Using data driven style

In this exercise you get a simple test case file that you need to convert to use data driven style and to improve the test cases a bit.

The test case file communicates with capture software of the PSoC board. The device under test is the software on the LPCXpresso board. It is a morse code transmitter software that converts text to morse code and transmits them at user specified speed. The morse transmitter accepts commands and text in both lower and upper case. Letters from A-Z and numbers 0-9 have a morse code and a space is “transmitted” as a pause between words. Characters that don’t have a defined symbol are converted to “X”. For example, command “`text hello, world!`” will be received as “`HELLOX WORLDX`”. Your test cases must verify that undefined characters are transmitted as specified above and that text containing multiple words are also received correctly.

The capture board runs morse code decoder software that receives and decodes morse codes and communicates with a test library written in Python.

You can find the morse code transmitter sources here: https://gitlab.metropolia.fi/lansk/morsesender. Note that when you check out the project you need to add the projects to MCUxpresso workspace.

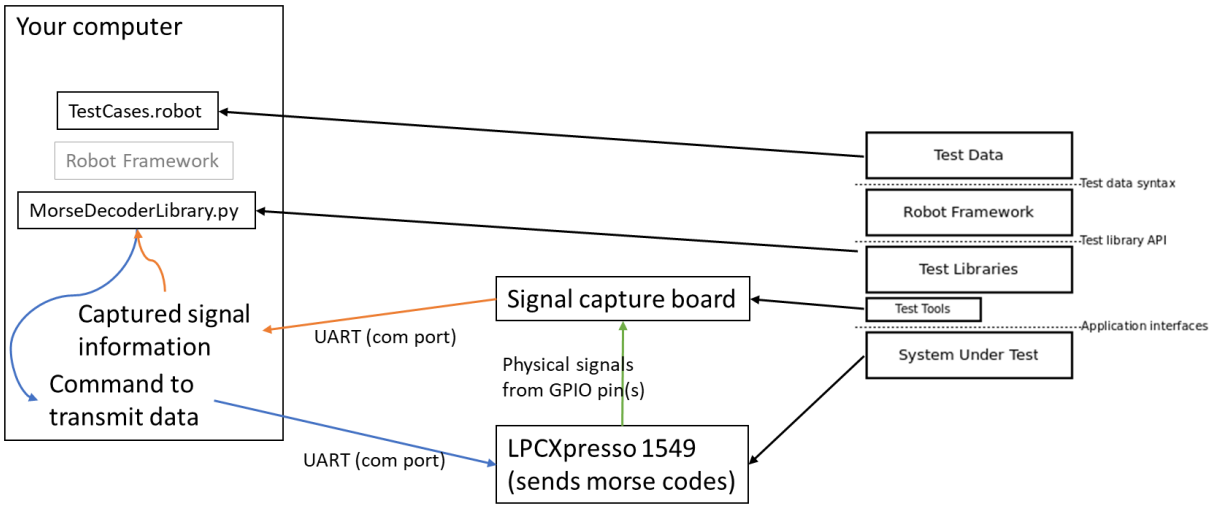
You can find the morse decoder sources here: https://gitlab.metropolia.fi/lansk/morsedecoder. The precompiled .hex file can be found in the archive file attached to this exercise.

To run the examples, you need to install pyserial package:

```
pip install pyserial
```

You also need to edit the comport names to match the ports in your system.

If you get an error about non-existing keywords, it means that the MorseDecodeLibrary failed to initialize properly. That can happen if pyserial package is not installed or if the comport names don’t match the ports in your system.

