



Flexible and Rapid Development with MCUXpresso

Data Visualization with FreeMASTER

Objectives

In this lab, you will learn:

- How use Application Code Hub to import an example into the VS Code workspace.
- How to build, clean, debug, and run the example.
- How to use an elf file in FreeMASTER to visualize data at run-time.
- How to select variables to watch.
- How to create a virtual oscilloscope to view variable data.

Magnetic Field Data Tracking Lab

The NXP Application Code Hub provides a complete example of how to use the MCXA153 microcontroller in a magnetic field tracking application. This lab will walk through the steps to import, modify, build, program and debug the example. The final section of the lab shows how to configure a FreeMASTER project as a data visualization tool for the acquired sensor data from the FRDM-MCXA153 development board.



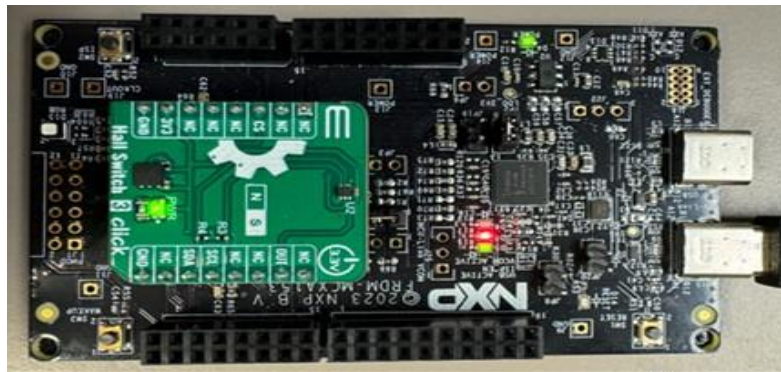
1. Verify Installation / Hardware Setup

The prework instructions should be completed before starting this lab. The instructions walk through the installation for the required software.

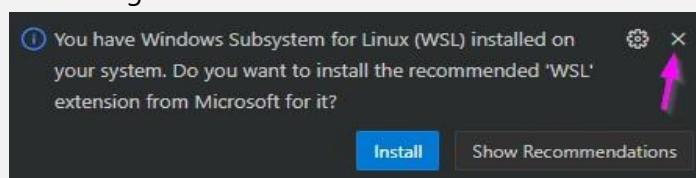
After finishing, the system should have the following:

- [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.
- [MCUXpresso 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 Hall Switch 3 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:



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



2. Import Project with Application Code Hub

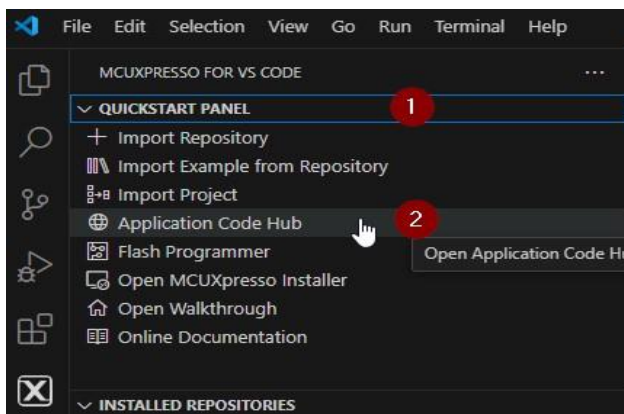
The Application Code Hub (ACH) repository enables engineers to easily find microcontroller software examples, code snippets, application software packs and demos developed by NXP in-house experts. This space provides a quick, easy and consistent way to find microcontroller applications. Find more information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

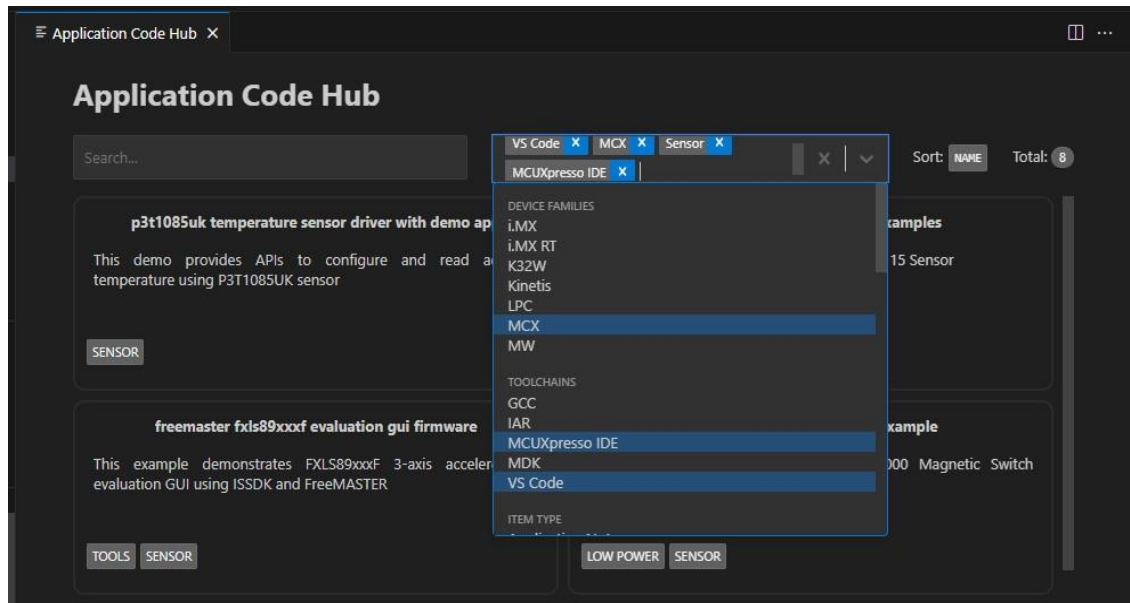
1. Go to the Quick Start Panel
2. Select Application Code Hub





3. Filter Visible Examples (MCX + Sensors)

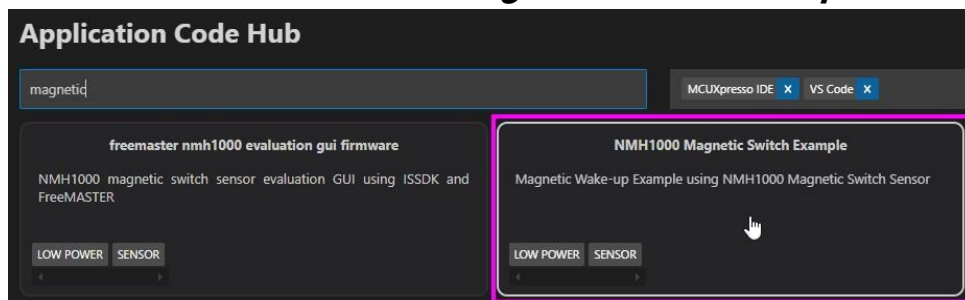
Go to the filter section next. It is a dropdown menu next to the Search bar and select two additional filters. Select **MCX** in the Device Families Section and **Sensor** in the Categories section of the filters.



4. Search for Keywords in Examples

Search for the keyword '**magnetic**'.

Select the demo ***"NMH1000 Magnetic Switch Example"***.

