Flexible and Rapid Development with MCUXpresso

FreeMASTER Heart Rate Lab

Objectives

In this lab, you will learn:

- How to use the MCUXpresso Installer to obtain NXP Software (FreeMASTER)
- How to use Application Code Hub to import an example into the VS Code workspace
- How to build, clean, debug, and run the example.
- How to connect the Serial Monitor for UART console
- How FreeMASTER can be used as a real-time debug monitor and data visualization tool

Heart Rate Monitor Lab

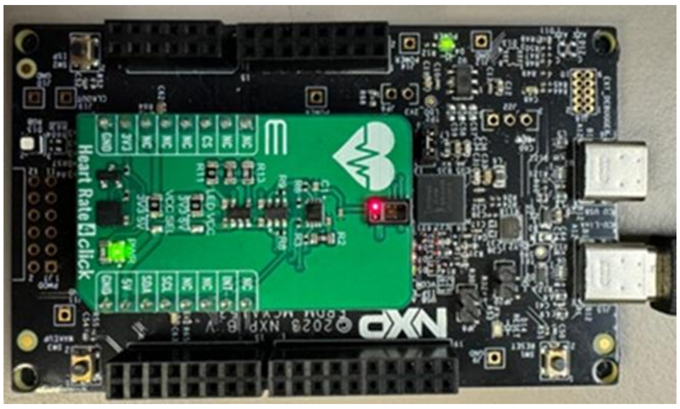
The NXP Application Code Hub provides a complete example of how to use the MCXA-153 microcontroller in a Heart Rate and SPO2 monitor application. This lab will walk through the steps to import, build, program and debug the example. The final section of the lab shows how to use FreeMASTER as a data-visualization tool for the acquired sensor data from the FRDM-MCXA153 evaluation board.

1. Verify Installation / Hardware Setup

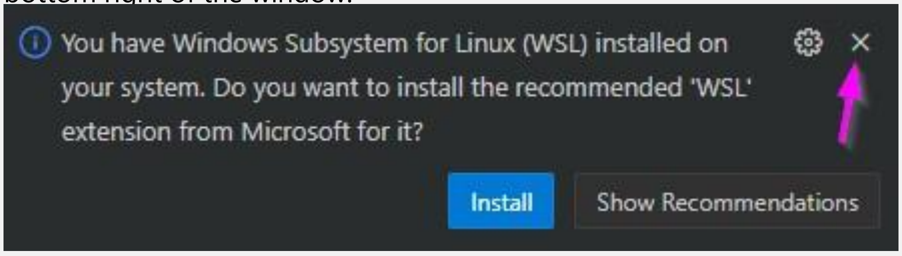
The prework instructions should be completed before starting this lab. The instructions walk through the installation for the required software.
After finishing, the system should have the following:

- Visual Studio Code - Base software tool distributed by Microsoft
- MCUXpresso for Visual Studio Code Extension - NXP developed extension to configure VS Code for embedded MCU development.
- MCXUXpresso Installer - Required to install 3rd party tools for MCUXpresso projects to correctly import/build/debug.
- FRDM-MCXA153 SDK - Required to provide software libraries for middleware and drivers referenced in the example.
- FreeMASTER - Data visualization tool.
- Link to Installation and Preparation

Orientation reference for the Heart Rate 4 Click module on the FRDM-MCXA153 board:



NOTE: When launching MCUXpresso for VS Code you might see the following notification on the bottom right of the window:

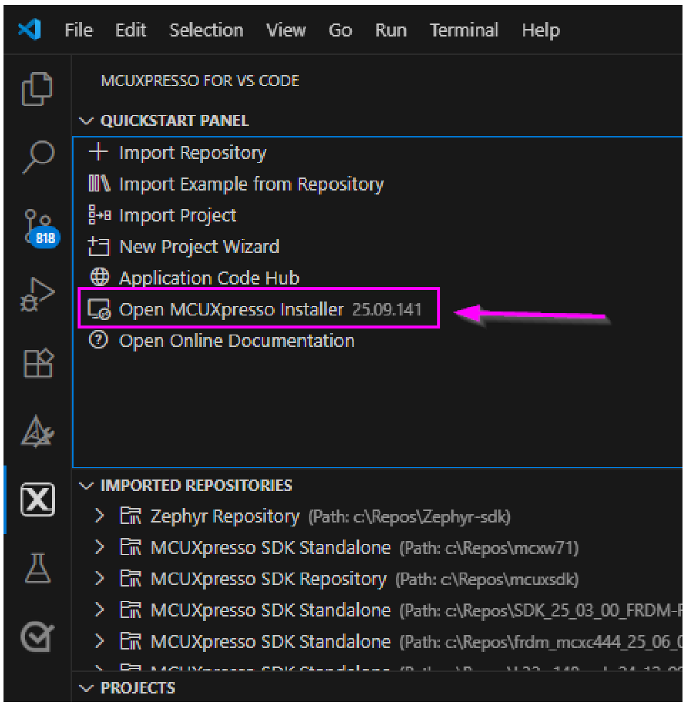


You have Windows Subsystem for Linux (WSL) installed on your system. Do you want to install the recommended 'WSL' extension from Microsoft for it?

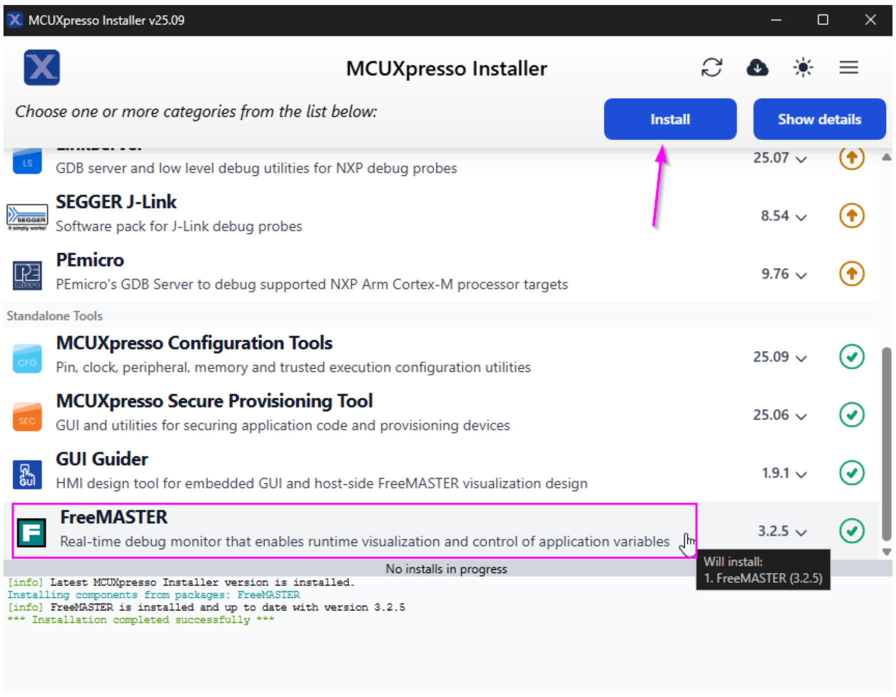
We will not need to install this for this lab. **Close the notification** by clicking the **x** .

To install FreeMASTER:

- Launch the MCUXpresso installer from the MCUXpresso Extension.


File
Edit
Selection
View
Go
Run
Terminal
Help
818
MCUXPRESSO FOR VS CODE
QUICKSTART PANEL
Import Repository
Import Example from Repository
Import Project
New Project Wizard
Application Code Hub
Open MCUXpresso Installer 25.09.141
Open Online Documentation
IMPORTED REPOSITORIES
Zephyr Repository (Path: c:\Repos\Zephyr-sdk)
MCUXpresso SDK Standalone (Path: c:\Repos\mcxw71)
MCUXpresso SDK Repository (Path: c:\Repos\mcuxsdk)
MCUXpresso SDK Standalone (Path: c:\Repos\SDK_25_03_00_FRDM-
MCUXpresso SDK Standalone (Path: c:\Repos\frdm_mcxc444_25_06_
PROJECTS

- Select FreeMASTER and click install. You will be prompted for your NXP account credentials when the installation begins.