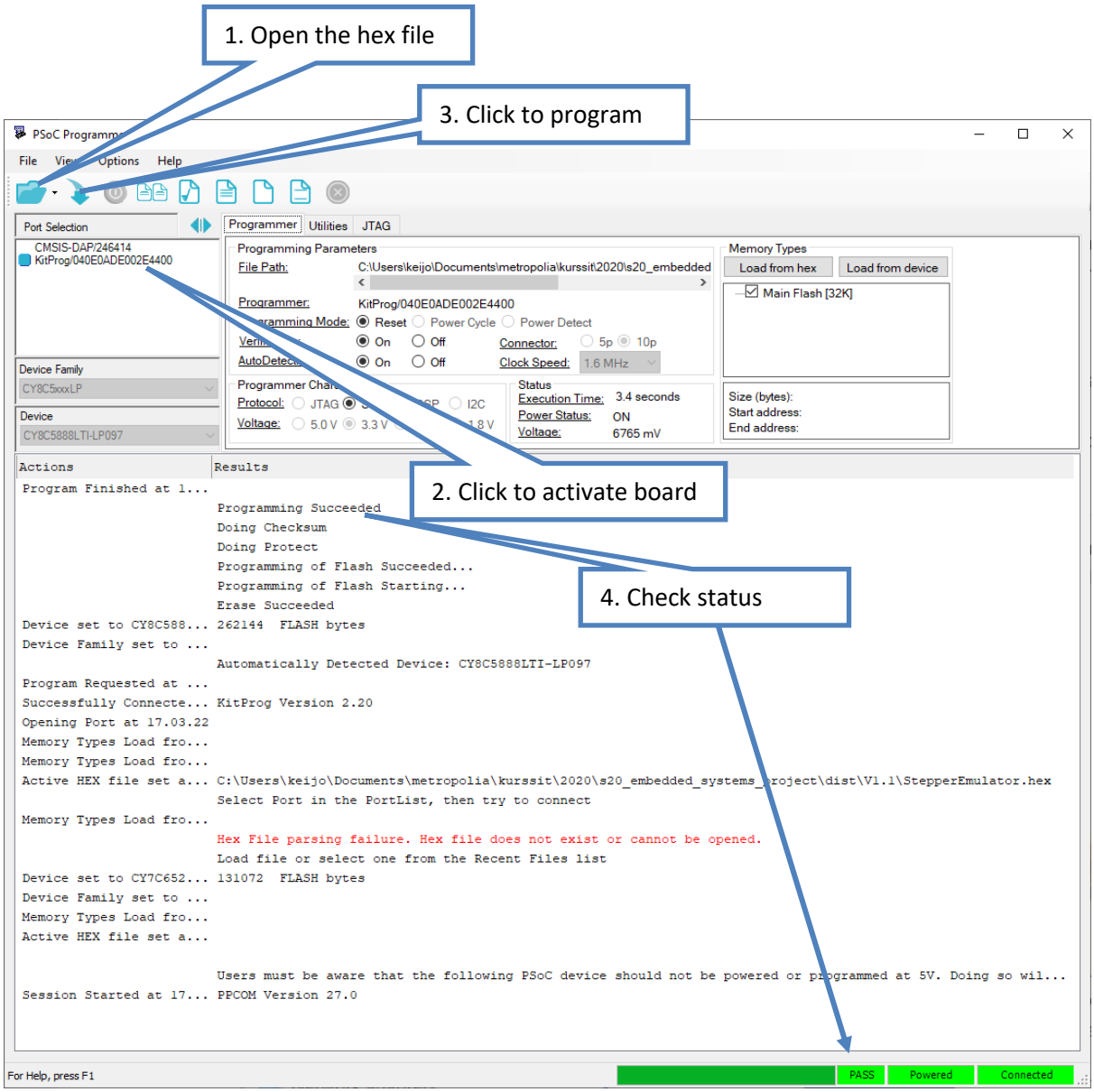


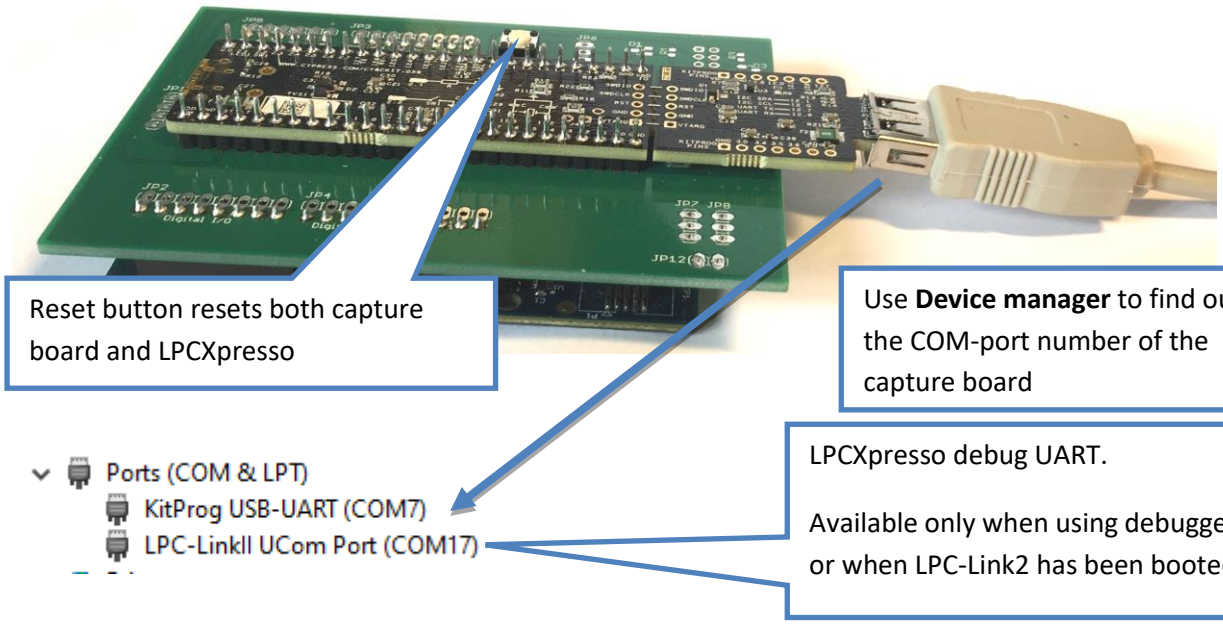
Programming the signal capture board

On the following pages you can find how to program the capture board. Unfortunately, PSoC programmer runs only on windows.

When the board is programmed, we use Kitprog USB-UART to communicate with the board. Kitprog USB-UART uses standard USB drivers and has been tested to work also on OSX and Linux. Once the board has been programmed the rest of the exercise is operating system independent: MCUXpresso IDE is available for Windows, Linux and OSX.

The PSoC-board needs to be programmed with the capture software. The hex-file of the capture software and the programmer can be downloaded from the course workspace. When you install the programmer, PSoC programmer, you get drivers for both the programmer (KitProg) and the USB-UART. USB UART drivers are needed to communicate with the PC-based simulator. PSoC programmer is not needed after the capture board has been programmed.





Reset button resets both capture board and LPCXpresso

Use **Device manager** to find out the COM-port number of the capture board

LPCXpresso debug UART.

Available only when using debugger or when LPC-Link2 has been booted.

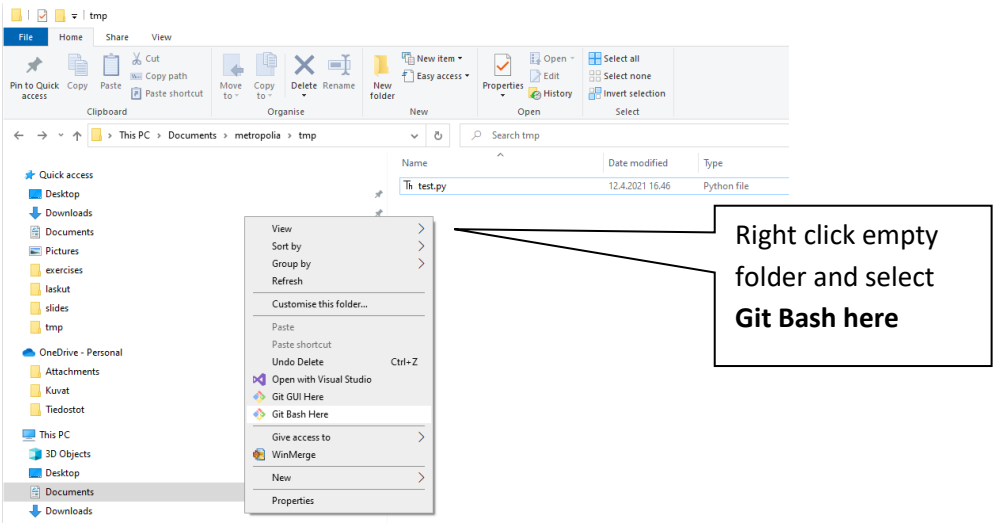
Note that because reset pin of PSoC and LPCXpresso are connected together the PSoC board is reset always when LPCXpresso is reset (by LPCXpresso debugger or by reset button).

If your capture board shows up as a mass storage device press and hold reset for about 5 seconds. When the board is in mass storage programmer mode the green led near USB-connector blinks. Solid green means that the board is in standard mode. USB UART works only in standard mode. Note that mass storage mode can’t be used to program the capture board.

How to import a project from git to MCUXpresso

The following assumes that you have git installed.

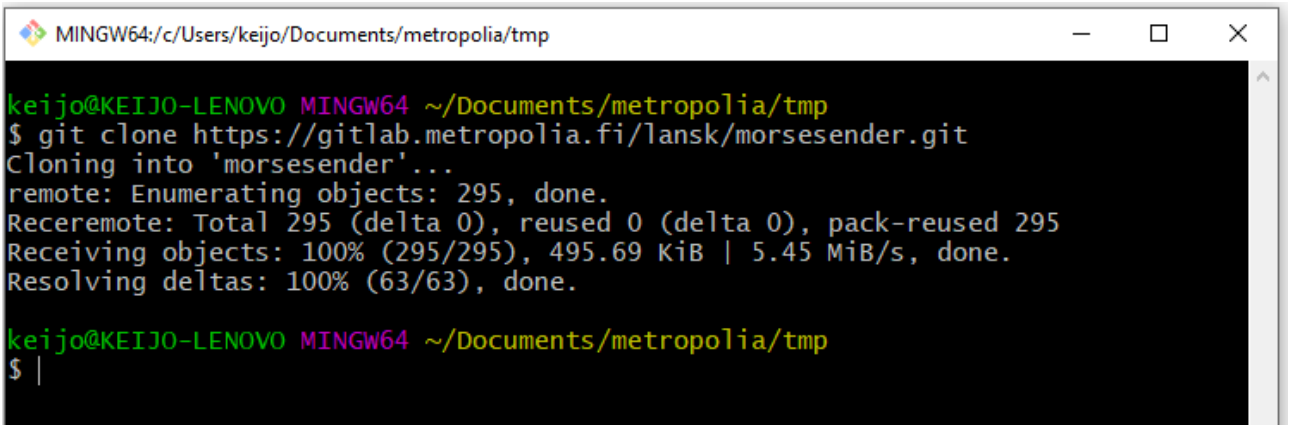
Select a suitable folder for the project. The folder does not need to be empty. When you check out the project from git it will be checked out to a folder with the same name as the project.