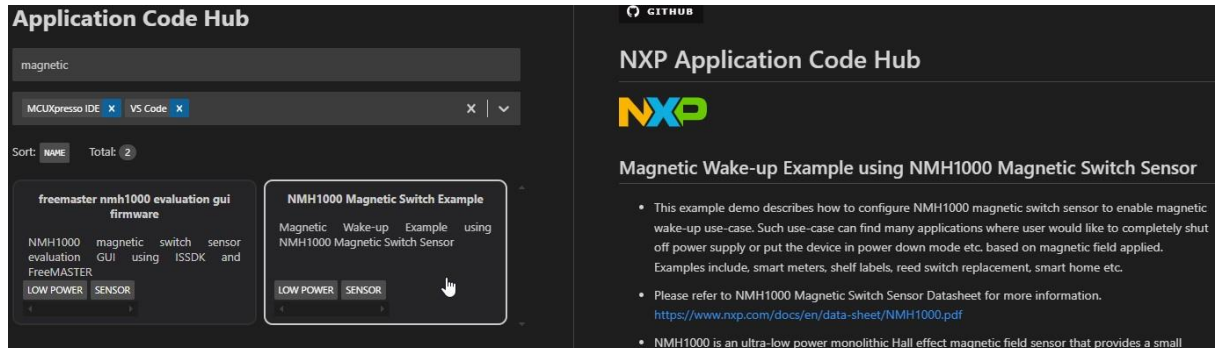


5. Read Overview of the Example

The Application Code Hub provides a consistent Readme Overview for every project. The NMH1000 Magnetic Switch Example 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.



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 **MagData-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.

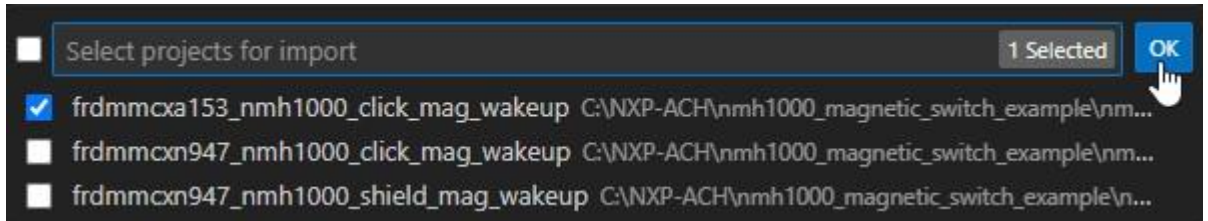
If a valid project is not available, the wizard only displays **Import Repository**, to allow a code repo, without a project, to be added to workspace.



8. Select Detected Project(s)



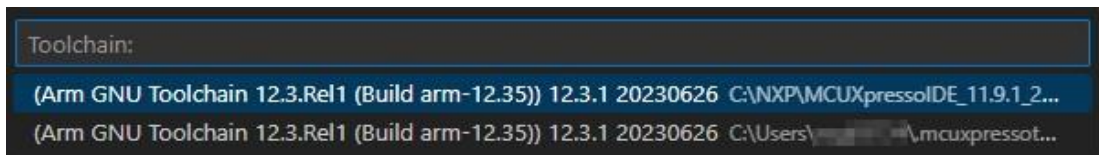
The import wizard will scan the example repo and list valid projects that were discovered. This allows the user to select only the projects they want created. Select the **mcuxpresso** project listed at the top of the VS Code window below the Workspace Search bar.



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.




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 Magnetic Switch 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

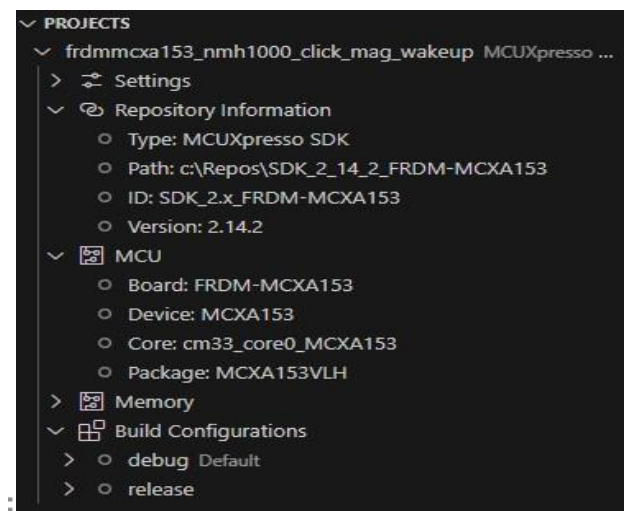
The MCUXpresso for VS Code extension includes a PROJECTS section to help users access useful project information.

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).





2. Associate the Required MCUXpresso SDK

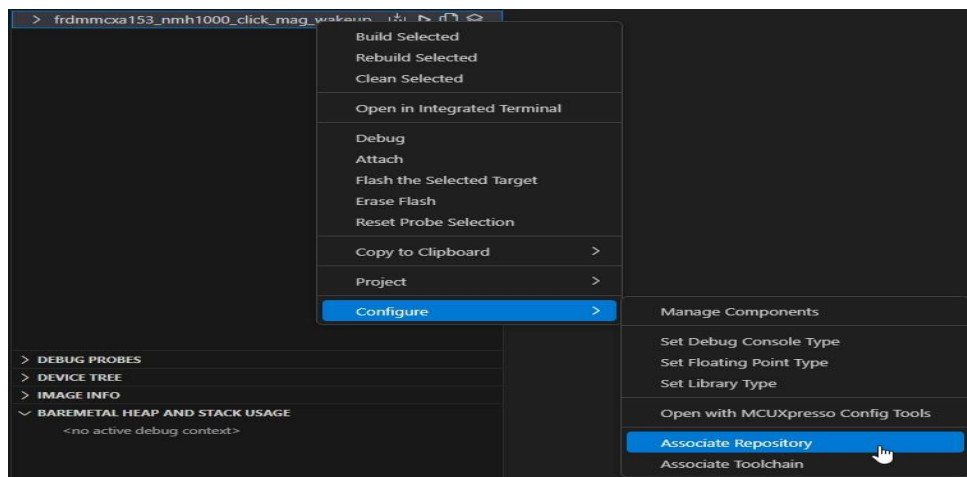
Sometimes, Projects imported/converted from the Application Code Hub do not automatically have a valid SDK associated to them. This can be caused by the required SDK not being installed and imported into the workspace.

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.