Choose one or more categories from the list below:

GDB server and low level debug utilities for NXP debug probes 25.07

SEGGER J-Link 8.54
Software pack for J-Link debug probes

PEmicro 9.76
PEmicro's GDB Server to debug supported NXP Arm Cortex-M processor targets

Standalone Tools

MCUXpresso Configuration Tools 25.09
Pin, clock, peripheral, memory and trusted execution configuration utilities

MCUXpresso Secure Provisioning Tool 25.06
GUI and utilities for securing application code and provisioning devices

GUI Guider 1.9.1
HMI design tool for embedded GUI and host-side FreeMASTER visualization design

FreeMASTER 3.2.5
Real-time debug monitor that enables runtime visualization and control of application variables

No installs in progress

```
[info] Latest MCUXpresso Installer version is installed.
Installing components from packages: FreeMASTER
[info] FreeMASTER is installed and up to date with version 3.2.5
*** Installation completed successfully ***
```

2. Import Project with Application Code Hub

The Application Code Hub (ACH) repository enables engineers to easily find microcontroller software examples, code snippets, application software packs and demos developed by NXP in-house experts. This space provides a quick, easy and consistent way to find microcontroller applications. Find more information at www.nxp.com/ach

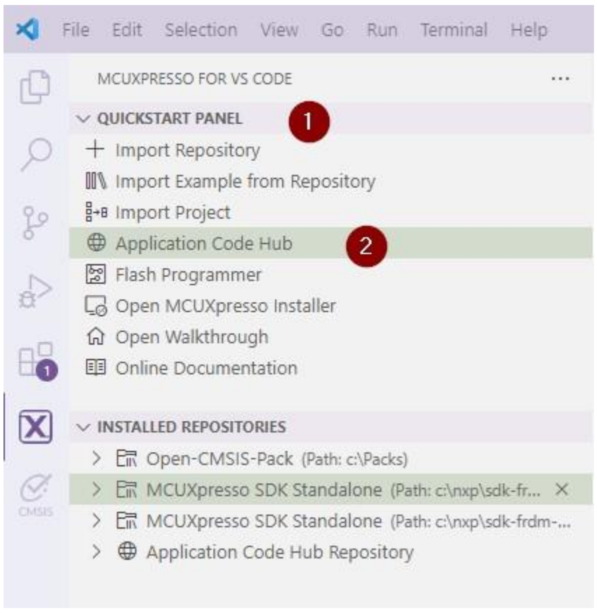
The MCUXpresso for VS Code extension has integrated the Application Code Hub into the development environment. The extension provides an Import Wizard to simplify adding examples from the Application Code Hub. The wizard handles the steps required to clone the selected project repository.

> ***NOTE:*** The Application Code Hub examples are delivered from GitHub at www.github.com/nxpappcodehub The integrated viewer makes it easier for customers to explore. Alternatively, customers can manually add the projects using traditional methods to clone GitHub repositories and importing the local project folder.

Here are the steps to import a new Example from the Application Code Hub:

1. Go to the Quick Start Panel

2. Select Application Code Hub


File Edit Selection View Go Run Terminal Help
MCUXPRESSO FOR VS CODE
QUICKSTART PANEL
1
Import Repository
Import Example from Repository
Import Project
Application Code Hub
2
Flash Programmer
Open MCUXpresso Installer
Open Walkthrough
Online Documentation
INSTALLED REPOSITORIES
Open-CMSIS-Pack (Path: c:\Packs)
MCUXpresso SDK Standalone (Path: c:\nxp\sdk-fr...
MCUXpresso SDK Standalone (Path: c:\nxp\sdk-frdm-...
Application Code Hub Repository
CMSIS

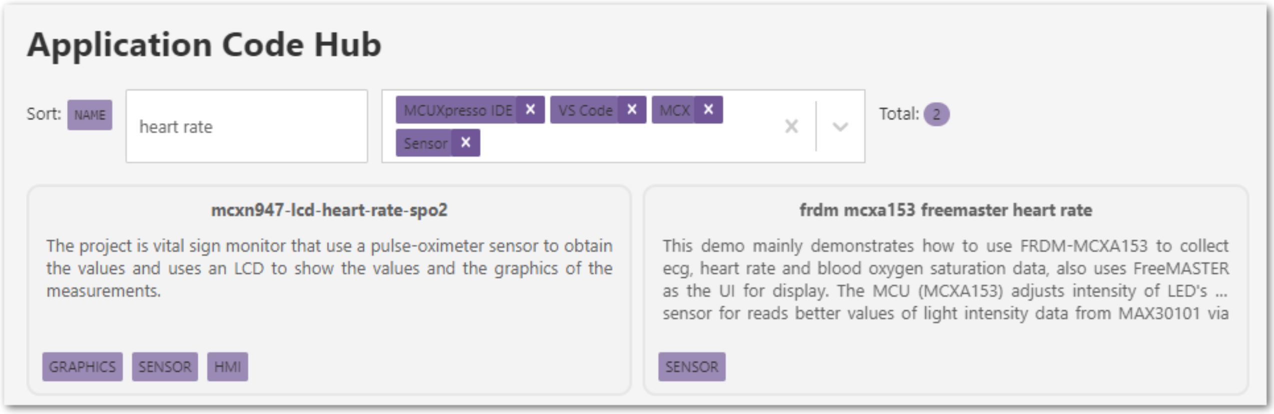
3.Filter Visible Examples (MCX + Sensors)

Go to the filter section next to the Search bar and select two filters. **MCX** in the Device Families Section and **Sensor** in the Categories section of the filters.

4.Search for Keywords in Examples

Search for the keyword '**heart rate**'.

Select the demo ***"frdm mcxa153 freemaster heart rate"***.



Application Code Hub

mcxn947-lcd-heart-rate-spo2

The project is vital sign monitor that use a pulse-oximeter sensor to obtain the values and uses an LCD to show the values and the graphics of the measurements.