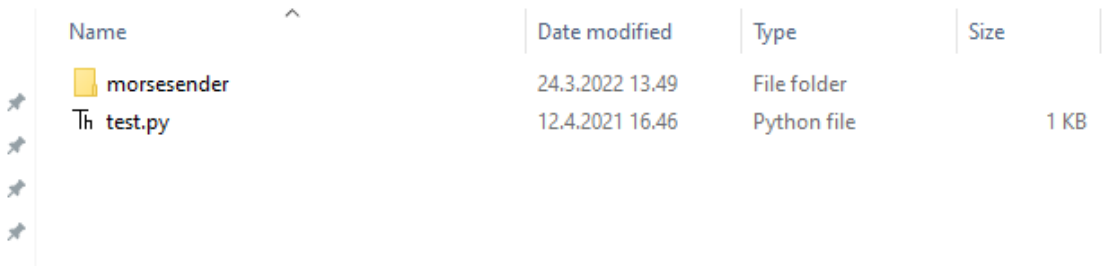
You can also go to the directory manually using your favorite terminal.

Execute command:

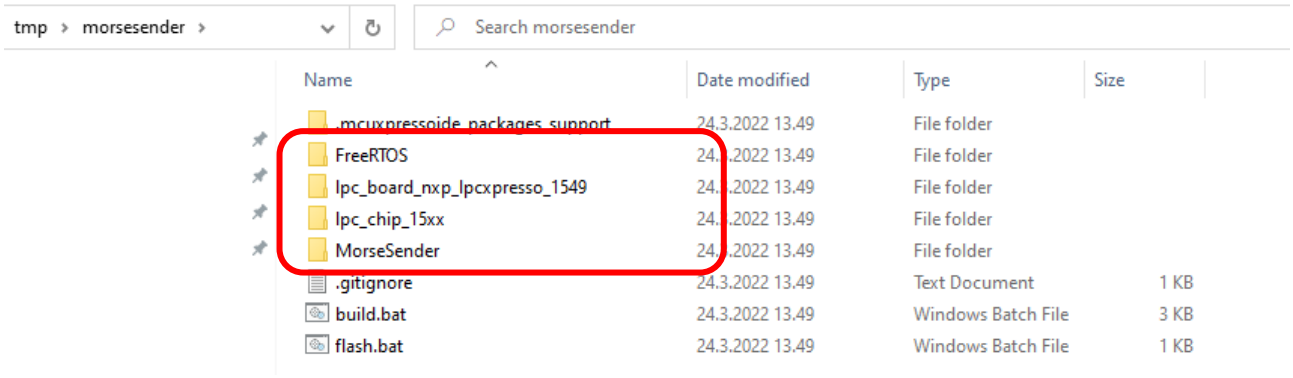
```
git clone https://gitlab.metropolia.fi/lansk/morsesender.git
```



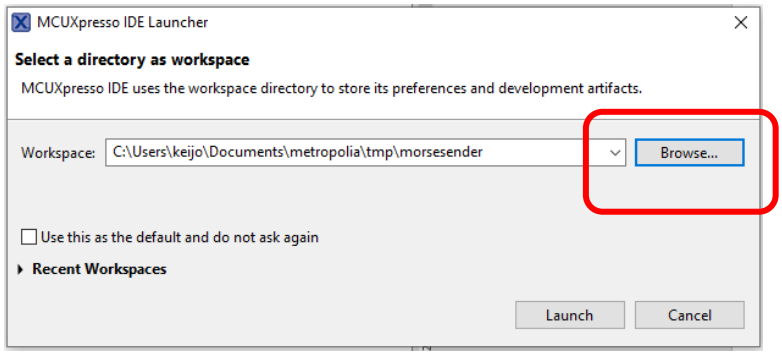
When the command completes you have a folder called **`morsesender`** in the folder where you executed the command.



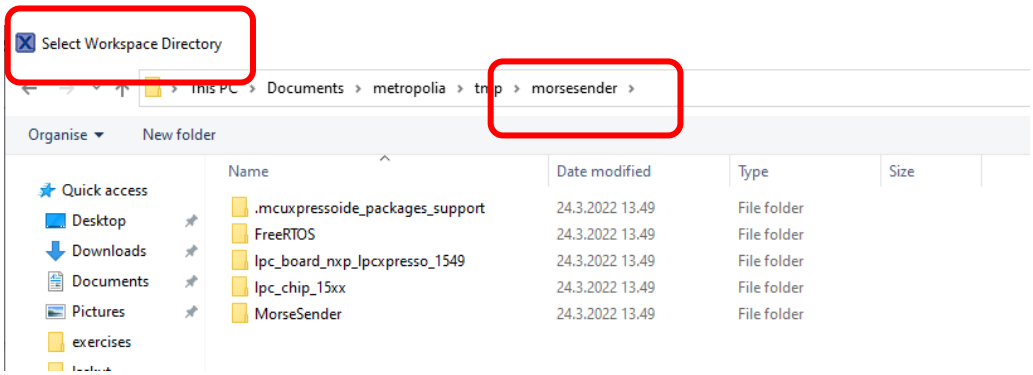
Inside the folder there are four folders that make the morse code sender project



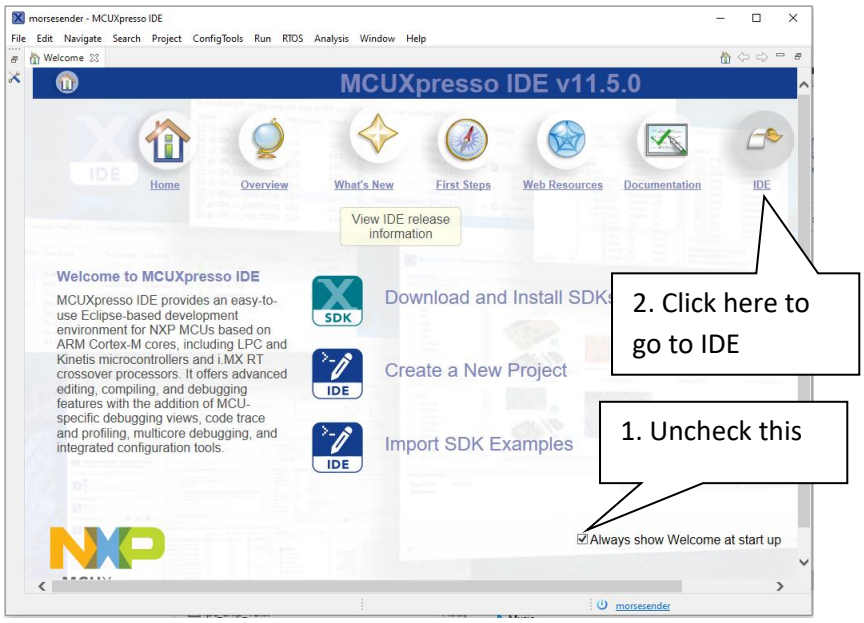
Then launch MCUXpresso. At startup you'll see a dialog that asks you to select the workspace directory. Click browse and go to the folder where you checked out the project code