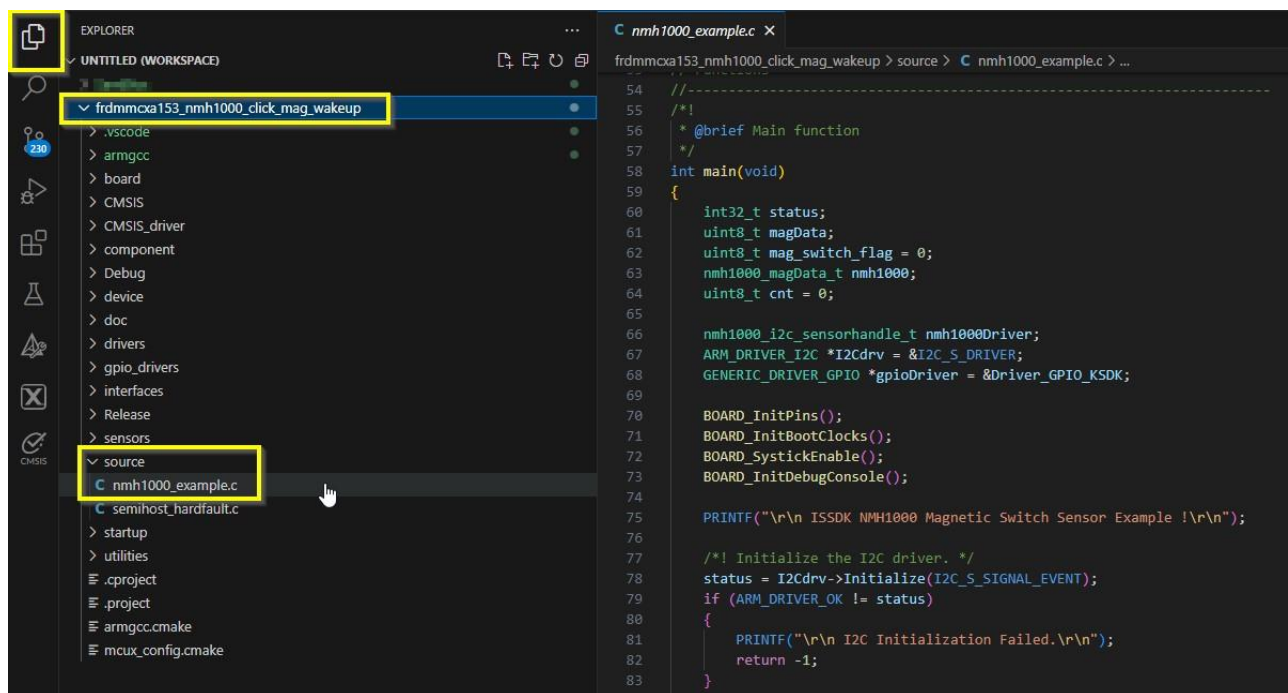


3. Working with Source Files

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 the project name and you will find the source files.
- In this example the main function is in the file **nmh1000_example.c**. Click on this file to reveal the source code.



NOTE: To track data in FreeMASTER, variables must be declared globally.



- Navigate to line 61 in the main function. Comment out the variable declaration **uint8_t magData**;
- Declare **magData** as a global variable above the main function.
- Here is how the modified code should look:

```
//-----  
--  
// Global  
//-----  
--  
uint8_t magData;  
  
//-----  
--  
// Functions  
//-----  
--  
/*!  
 * @brief Main function  
 */  
int main(void)  
{  
    int32_t status;  
    //uint8_t magData;  
    uint8_t mag_switch_flag = 0;  
    nmh1000_magData_t nmh1000;  
    uint8_t cnt = 0;
```

- Press Ctrl+s to save.

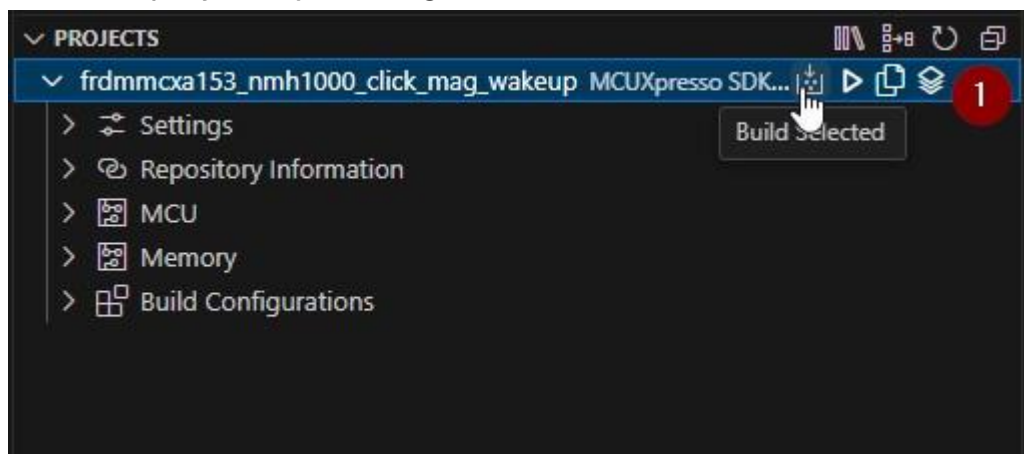


4. Build the Application

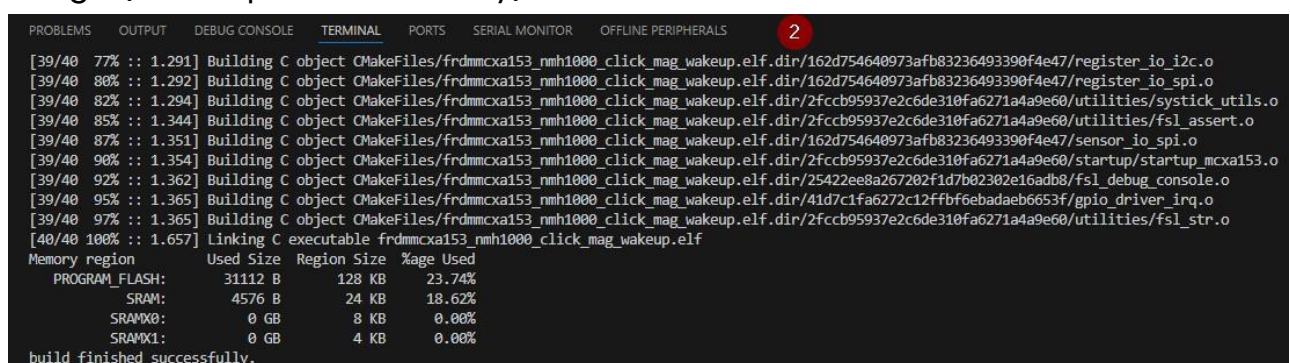
The *frdmmcxa153_nmh1000_click_mag_wakeup* project needs to build the application image. After the code builds without any errors, the application can be run on the FRDM board.

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).



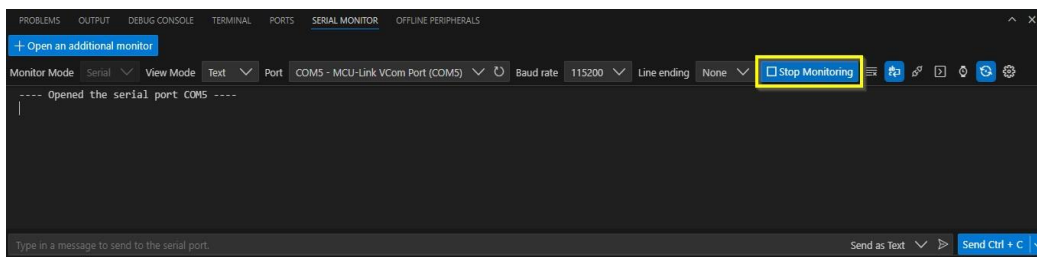


5. Connect Serial Monitor to the board

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.