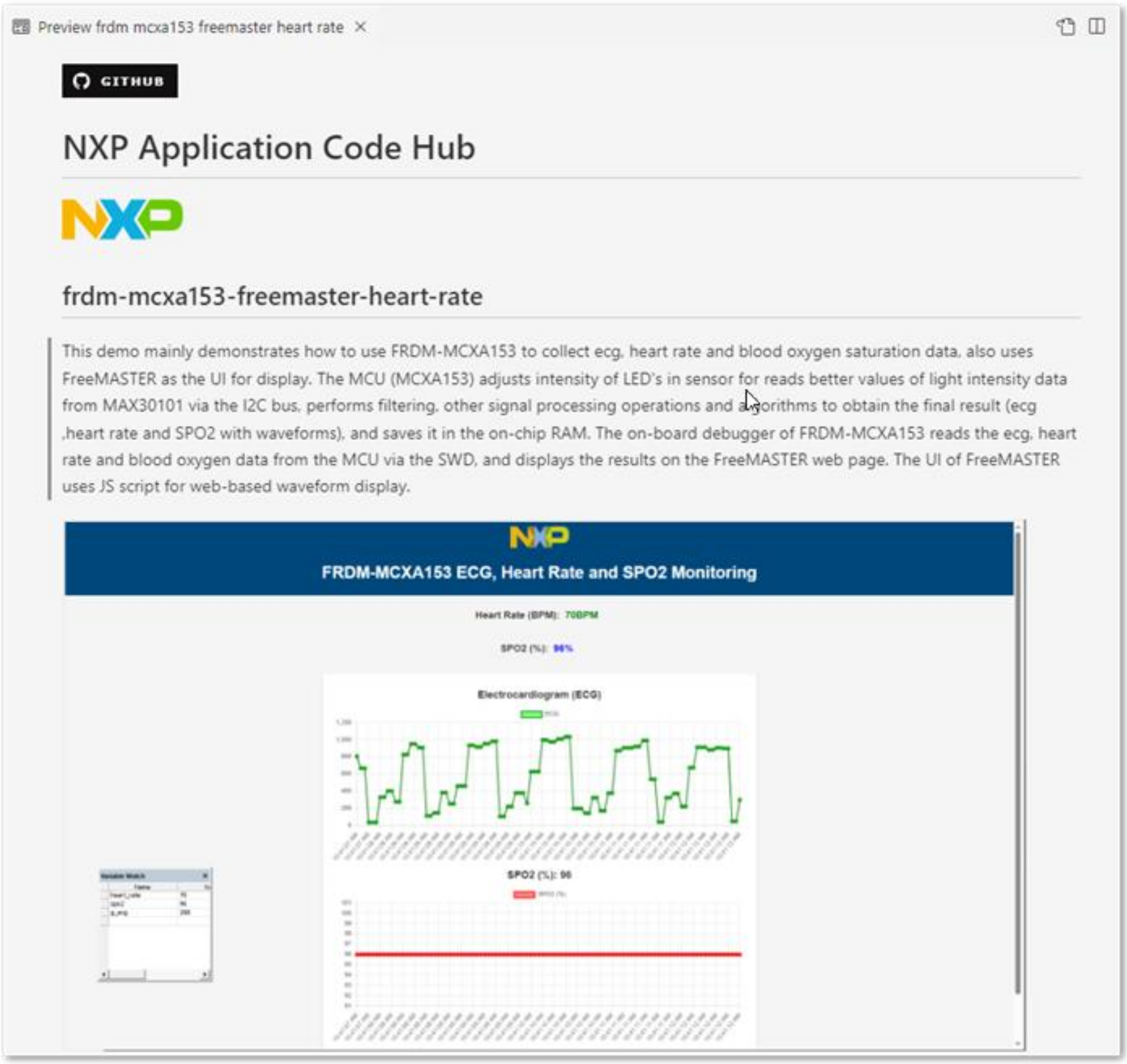
frdm mcxa153 freemaster heart rate

This demo mainly demonstrates how to use FRDM-MCXA153 to collect ecg, heart rate and blood oxygen saturation data, also uses FreeMASTER as the UI for display. The MCU (MCXA153) adjusts intensity of LED's ... sensor for reads better values of light intensity data from MAX30101 via

5.Read Overview of Heart Rate Demo

The Application Code Hub provides a consistent Readme Overview for every project. The FreeMASTER Heart Rate demo overview is previewed after clicking on the application card.

Scroll through the readme to become familiar with the available contents like required **hardware**, **software** and **setup** instructions.



NXP Application Code Hub

frdm-mcxa153-freemaster-heart-rate

This demo mainly demonstrates how to use FRDM-MCXA153 to collect ecg, heart rate and blood oxygen saturation data, also uses FreeMASTER as the UI for display. The MCU (MCXA153) adjusts intensity of LED's in sensor for reads better values of light intensity data from MAX30101 via the I2C bus, performs filtering, other signal processing operations and algorithms to obtain the final result (ecg ,heart rate and SPO2 with waveforms), and saves it in the on-chip RAM. The on-board debugger of FRDM-MCXA153 reads the ecg, heart rate and blood oxygen data from the MCU via the SWD, and displays the results on the FreeMASTER web page. The UI of FreeMASTER uses JS script for web-based waveform display.

6.Select Destination for Project

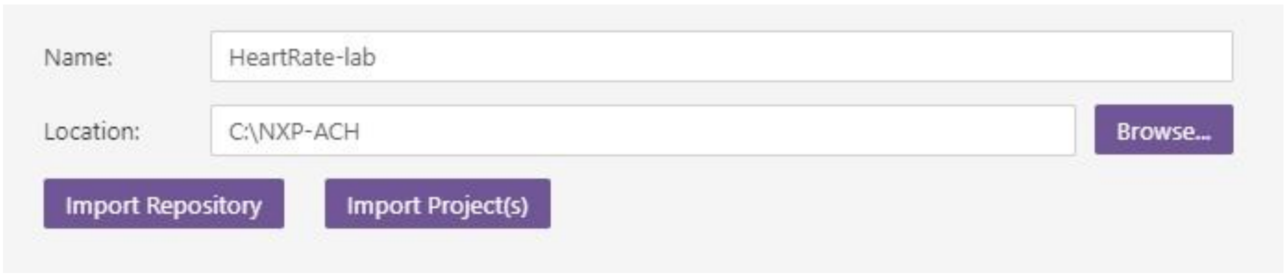
The wizard automatically provides a prompt to browse to a desired destination folder. Create the destination C:\NXP-ACH to store the project here. Or you can specify a custom location.

> ***NOTE:*** Too many characters may cause an issue during the build process. This issue is currently being addressed. Please rename the project to **HeartRate-lab** and use a project path close to your root folder as shown below in step 7.

7. Import Project into Workspace

   Select **Import Project(s)** after entering the desired location.

   If a valid project is not available, the wizard only displays **Import Repository**, to allow a code repo, without a project, to be added to workspace.

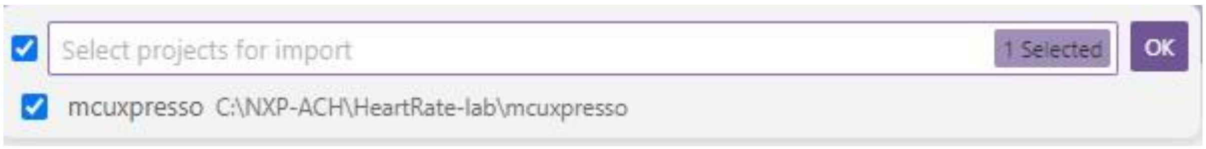


Name: HeartRate-lab

Location: C:\NXP-ACH

8. Select Detected Project(s)

   The import wizard will scan the example repo and list valid projects that were discovered. This allows the user to select only the projects they want created. Select the **mcuxpresso** project listed at the top of the VS Code window below the Workspace Search bar.

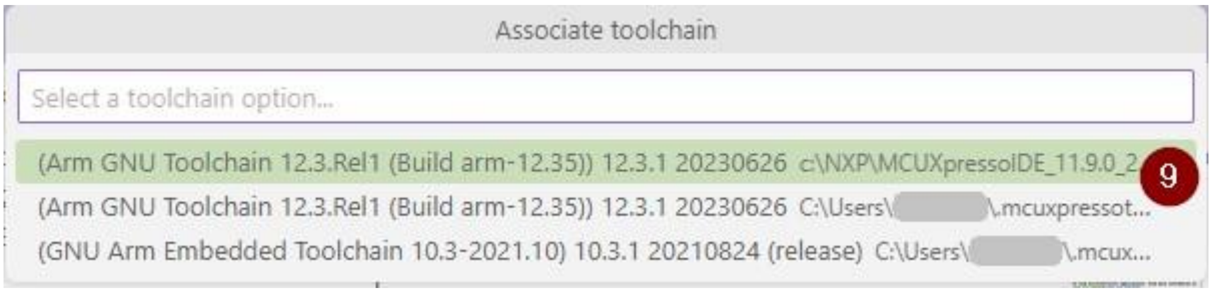


mcuxpresso C:\NXP-ACH\HeartRate-lab\mcuxpresso

9.Choose a Toolchain

The last selection is to identify the Compiler toolchain to be used for the project. GCC will be used for this project.

Select **Arm GNU Toolchain 12.3.Rel1** (Or latest version available from MCUXpresso Installer prework) The scan may locate Compilers associated with MCUXPresso IDE. Verify the path and version between listed compilers.


Associate toolchain

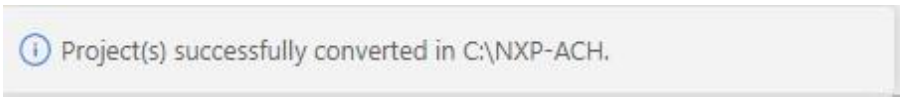
Select a toolchain option...