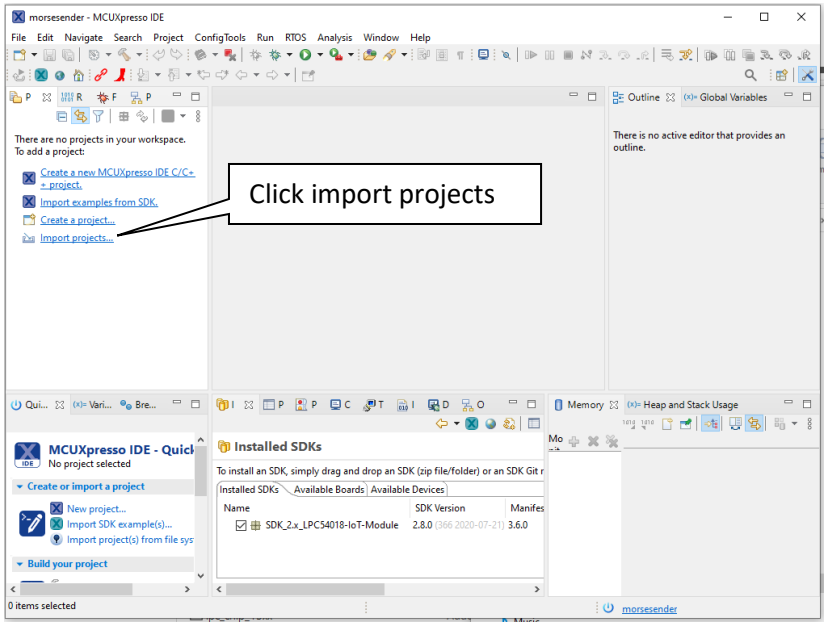
Navigate to the **`morsesender`** folder (see below) and click Select folder.



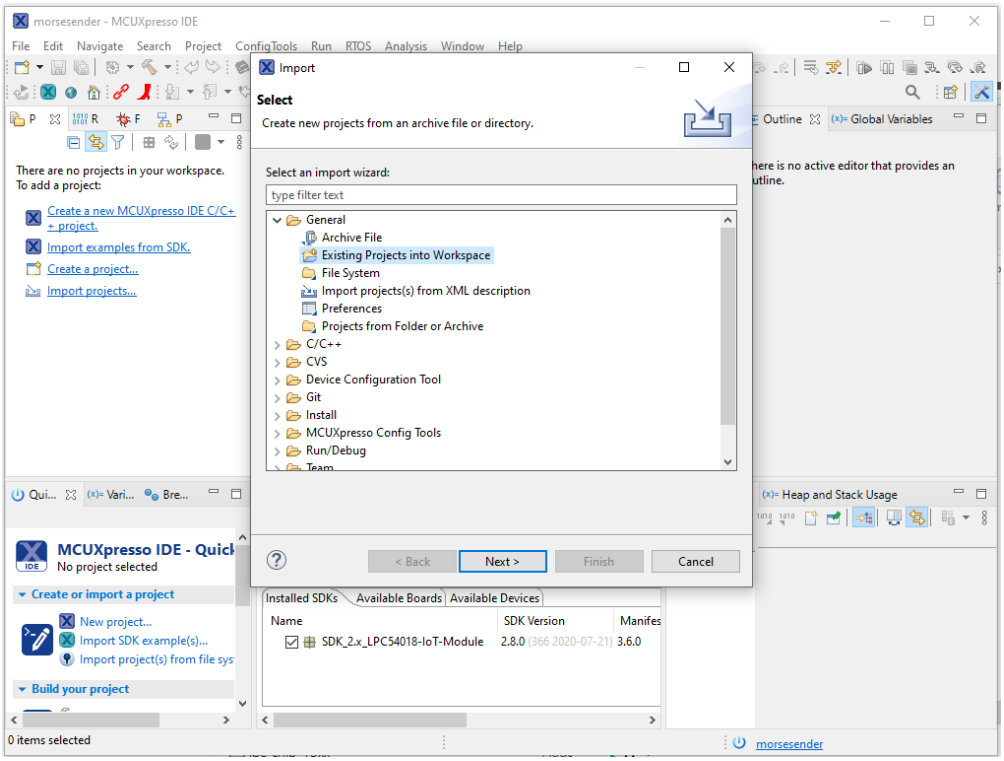
When you have selected the folder, you will see the welcome screen.



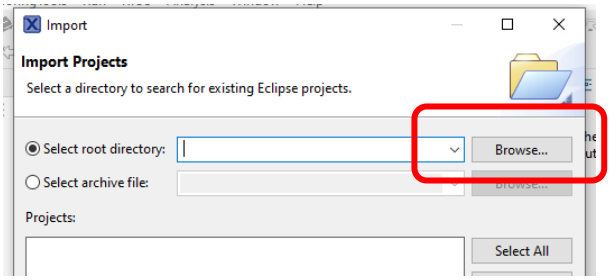
On the first run the project explorer is empty so you need to import the projects to IDE.



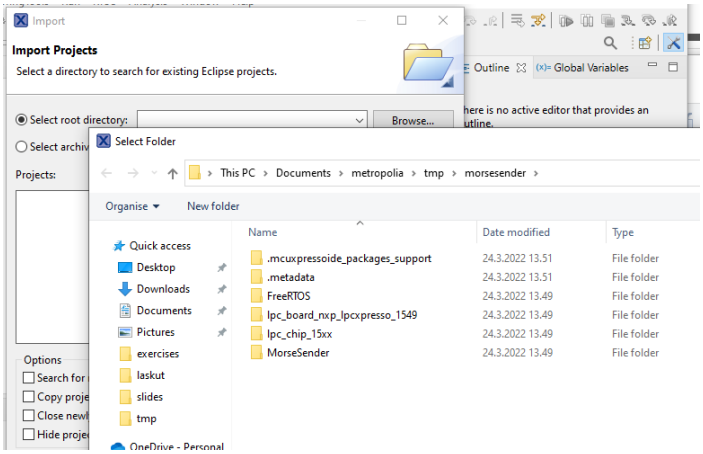
When Import dialog opens, select General / Existing Projects into Workspace and click next.