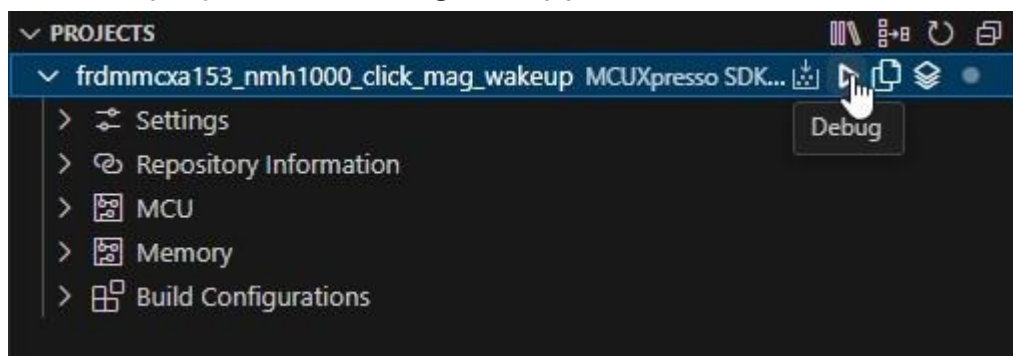




6. Flash/Debug the Application

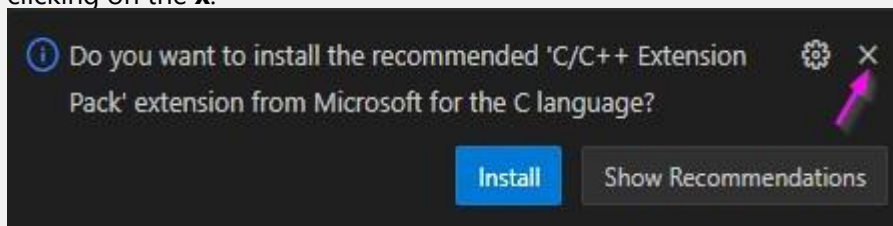
This section uses the on-board debugger to connect to the MCU, and program the flash. LinkServer from NXP manages the GDB server for communicating with the NXP MCULink on-board debug probe. It includes support for flash programming.

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



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**.





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 application to launch inside main().



3. View sensor information in the Serial Terminal The magnetic switch application will display the following information in the serial port.

```
---- Opened the serial port COM5 ----  
  
ISSDK NMH1000 Magnetic Switch Sensor Example !  
  
Successfully Initialized NMH1000 Sensor  
  
Successfully Applied Sensor Poll Configuration.  
  
Waiting for Magnetic Field to Change
```

Move a magnet over the sensor on the click board. The following should be displayed in the SERIAL MONITOR tab after a change in the magnetic field is detected:

```
Waiting for Magnetic Field to Change  
Mag Threshold (50) Crossed: Mag Out = 99  
Mag Wake Up Detected
```




7. Communication with FreeMASTER

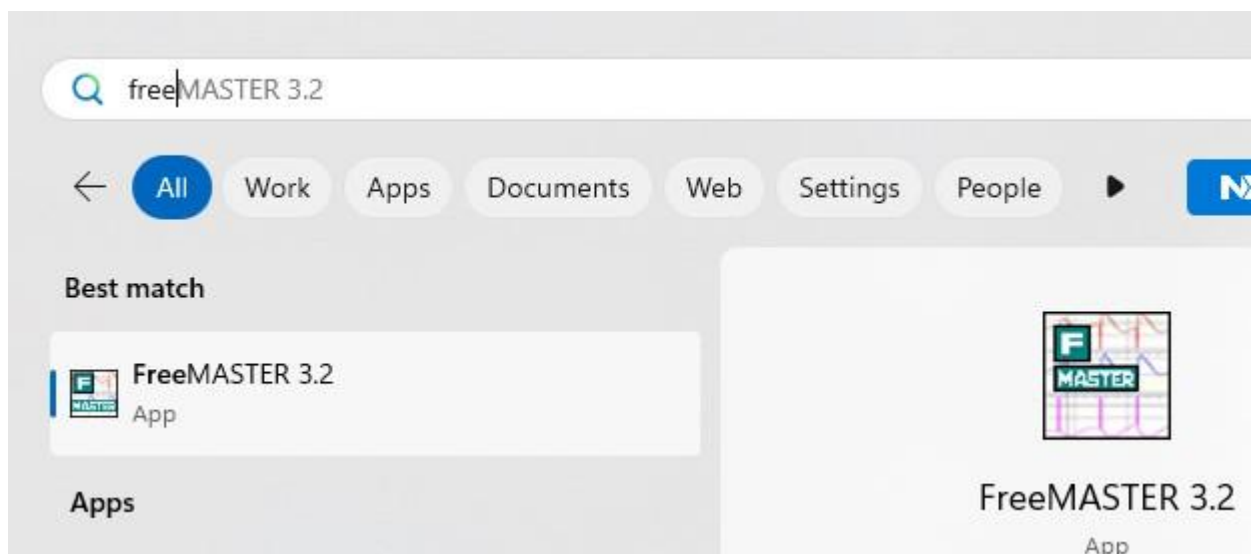
FreeMASTER is a standalone application provided by NXP to help developers visualize, monitor and manipulate data available from their projects.

The following steps will demonstrate how to configure FreeMASTER for any project.