(Arm GNU Toolchain 12.3.Rel1 (Build arm-12.35)) 12.3.1 20230626 c:\NXP\MCUXpressoIDE_11.9.0_2

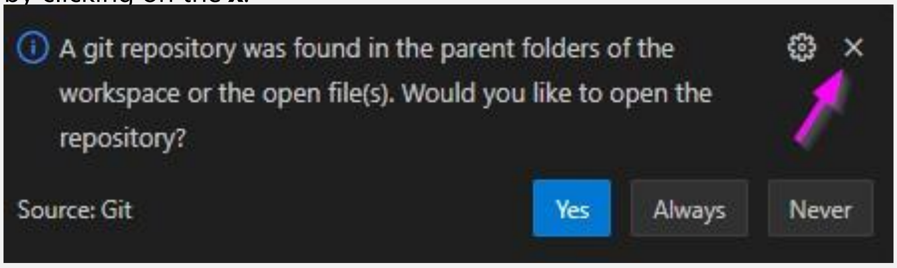
(Arm GNU Toolchain 12.3.Rel1 (Build arm-12.35)) 12.3.1 20230626 C:\Users\[redacted]\.mcuxpressot...

(GNU Arm Embedded Toolchain 10.3-2021.10) 10.3.1 20210824 (release) C:\Users\[redacted]\.mcux...

At this time the wizard completes importing the project into the workspace. A ***Successful Conversion*** notification is displayed at the bottom of the screen. It is important to recognize that the selected Heart Rate example is a working project within the MCUXpresso IDE (Eclipse based). The VS Code extension has the ability to convert an existing project with a few requirements.


Project(s) successfully converted in C:\NXP-ACH.

NOTE: The git notification on the bottom right can be ignored for this demo. **Close the notification** by clicking on the **x**.


A git repository was found in the parent folders of the workspace or the open file(s). Would you like to open the repository?

Source: Git

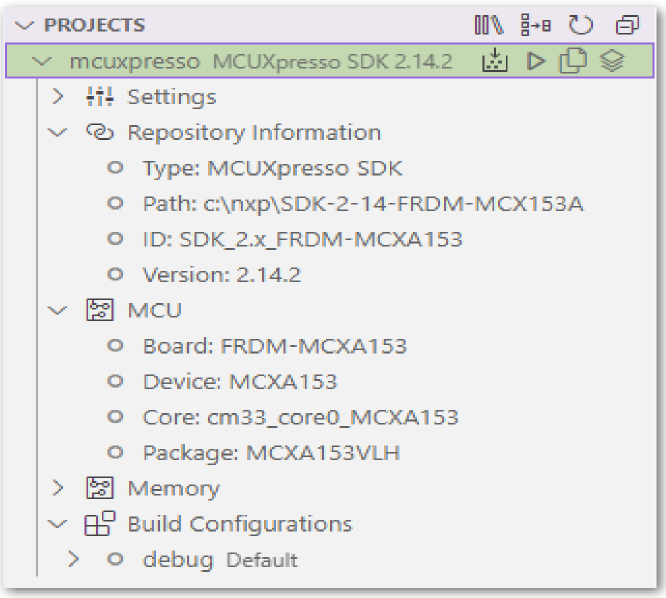
3. Navigating a Project in VS Code

The MCUXpresso for VS Code extension includes a PROJECTS section to help users access useful project information.
A user can review and modify project information with the following steps.

1. Review Project Details
   Project details are shown in the Dropdown menu of the Projects Section in the MCUXpresso Extension Navigation Pane

   - ***Repository Information***: Verify the necessary SDK is associated with converted project.
     NOTE: Step 2 below details how this can be corrected.

   - ***MCU***: Understand the targeted device and board.
   - **Memory :** View quick memory map of project.
   - **Build Configuration :** Select build configuration from available list (i.e. Debug or Release).



```
PROJECTS
  mcuxpresso  MCUXpresso SDK 2.14.2
    > Settings
    v Repository Information
        o Type: MCUXpresso SDK
        o Path: c:\nxp\SDK-2-14-FRDM-MCX153A
        o ID: SDK_2.x_FRDM-MCXA153
        o Version: 2.14.2
    v MCU
        o Board: FRDM-MCXA153
        o Device: MCXA153
        o Core: cm33_core0_MCXA153
        o Package: MCXA153VLH
    > Memory
    v Build Configurations
      > o debug  Default
```

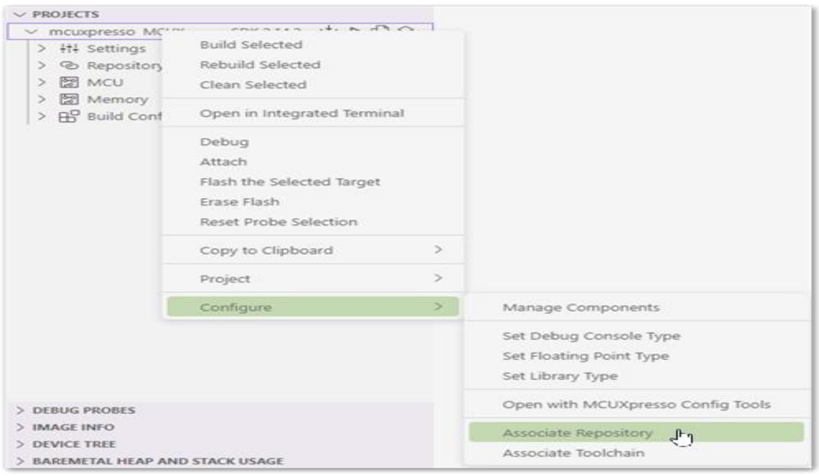
2. Associate the Required MCUXpresso SDK

Sometimes, Projects imported/converted from the Application Code Hub do not automatically have a valid SDK associated to them. This can be caused by the required SDK not being installed and imported into the workspace.

The prework for this lab installed the expected FRDM-MCXA153 SDK. The import wizard should associate the information from the project with the available SDK.
If Project information is blank (i.e. Repository Information), you must manually associate an SDK with a project.

The Following steps will assign the FRDM-MCXA153 SDK to the project:

- **Right Click** on the Project to fix.
- In the Drop-down menu, **Select Configure -> Associate Repository**
- **Select SDK-FRDM-MCXA153** from a list of available SDK repos listed at the top of VS Code window below search bar.



Now the correct information should be displayed for the project.

