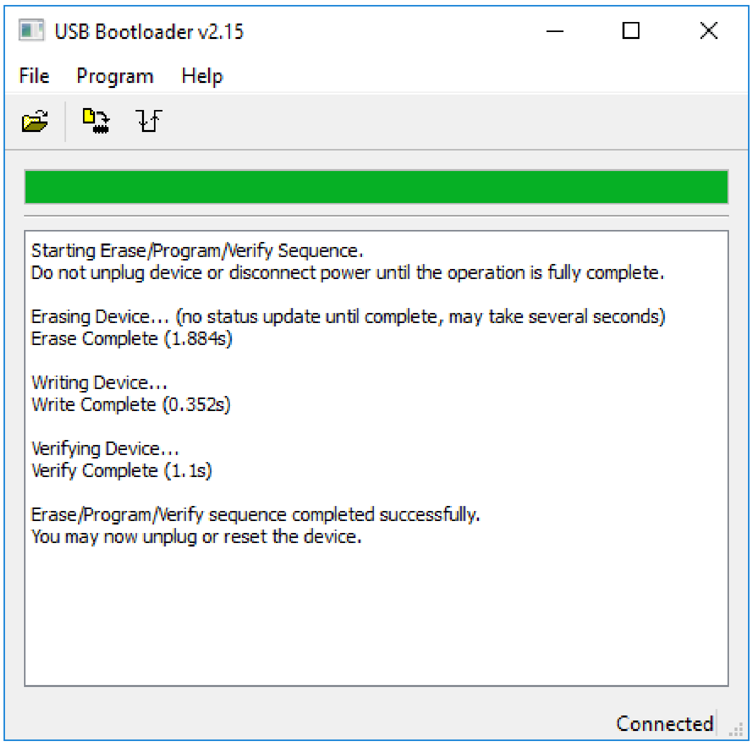
Using the dialog window that appear (Figure 28), select the correct image to load and press "Open". Now you can flash the device using the option "Erase/Program/verify device" (Figure 29).



```
Starting Erase/Program/Verify Sequence.
Do not unplug device or disconnect power until the operation is fully complete.

Erasing Device... (no status update until complete, may take several seconds)
Erase Complete (1.884s)

Writing Device...
Write Complete (0.352s)

Verifying Device...
```

Figure 29 - Programming the Base Board

At the end of bootloading process the host application will show the result of the operation (Figure 30). If everything is ok you can reset the Base Board, the new application will start running.



```
Starting Erase/Program/Verify Sequence.
Do not unplug device or disconnect power until the operation is fully complete.

Erasing Device... (no status update until complete, may take several seconds)
Erase Complete (1.884s)

Writing Device...
Write Complete (0.352s)

Verifying Device...
Verify Complete (1.1s)

Erase/Program/Verify sequence completed successfully.
You may now unplug or reset the device.
```

Figure 30 - Bootloading process completed

Program the Base Board with an external programmer

The other option to program the Base Board is to use an external programmer. This option requires an external device (like an MPLab PicKit or and MPLab ICD programmer), but offers more advanced features as well, like the possibility to using the debugging feature of the programmer/debugger.

To use this option you need to build the application with the “Standalone” configuration. Please note that flashing the device in this way will erase the bootloader, but you can re-flash it anyway, the hex image of the Bootloader is provided in the MSF installation folder. The Mercury Connector contains the Microchip ICSP interface to connect an external Microchip programmer and the Expansion Boards got a dedicated connector for this purpose, check the BB and EBs datasheets for more details.

MPLAB® PICKit 4 In-Circuit Debugger
(Part # PG164140)
MICROCHIP
MPLAB PICKIT 4 IN-CIRCUIT DEBUGGER
DEVELOPMENT TOOLS

Figure 31 - MPLab PicKit 4

9. Examples

Two examples are provided with the framework installation, in order to check the Mercury System features out of the box. The first one is a WiFi examples and the second one is a BT example. The following two sections describe what you need in term of hardware to run the examples and how they work.

BT Example

Description: this example allows to control a Relay via Bluetooth channel. You can run it with an Android smartphone using the provided Android app.

Hardware: the HW you need to run the example is:

- Base Board: BB110
- Modem Board: MB310
- Slave Board: SB110
- Expansion Board (optional): EB110

Working instructions:

1. Set the SB110 address to 0x01
2. Program the BB110 with the DemoBt hex (you can find it at the following path: MSF\MercuryProjects\00_Precompiled\DemoApps\DemoBt.X.production.hex
3. Assemble the system, either in stackable or planar configuration (in this second case using the expansion board)
4. Connect an USB cable and power the system
5. Associate the BT module (name HC-05, pass: 1234)
6. Now you can control the Relay using the provided App

Wifi Example

Description: this example allows to control a Relay via WiFi channel. You can run it with an Android smartphone using the provided Android app.

Hardware: the HW you need to run the example is:

- Base Board: BB110
- Modem Board: MB210
- Slave Board: SB110
- Expansion Board (optional): EB110

Working instructions:

1. Set the SB110 address to 0x01
2. Program the BB110 with the DemoWifi hex (you can find it at the following path: MSF\MercuryProjects\00_Precompiled\DemoApps\DemoWifi.X.production.hex
3. Assemble the system, either in stackable or planar configuration (in this second case using the expansion board)
4. Connect an USB cable and power the system
5. Connect to the “Mercury” WiFi network (WPA2 key: 1234567890)
6. Using the provided App, connect to the IP address “192.168.1.5”, port: 80)
7. Now you can control the Relay using the provided App