1. Stop any debug sessions.



2. Launch FreeMASTER

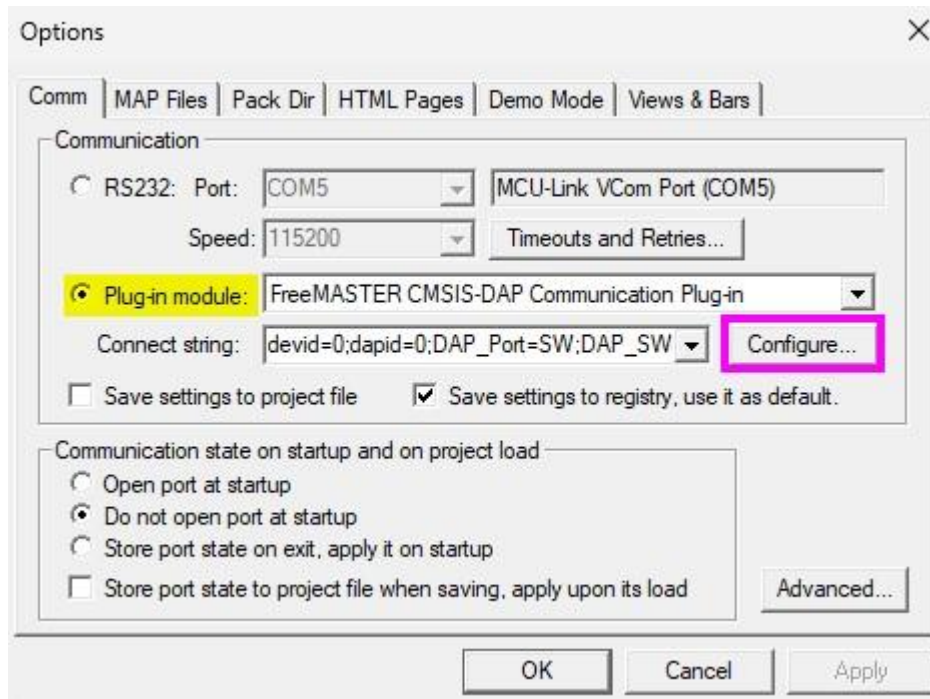


3. Start Communication with the target.

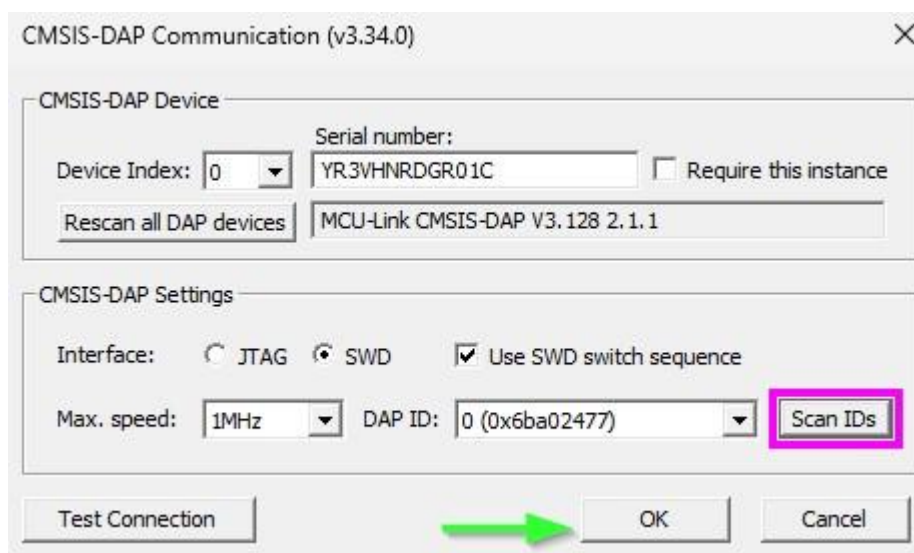
Click on **Project** on the top. Select **Options** and verify the options settings shown below.

NOTE: Your Comm port may be different.

Click **Configure**.

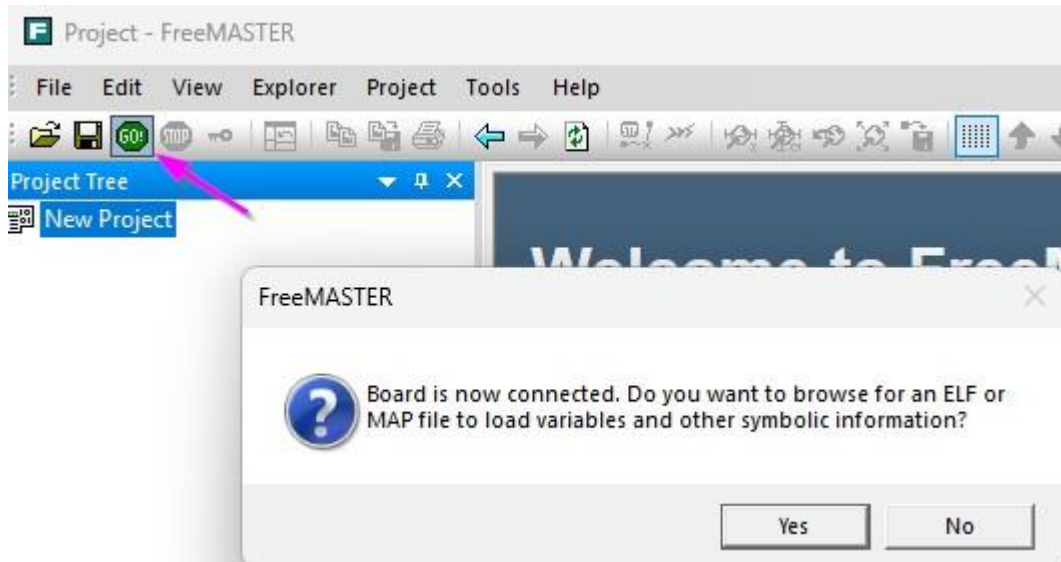


In the new pop up window click **Scan IDs** and the **DAP ID** field should populate. Click **OK** on both popups and navigate back to the main screen.





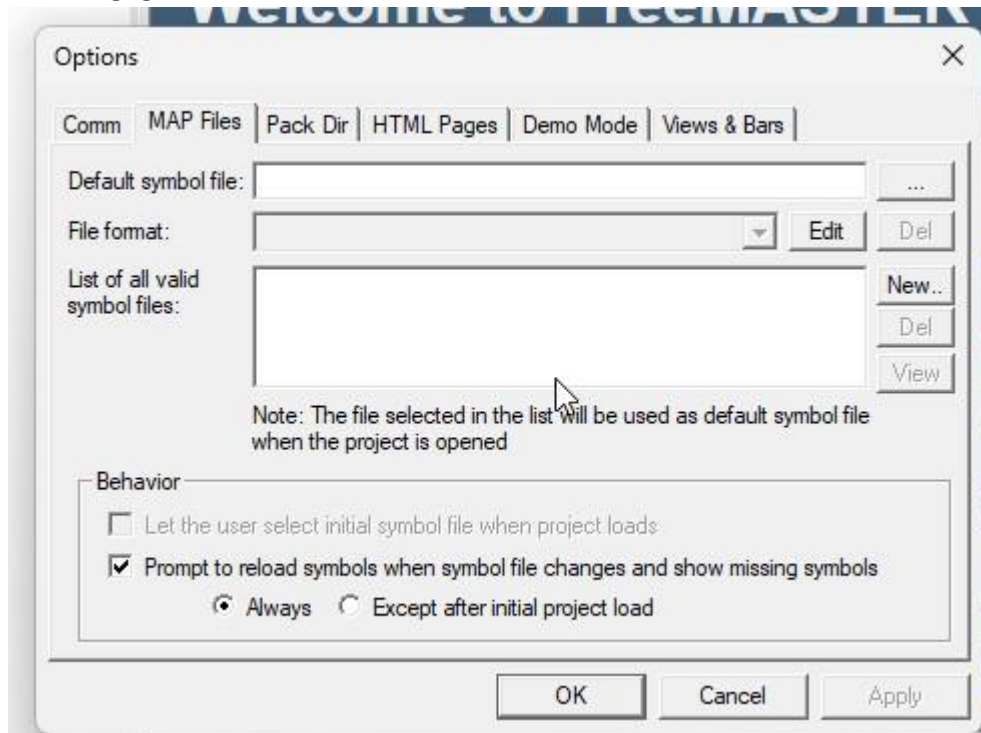
Click the green **GO!** button in the main window to start communication.