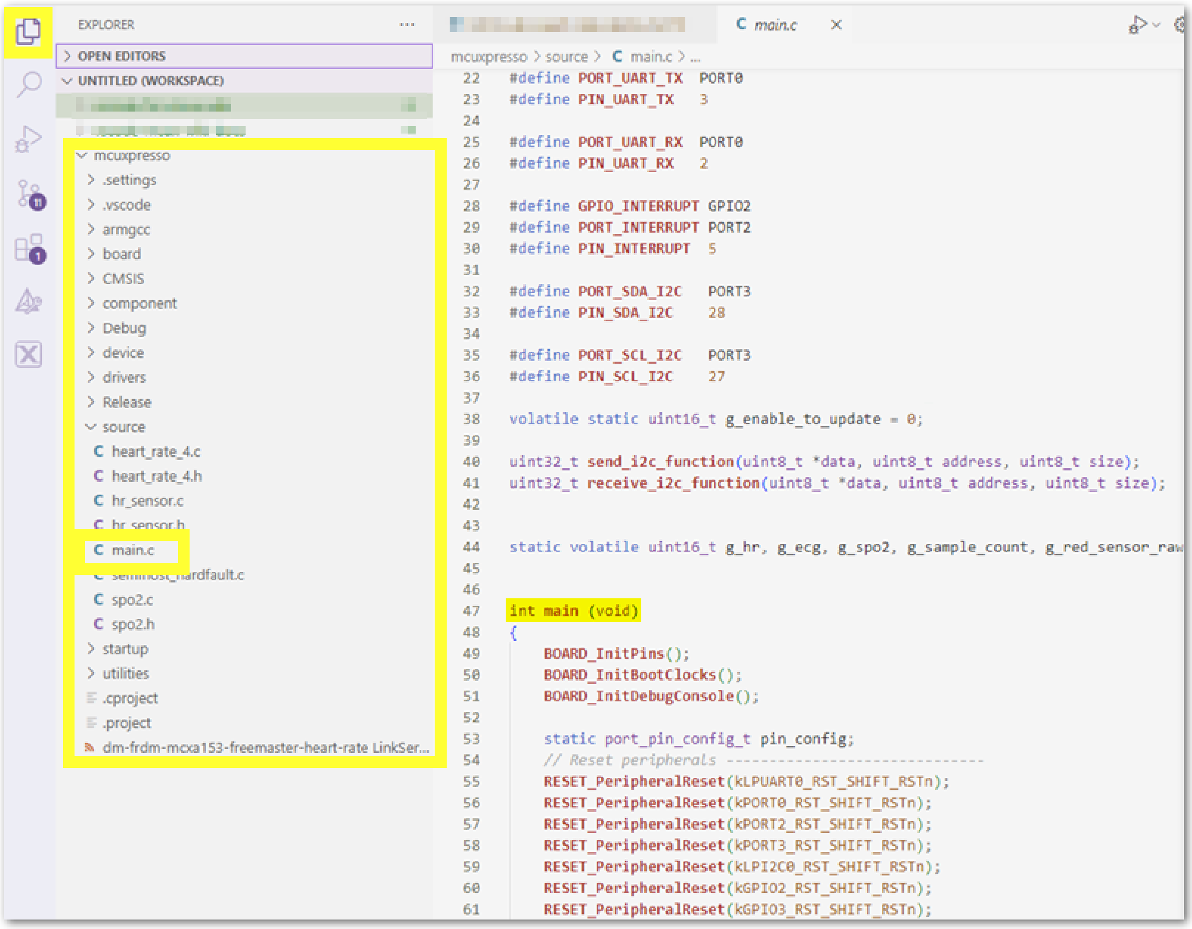
> ***NOTE:*** If a valid SDK is not assigned, a project will fail to build with an Error message noting that a CMAKE build configuration is not located.

3. Working with Source Files

The MCUXpresso extension **Projects** view does not include the source files. Visual Studio Code leverages a dedicated Explorer view to work with files associated with projects found in the workspace.

To view the project files:

- Click on the Explorer Icon at the top of the VS Code left navigation pane. (Highlighted in image below)
- Click on **mcuxpresso** and you will find the source files like main.cpp etc in the Dropdown.



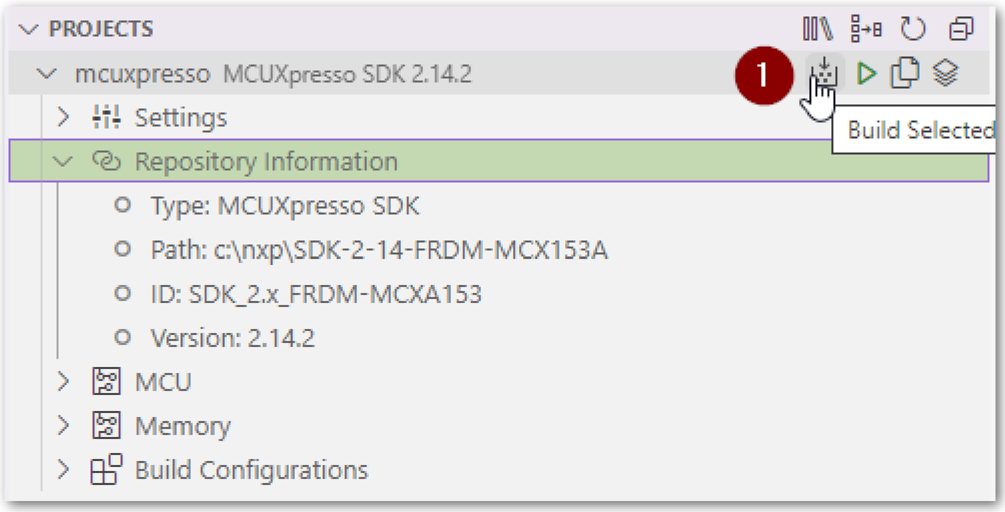
```c
#define PORT_UART_TX  PORT0
#define PIN_UART_TX   3

#define PORT_UART_RX  PORT0
#define PIN_UART_RX   2

#define GPIO_INTERRUPT GPIO2
#define PORT_INTERRUPT PORT2
#define PIN_INTERRUPT  5

#define PORT_SDA_I2C   PORT3
#define PIN_SDA_I2C    28

#define PORT_SCL_I2C   PORT3
#define PIN_SCL_I2C    27

volatile static uint16_t g_enable_to_update = 0;

uint32_t send_i2c_function(uint8_t *data, uint8_t address, uint8_t size);
uint32_t receive_i2c_function(uint8_t *data, uint8_t address, uint8_t size);


static volatile uint16_t g_hr, g_ecg, g_spo2, g_sample_count, g_red_sensor_raw


int main (void)
{
    BOARD_InitPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();

    static port_pin_config_t pin_config;
    // Reset peripherals ------------------------------
    RESET_PeripheralReset(kLPUART0_RST_SHIFT_RSTn);
    RESET_PeripheralReset(kPORT0_RST_SHIFT_RSTn);
    RESET_PeripheralReset(kPORT2_RST_SHIFT_RSTn);
    RESET_PeripheralReset(kPORT3_RST_SHIFT_RSTn);
    RESET_PeripheralReset(kLPI2C0_RST_SHIFT_RSTn);
    RESET_PeripheralReset(kGPIO2_RST_SHIFT_RSTn);
    RESET_PeripheralReset(kGPIO3_RST_SHIFT_RSTn);
```

4. Build the application

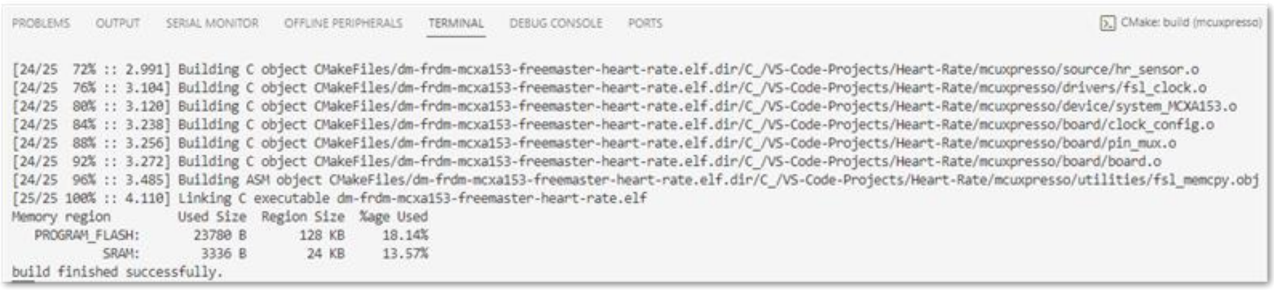
The *MCXA153 FreeMASTER Heart Rate* project needs to build the application image. After the code builds without any errors, the application can be run on the FRDM board.

The following steps require that you return to the MCUXpresso perspective by clicking the **MCUXpresso for VS Code** X icon in the left navigation pane.