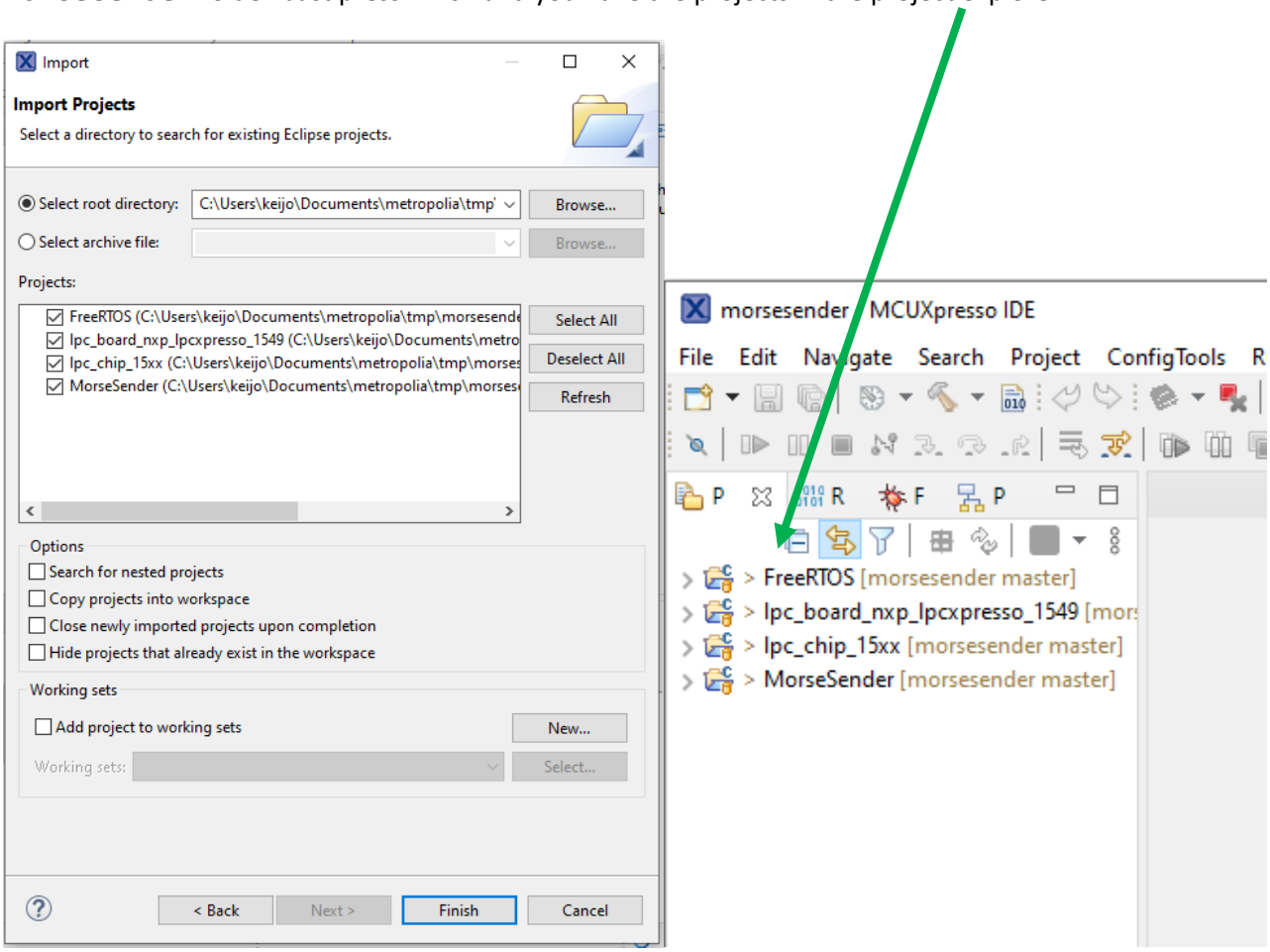
Next you are asked to select root directory. Just click browse.

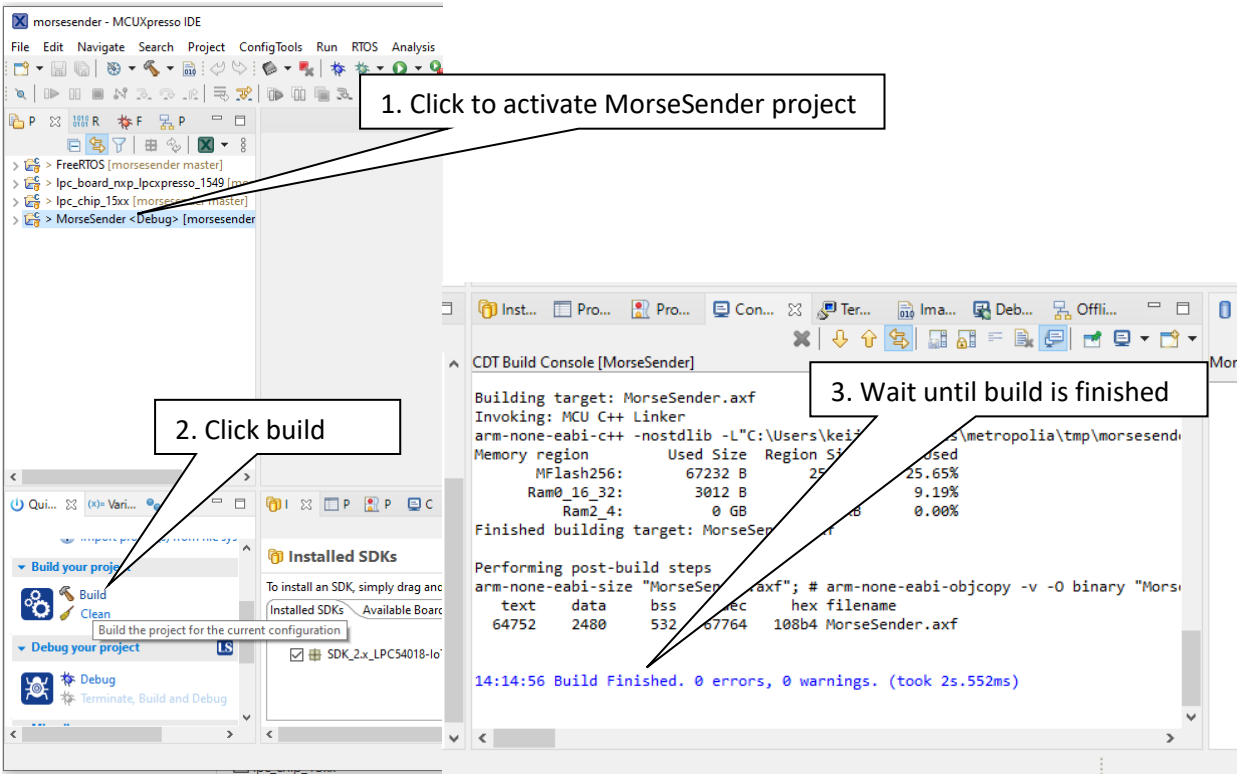


You should be in the correct folder, the one you checked out from git, so just click select folder.