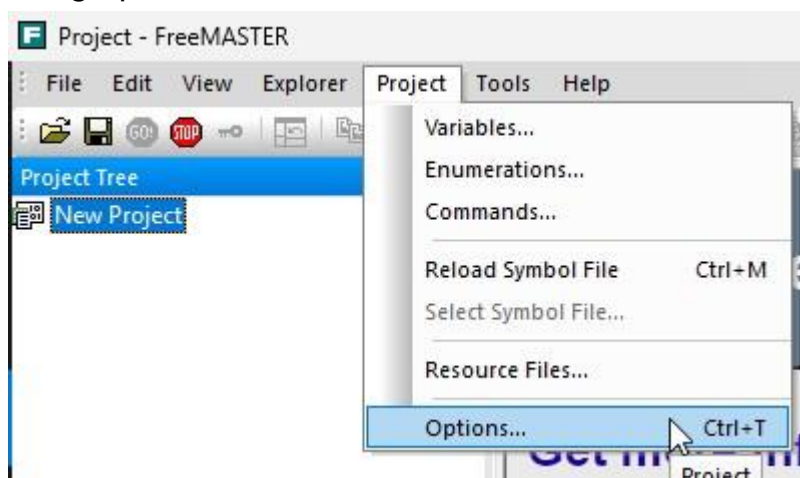
8. Importing MAP Files

The following steps will demonstrate how to import the necessary MAP file.

1. We can access MAP files menu by clicking *Yes* in the pop-up after clicking go.

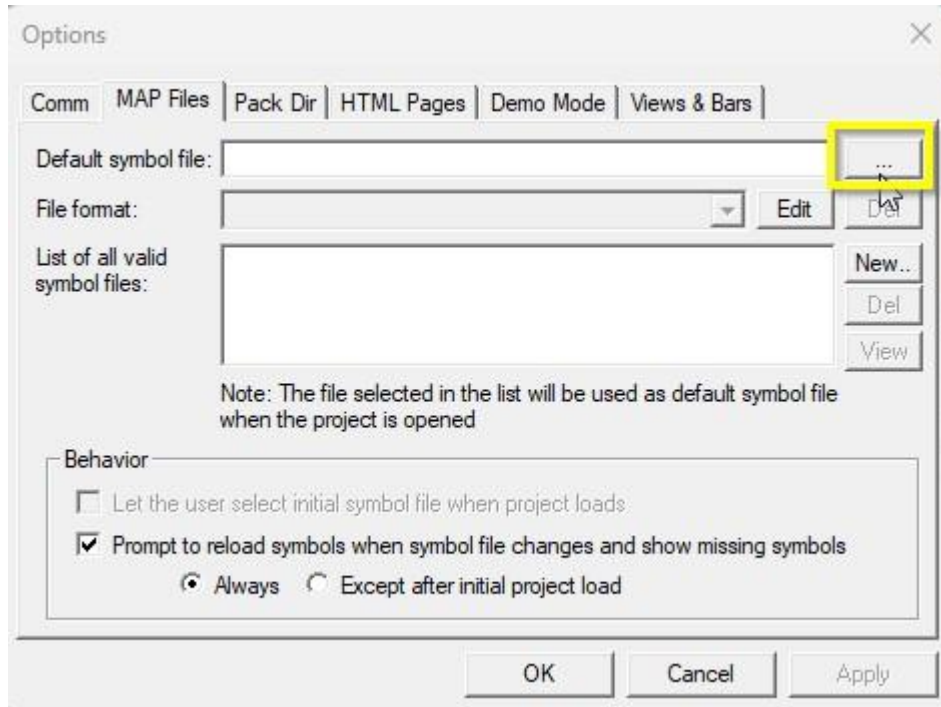


2. Otherwise, we can navigate to *Options* in the *Project* tab which will bring up the same menu.

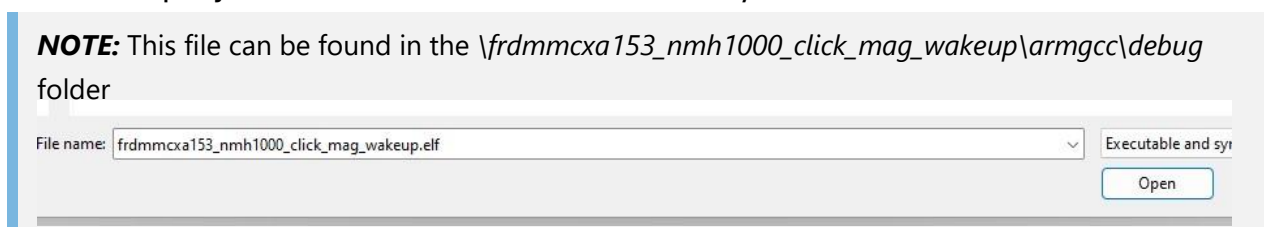




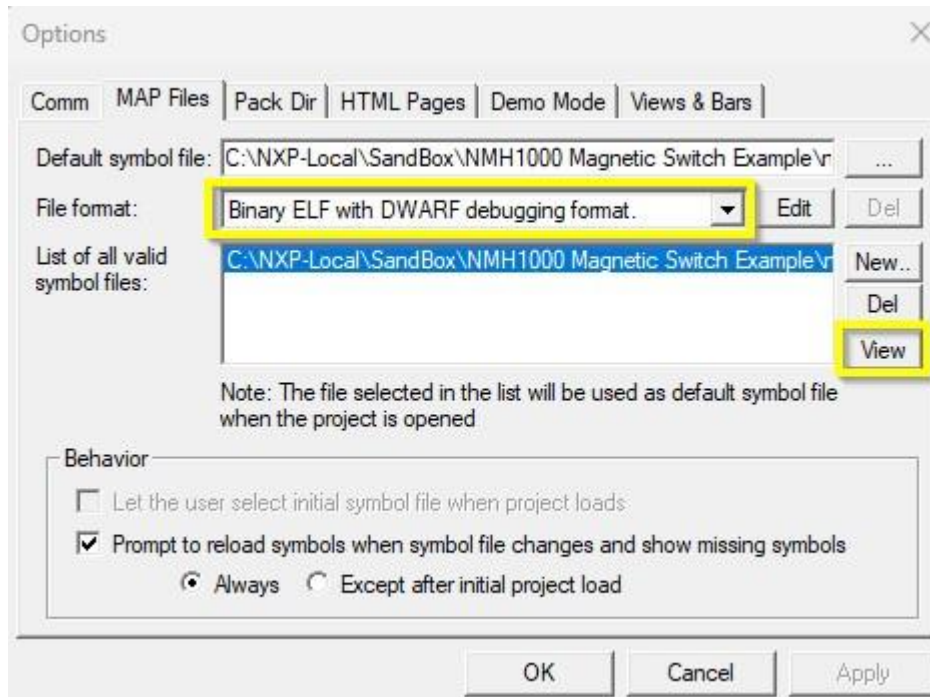
3. Click on the button with three dots to the right of the *Default symbol file* field



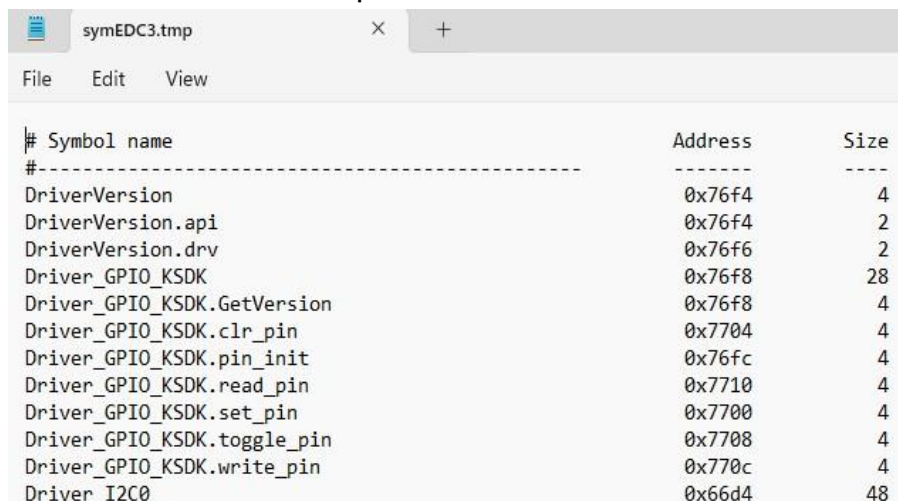
4. The File explorer should pop-up. Navigate to the directory where you saved the project. Select the *.elf* file and click *Open*.