1. Build the project by clicking the **Build Selected** icon.



After a successful build, the Terminal console displays the memory usage (or compiler errors if any).



```
[24/25  72% :: 2.991] Building C object CMakeFiles/dm-frdm-mcxa153-freemaster-heart-rate.elf.dir/C_/VS-Code-Projects/Heart-Rate/mcuxpresso/source/hr_sensor.o
[24/25  76% :: 3.104] Building C object CMakeFiles/dm-frdm-mcxa153-freemaster-heart-rate.elf.dir/C_/VS-Code-Projects/Heart-Rate/mcuxpresso/drivers/fsl_clock.o
[24/25  80% :: 3.120] Building C object CMakeFiles/dm-frdm-mcxa153-freemaster-heart-rate.elf.dir/C_/VS-Code-Projects/Heart-Rate/mcuxpresso/device/system_MCXA153.o
[24/25  84% :: 3.238] Building C object CMakeFiles/dm-frdm-mcxa153-freemaster-heart-rate.elf.dir/C_/VS-Code-Projects/Heart-Rate/mcuxpresso/board/clock_config.o
[24/25  88% :: 3.256] Building C object CMakeFiles/dm-frdm-mcxa153-freemaster-heart-rate.elf.dir/C_/VS-Code-Projects/Heart-Rate/mcuxpresso/board/pin_mux.o
[24/25  92% :: 3.272] Building C object CMakeFiles/dm-frdm-mcxa153-freemaster-heart-rate.elf.dir/C_/VS-Code-Projects/Heart-Rate/mcuxpresso/board/board.o
[24/25  96% :: 3.485] Building ASM object CMakeFiles/dm-frdm-mcxa153-freemaster-heart-rate.elf.dir/C_/VS-Code-Projects/Heart-Rate/mcuxpresso/utilities/fsl_memcpy.obj
[25/25 100% :: 4.110] Linking C executable dm-frdm-mcxa153-freemaster-heart-rate.elf
Memory region         Used Size  Region Size  %age Used
   PROGRAM_FLASH:        23780 B       128 KB     18.14%
            SRAM:         3336 B        24 KB     13.57%
build finished successfully.
```

5. Connect Serial Monitor to the board

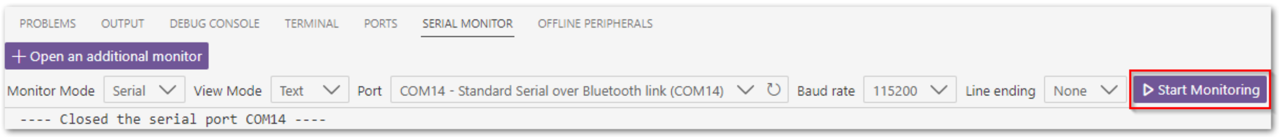
To use the Serial Monitor integrated into VS Code:

1. Connect the **USB-C cable to J15** to power the FRDM board. The onboard debugger provides a USBUART bridge to interface with the Serial monitor.

2. Click on the **SERIAL MONITOR** found as a tab in the Terminal window at the Bottom of VS Code Window.
   NOTE: The default COM settings are valid for NXP eval boards: "115200, None..."

3. Click **Start Monitoring** to connect the monitor to the FRDM board's auto-detected COM port.

   It can be disconnected by clicking **Stop Monitoring** after debug.



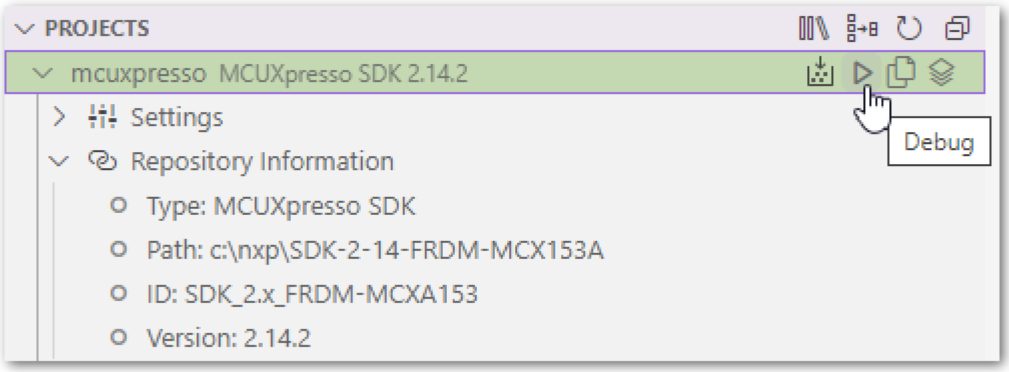
Monitor Mode Serial · View Mode Text · Port COM14 - Standard Serial over Bluetooth link (COM14) · Baud rate 115200 · Line ending None

```
---- Closed the serial port COM14 ----
```

6. Flash/Debug the Application

This section uses the on-board debugger to connect to the MCU, and program the flash. LinkServer from NXP manages the GDB server for communicating with the NXP MCULink on-board debug probe. It includes support for flash programming.

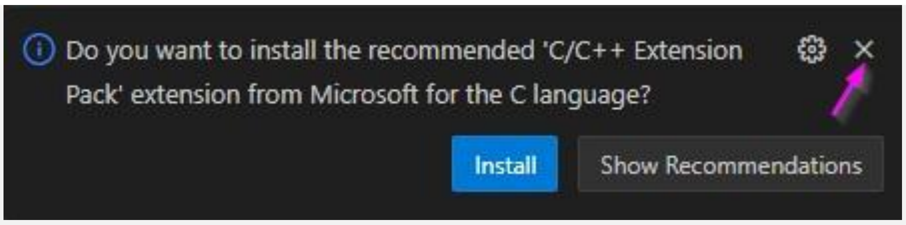
1. Click the play icon to **Debug** the application:



- Type: MCUXpresso SDK
- Path: c:\nxp\SDK-2-14-FRDM-MCX153A
- ID: SDK_2.x_FRDM-MCXA153
- Version: 2.14.2

The application is flashed to the FRDM board and VS Code switches to the Debug perspective. Return to the SERIAL MONITOR tab under the Terminals. It switches to the OUTPUT terminal when a Debug session is started.

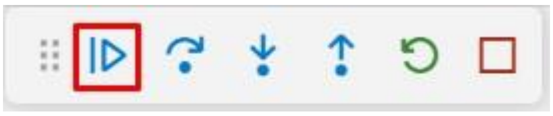
> ***NOTE:*** The C/C++ Extension Pack is not required and is optional. **Close this notification** by clicking on the **x**.



Do you want to install the recommended 'C/C++ Extension Pack' extension from Microsoft for the C language?

2. The execution will be paused.
Click **Continue/Play** icon to continue execution.
The application will advance to the start of main().

Click **Continue/Play** icon a 2nd time for the Heart Rate application to launch inside main().



3. View Heart Rate Values in Serial Terminal The Heart Rate application using the serial port to display information.

   The following should be displayed in the SERIAL MONITOR tab after *main()* starts.

```
sensor_init
init complete
```

   Place a finger on the sensor near the Heart silkscreened on Heart Rate 4 click board. The following should be displayed in the SERIAL MONITOR tab after a finger is placed on the sensor:

```
g_sample_count:0, hr:0 ecg:104, blood:0:
g_sample_count:0, hr:0 ecg:106, blood:0:
g_sample_count:0, hr:0 ecg:127, blood:0:
```

- A heart rate value will be calculated and displayed after *g_sample_count* reaches 250 valid samples:

```
g_sample_count:249, hr:0 ecg:104, blood:81:
g_sample_count:250, hr:72 ecg:86, blood:81:
g_sample_count:251, hr:72 ecg:68, blood:81:
```

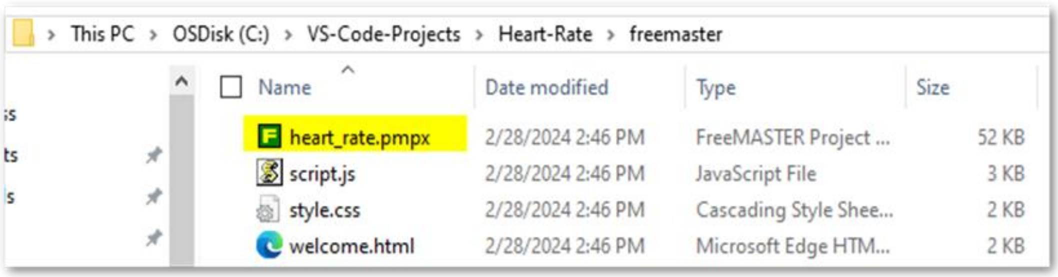
7. FreeMASTER Data Visualization

FreeMASTER is a standalone application provided by NXP to help developers visualize, monitor and manipulate data available from their projects. The Heart Rate example includes a **/freemaster** folder that helps users get started using the tool.