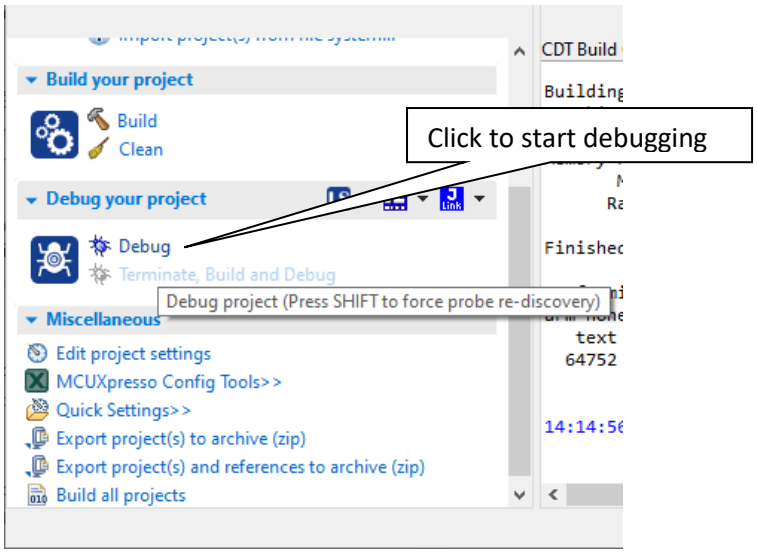


After selecting the folder, you should see the four projects with same names as the folders inside **`morsesender`** folder. Just press Finish and you have the projects in the project explorer.



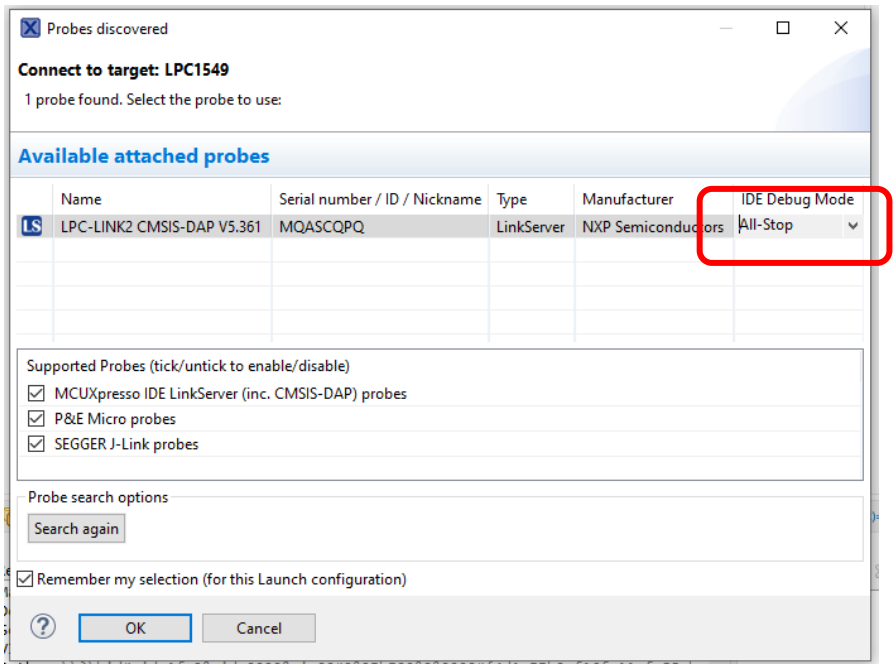


When build finishes, you can debug the program. Plug in the LPCXpresso board before starting debugging. The micro-usb cable must connected to the connector that has Link printed next to it – the connector next to the reset button of the board.

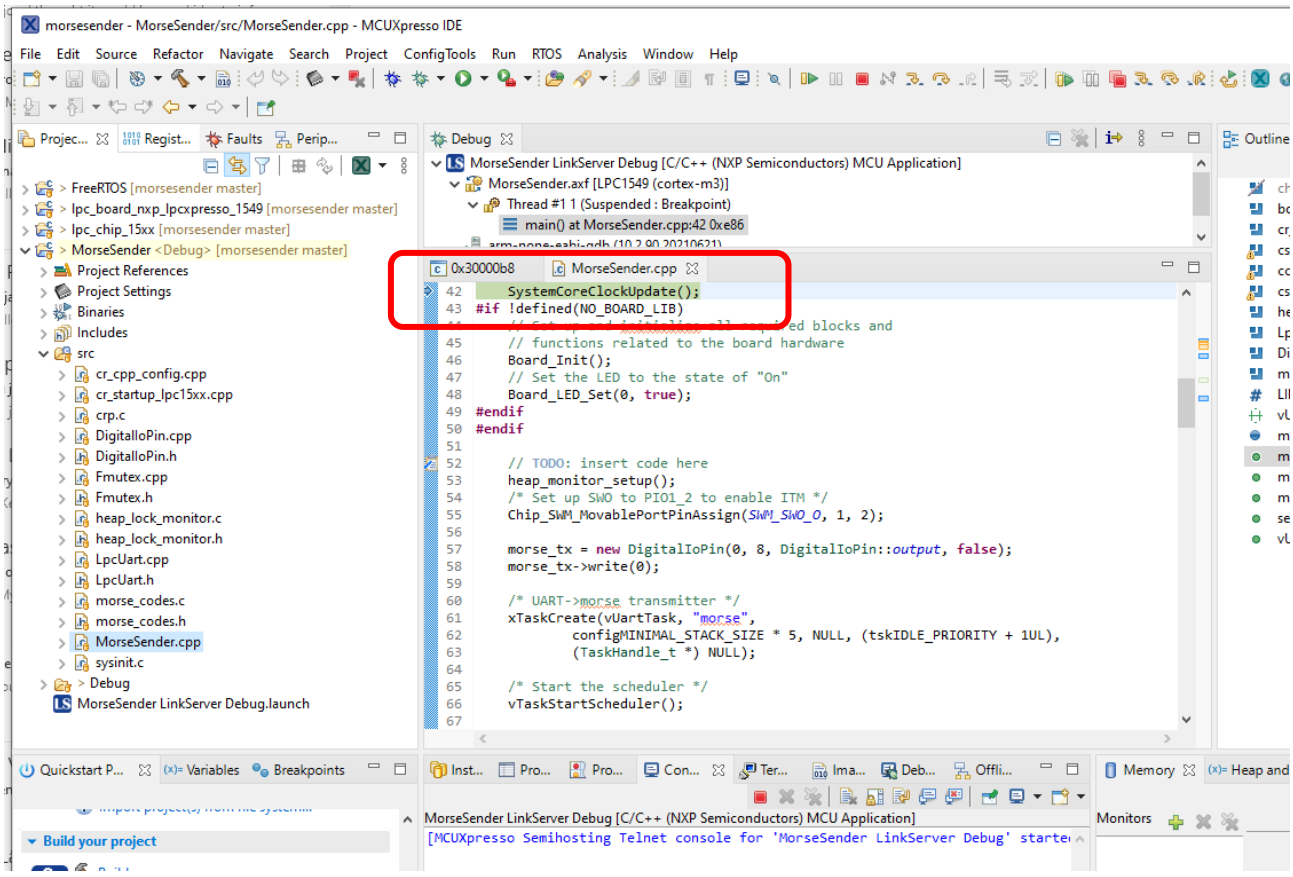


On the first debug round you will get a dialog box where you need to make one change:

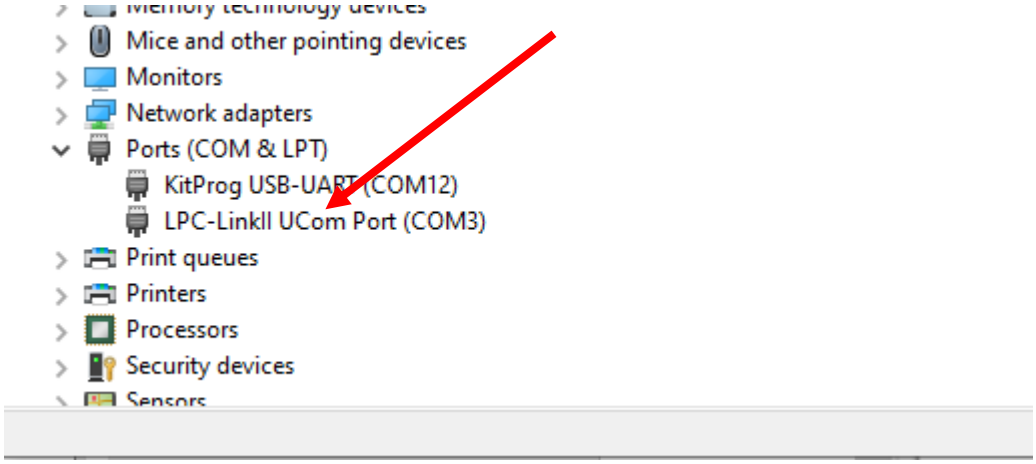
- IDE Debug Mode must be set to **All-Stop**



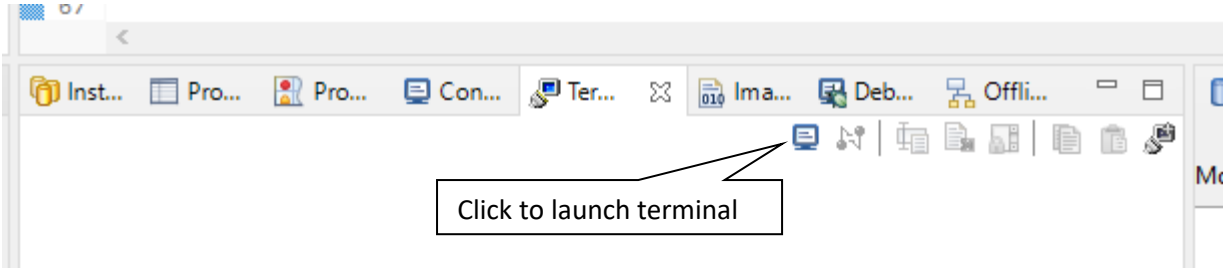
When you press OK the debugging starts. First the compiled program is written to the flash. Flash is non-volatile so the program will remain in the LPC memory even if the board is unplugged. When programming is complete, the first executable line of main() is highlighted in green.



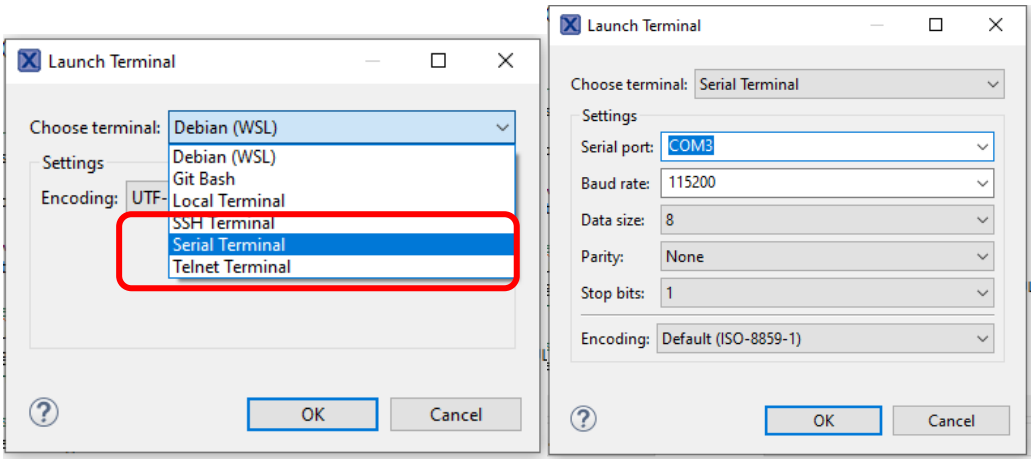
LPC-LinkII, the MCUXpresso HW-debugger that is needed to program the board is a “soft” debugger which means that it must be uploaded to the board when the board is plugged to your computer. This process happens automatically when you start debugging. In addition to debugging, it also creates a serial port (COM-port) that is used to communicate with the board. You can find the port number with Device manager after starting debugging.



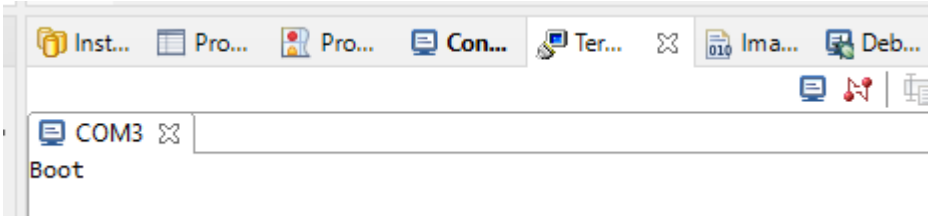
To monitor communication you can start a terminal by selecting the terminal tab and launching a new terminal.



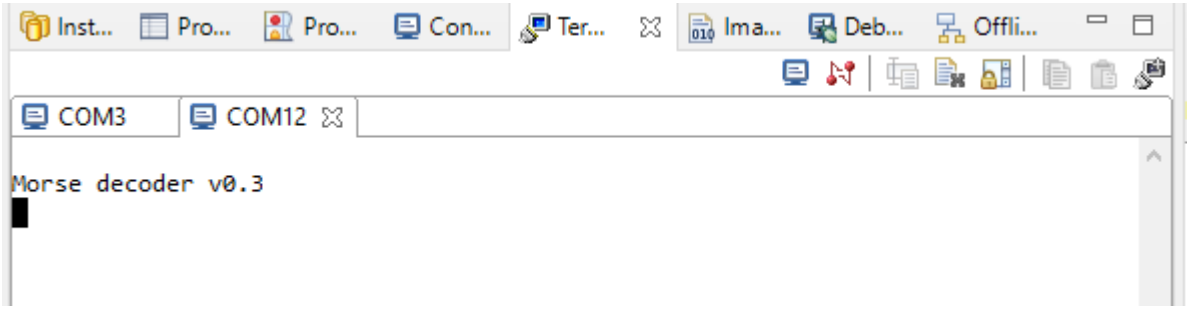
Select Serial terminal in the Launch Terminal dialog and select the COM port of LPCLink-II. Leave the other setting at their defaults.



When the morse sender program start it prints “Boot” to the terminal if the terminal is open.