5. Once the file is imported, ensure that the selected file format is *Binary ELF with DWARF debugging format*. Click *View* to verify that the project variables are read correctly.



After clicking *View* a text file containing variable information should open.



Once verified that this file is populated, the text file can now be closed.

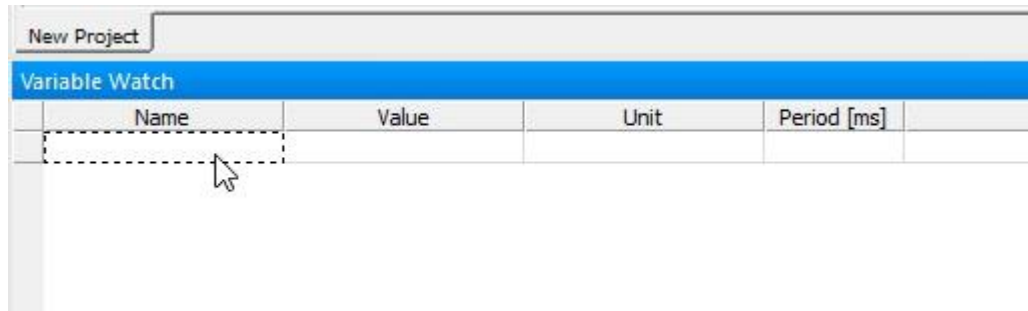
6. After examining the text file we can click *OK* in the *Options* menu.



9. Adding Watch Variables

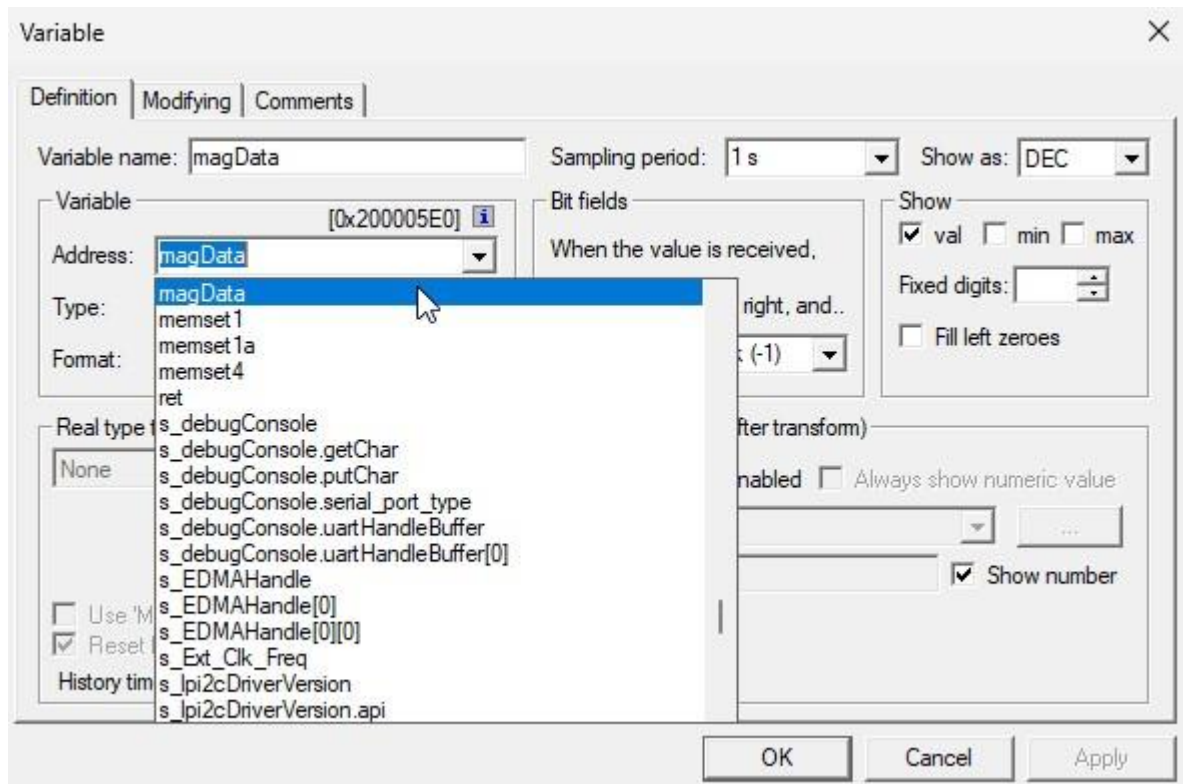
This section will demonstrate how to add project variables that the user desires to track.

1. Navigate to the *Variable Watch* view at the bottom. Double click on the empty field under *Name*.



The variable options menu should pop-up.

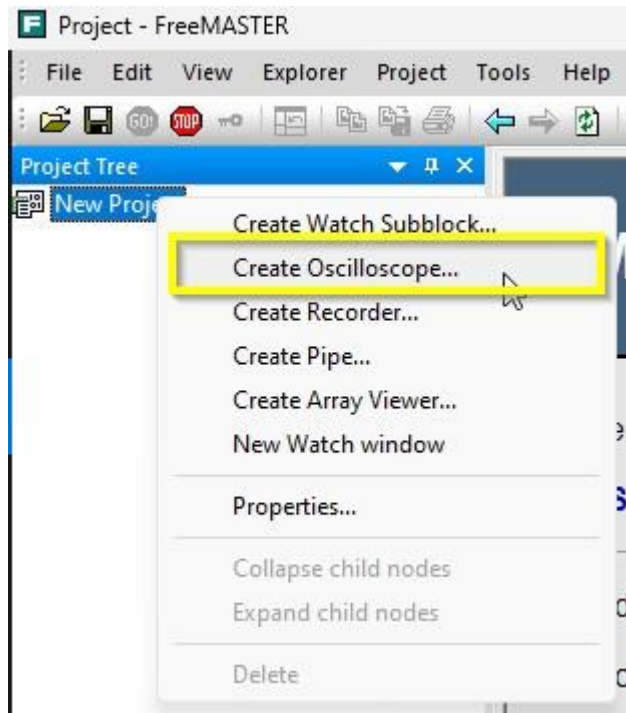
2. Open the *Address* dropdown and find the *magData* variable.





3. This variable is declared as *uint8_t* in the code. Therefore, select *Unsigned int* in the *Type* field. Select 1 in the *Size* field. Choose the *Sampling period* as 1 s. Show as DEC. Use the default settings for all other fields. Click *OK*.

4. To create a component to visually track the data, navigate to the *Project Tree*. Right click on *New Project*. Select *Create Oscilloscope*



The *Oscilloscope Properties* window should pop-up.