The settings in the Application Code Hub were established for an MCUXpresso IDE based project. There are a few changes that need to be made after the project is converted to a VS Code project.
The following steps will properly configure FreeMASTER to work with the Heart Rate example project:

1. Launch FreeMASTER Application
   There are two options for launching FreeMASTER.

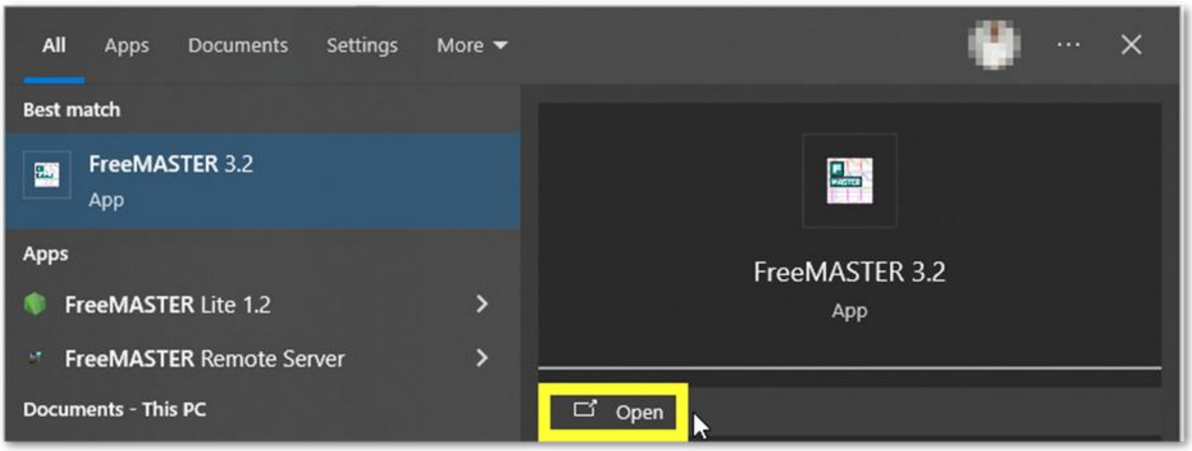
   - Click on the **heart_rate.pmpx** using File Explorer. The FreeMASTER application should be associated with .pmpx file extensions. This will also automatically load the included project settings. (Folder location may vary based on choices made in *Section 2*).



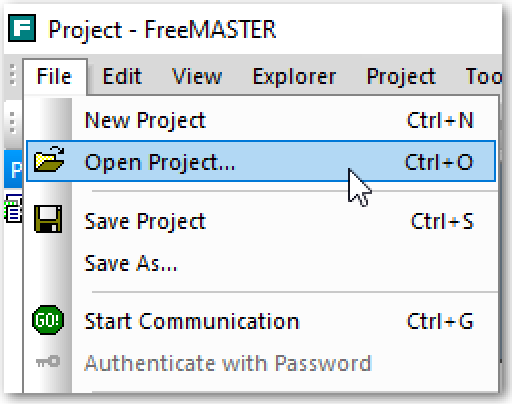
This PC › OSDisk (C:) › VS-Code-Projects › Heart-Rate › freemaster

Name	Date modified	Type	Size
heart_rate.pmpx	2/28/2024 2:46 PM	FreeMASTER Project ...	52 KB
script.js	2/28/2024 2:46 PM	JavaScript File	3 KB
style.css	2/28/2024 2:46 PM	Cascading Style Shee...	2 KB
welcome.html	2/28/2024 2:46 PM	Microsoft Edge HTM...	2 KB

- Launch FreeMASTER by searching Windows Applications.



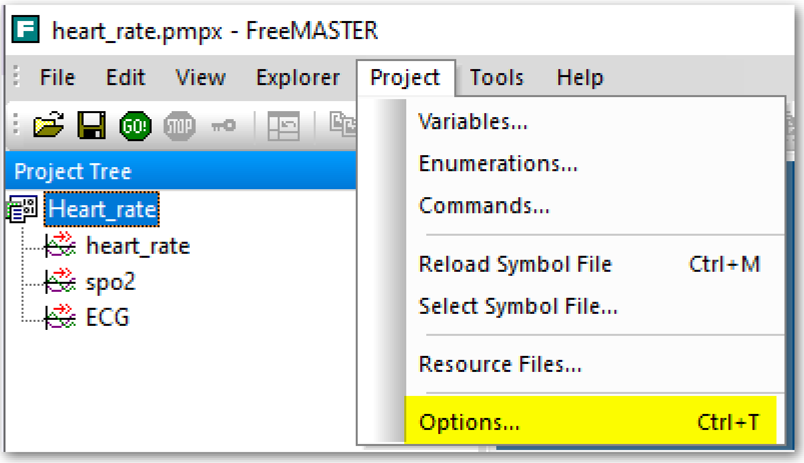
This will not load project settings. You will be required to Open Project using the FreeMaster menu as shown. Open the .pmpx project file located in the location chosen in *Section 2*)



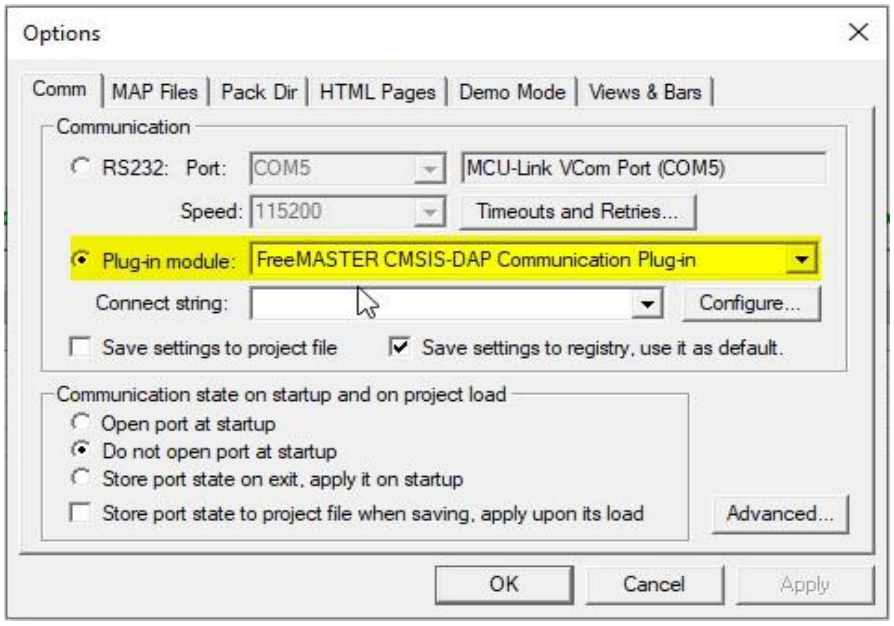
2. Verify Project Options

FreeMASTER has a few key settings to verify once a project is opened. A user should verify they are correctly set for the type of Debug Probe and location of the project output files.

1. **Click Project -> Options** from the menu bar.



2. Verify that the correct method is set for communicating with the board. The on-board Debug Probe for the FRDM-MCXA153 is by default shipped with NXP CMSIS-DAP firmware. Select **FreeMASTER CMSIS-DAP Communication Plug-in** found under the **Comm** tab, for **Plug-in module:**



Options

Comm | MAP Files | Pack Dir | HTML Pages | Demo Mode | Views & Bars

Communication

RS232: Port: COM5 — MCU-Link VCom Port (COM5)

Speed: 115200 — Timeouts and Retries...