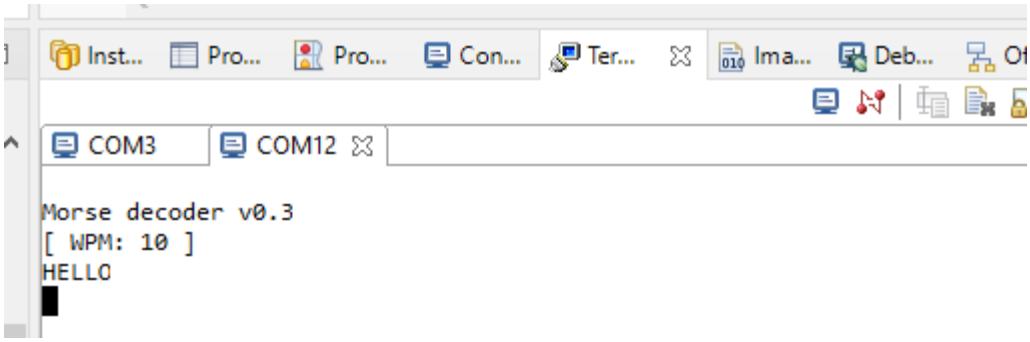


You can also start a terminal for morse decoder that runs on the signal capture board. Just repeat the above but select KitProg COM as the port to use.

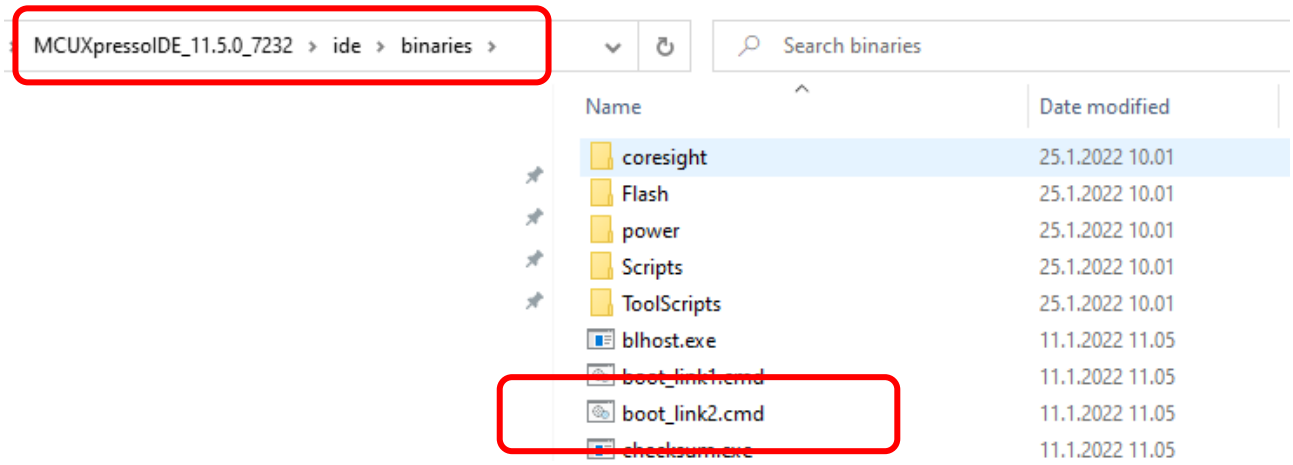


Morse sender does not echo typed text back so you have to be careful when typing commands manually. You can type commands in either upper or lower case. If command is invalid it is silently ignored. Valid

commands are executed. For example, if you type “text hello” and press enter the word “hello” will be sent in morse code. The signal capture board will receive and decode the morse code and output the result to its COM-port. Below you can see output from Morse decoder after command “text hello” was executed on the LPC1549.

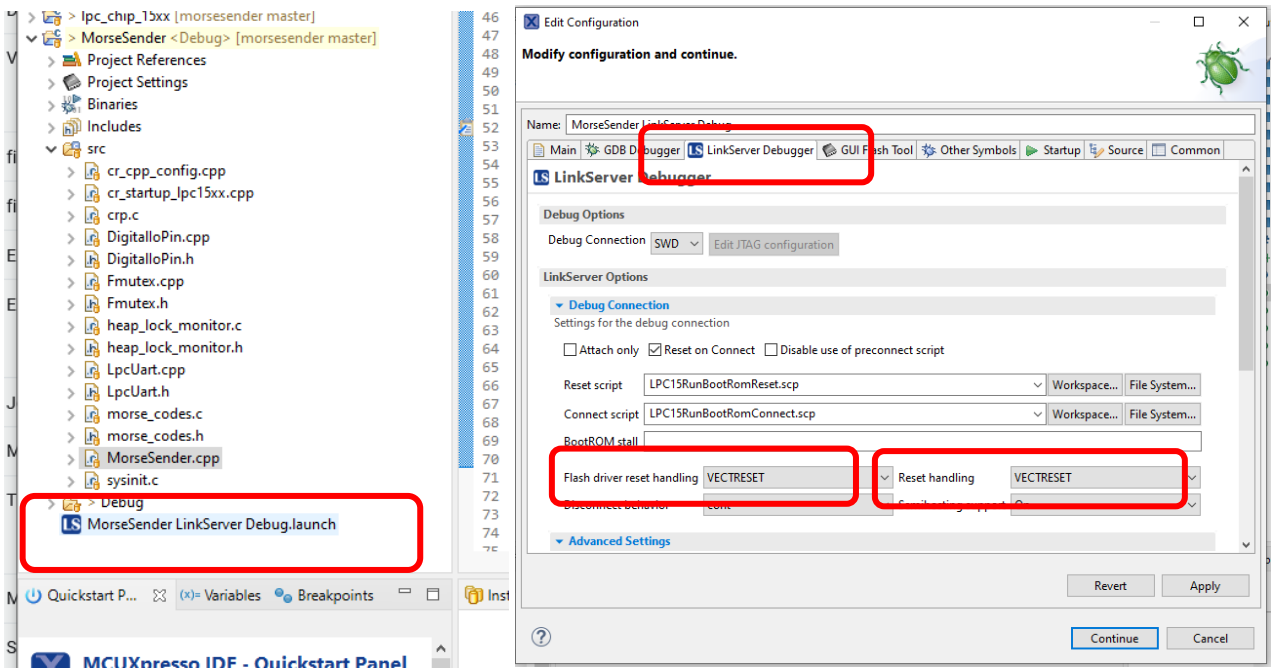


Once the LPC1549 has been programmed it can be used without MCUXpresso provided that LPCLink-II has been activated. LPCLink-II can be activated without starting the IDE by running a script called “boot_link2”. The script is installed in <your install dir>/MCUXpressoIDE_11.5.0_7232/ide/binaries



By running boot_link2 you can bring back the LPCLink-II com port without starting the IDE.

Sometimes the debugger may not be able to download the code to LPC1549 board. If that happens double-click on the launch configuration and change reset handling to VECTRESET. The click Apply and Continue and try again.



If debugging still fails check you firewall settings. The IDE communicates with the debugger server over network. The IDE and debugger servers must have permission to use network to work properly.