5. Give the oscilloscope a name relevant to the variable name. Try *Magnetic Field Data* and keep the default settings.

Oscilloscope Properties

Main Settings | Variables | Data Capture

Name:

Description URL: ...

Oscilloscope Properties

Period: (0 = maximum)

Buffer: points/subset

☐ Ignore identical consecutive samples

Graph Legend

☒ Legend visible ☒ Top ☐ Bottom

☒ Inside graph ☐ Left ☐ Right

Grid

☒ Horizontal

☒ Vertical

Graph Type

☒ Time graph

☐ X-Y graph

☐ Polar graph

Graph Setup

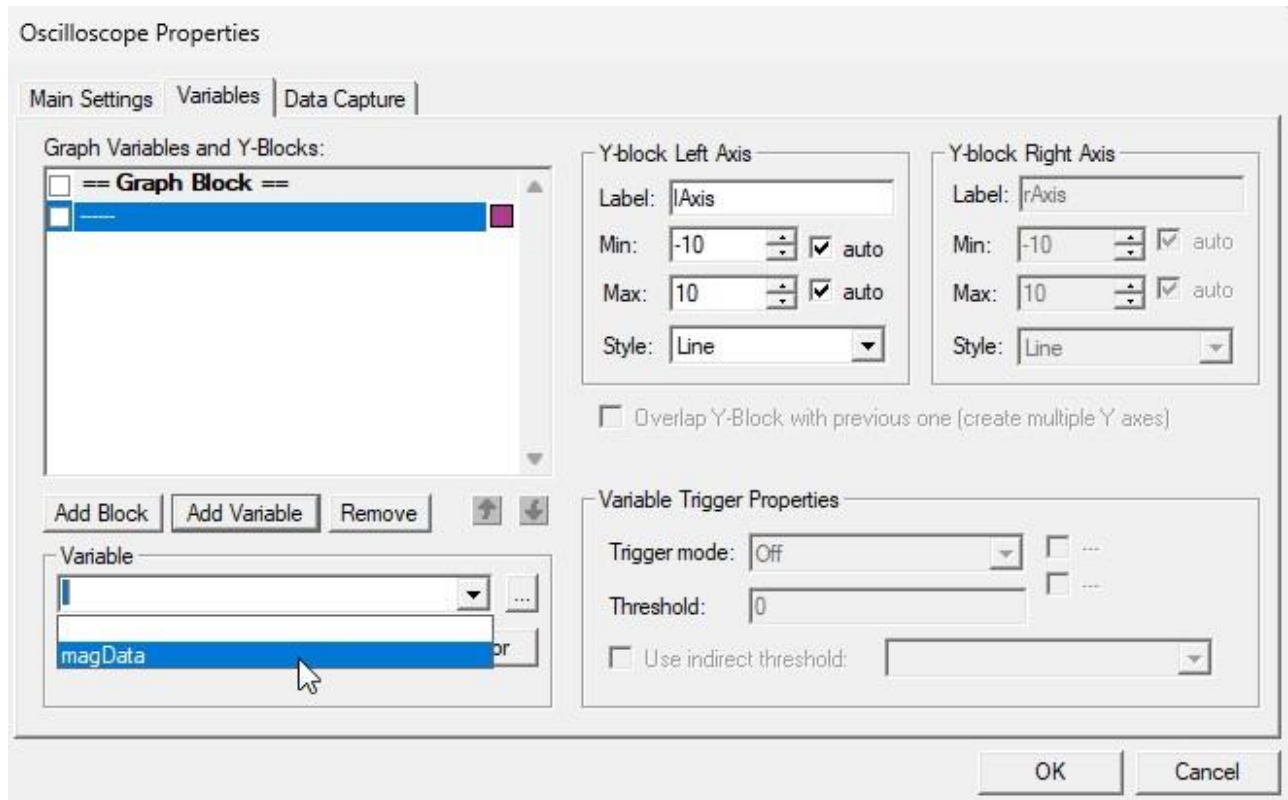
X-axis label:

X-axis units: ☒ Append units name to the label

X-axis width: ☒ Auto-scale when lower than width

OK Cancel

6. Click on the *Variables* tab and click on *Add Variable*. Variables that have been added to the Variable Watch list will enumerate, select *magData*.

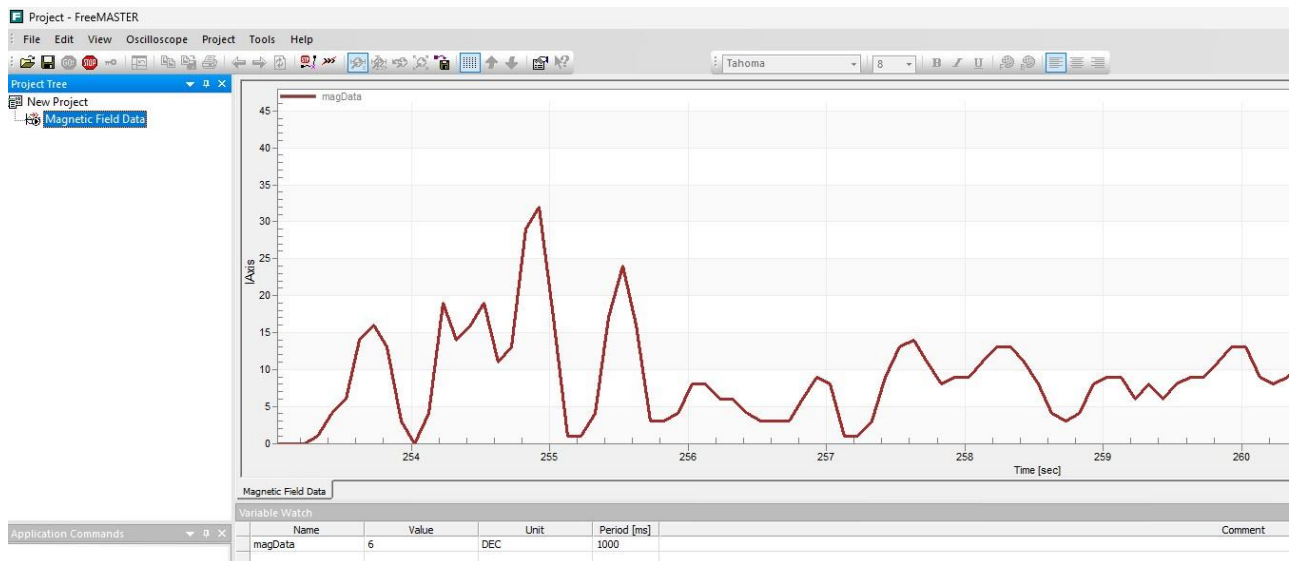
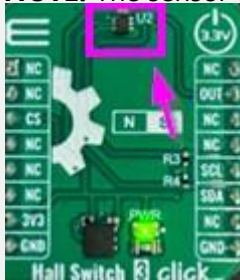


Use the default settings for all the other fields and click *OK*.

7. The configured oscilloscope will appear in the *Project Tree* and data will begin plotting in real time.
8. Take a magnet and move it around the sensor, the plot should capture the changes in the magnetic field.



NOTE: The sensor on the click module is shown here:



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