[✓ Plug-in module: FreeMASTER CMSIS-DAP Communication Plug-in]

Connect string: — Configure...

☐ Save settings to project file ☑ Save settings to registry, use it as default.

Communication state on startup and on project load

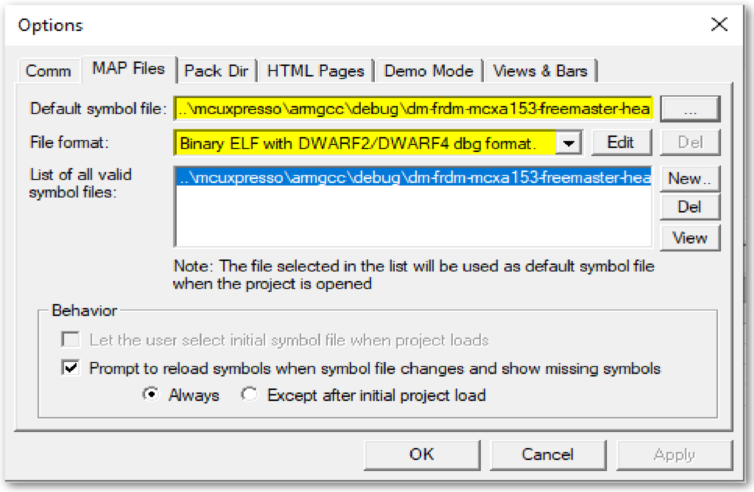
Open port at startup [✓ Do not open port at startup] Store port state on exit, apply it on startup

☐ Store port state to project file when saving, apply upon its load

Advanced...

NOTE: SEGGER and PEMicro options are available for alternate debug probes or on-board firmware

3. Verify that the correct **Default symbol file** is targeted for the VS Code project. The symbol file in VS Code projects is output under an **/armgcc** folder. Select the **/armgcc** folder within VS Code project **MAP Files** tab. The window will autodetect the Binary ELF File, and display this under **File format:**.



Options

Comm | MAP Files | Pack Dir | HTML Pages | Demo Mode | Views & Bars

Default symbol file: ..\mcuxpresso\armgcc\debug\dm-frdm-mcxa153-freemaster-hea

File format: Binary ELF with DWARF2/DWARF4 dbg format.

List of all valid symbol files: ..\mcuxpresso\armgcc\debug\dm-frdm-mcxa153-freemaster-hea

Note: The file selected in the list will be used as default symbol file when the project is opened

Behavior

☐ Let the user select initial symbol file when project loads

☑ Prompt to reload symbols when symbol file changes and show missing symbols

[✓ Always] Except after initial project load

4. Visualize Data From Heart Rate Project
   The NXP software team has included a default visualization for the Heart Rate project. The demonstration showcases the different styles for project data to be shown.

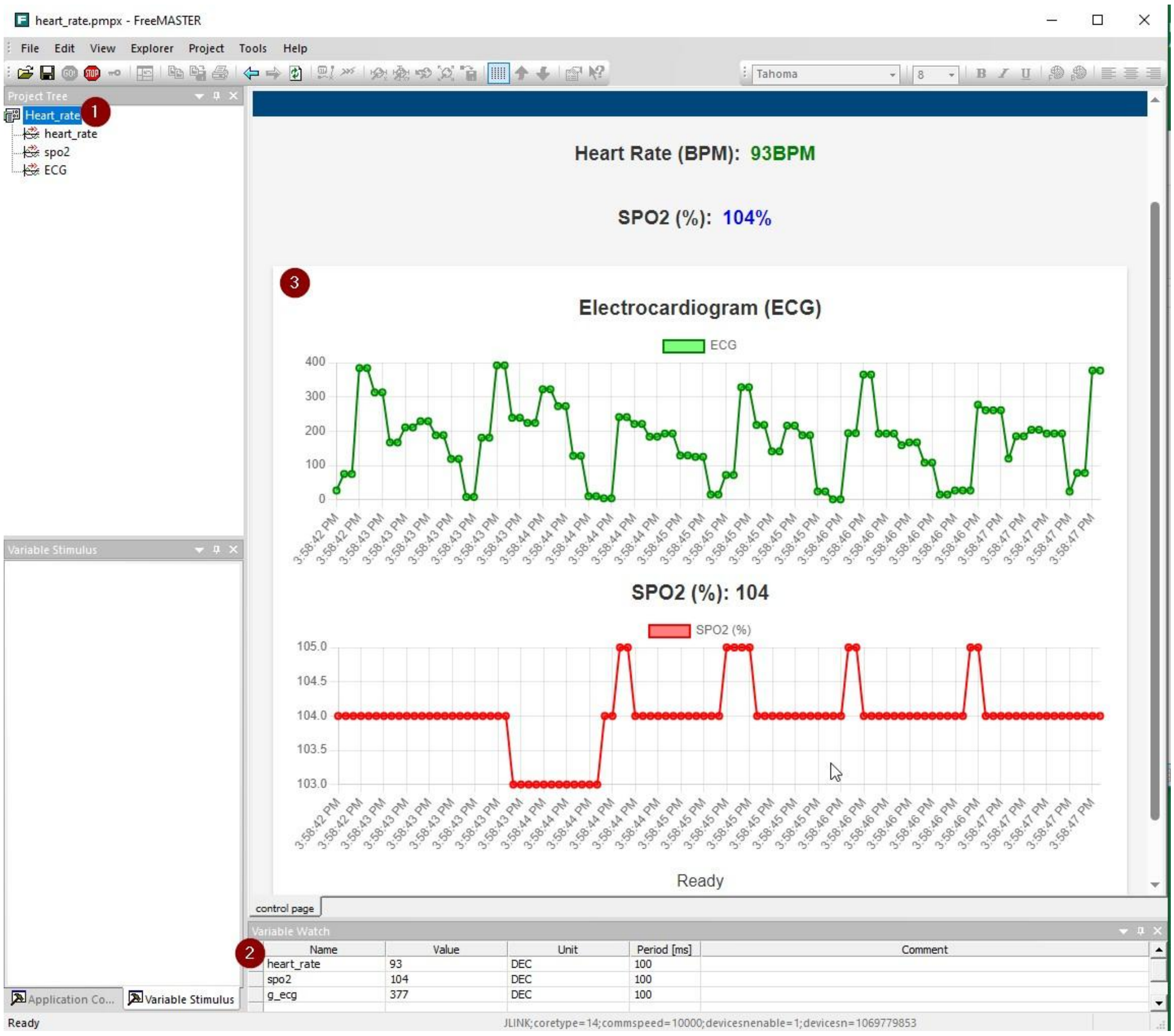
The following visualization settings are preset for the Heart Rate project:

- **Welcome HTML Page**: The HTML Pages (Under Options) points to welcome.html file. This provides structured web view for displaying elements. Beyond scope of this lab, but .html file can be reviewed to see how target values/charts are referenced in html.

- **Oscilloscope Visual**: View plots the values of a project variable. The plot axis are configured for scale and color. Heart Rate, SPO2 and ECG are configured.

- **Variable Watch Table**: After variables are configured to be tracked, they can be added to this table view.
  Click GO icon on the menu bar to initiate the project data visualization! The following points are highlighted for the FreeMASTER output.

  1. Clicking on the elements listed under the Project Tree changes the view to the specific Oscilloscope Visual.
  2. View the captured values for the variables in a Table view.
  3. Visualization of data organized based on the layout defined in the Welcome.html.



Heart Rate (BPM): 93BPM

SPO2 (%): 104%

Name	Value	Unit	Period [ms]	Comment
heart_rate	93	DEC	100	
spo2	104	DEC	100	
g_ecg	377	DEC	100	

Ready

JLINK;coretype=14;commspeed=10000;devicesnenable=1;devicesn=1069779853

NOTE: You have now successfully completed this lab exercise. To learn more about data visualization features using FreeMASTER please visit: FreeMASTER Community Resources