Bringing up STM32F411CEU6

## 

## Project Creation

Follow this official STM32 video to create the initial project. Of course, you’ll need to make sure you specify the device according to what you are actually using.

<https://www.youtube.com/watch?v=eumKLXNlM0U>

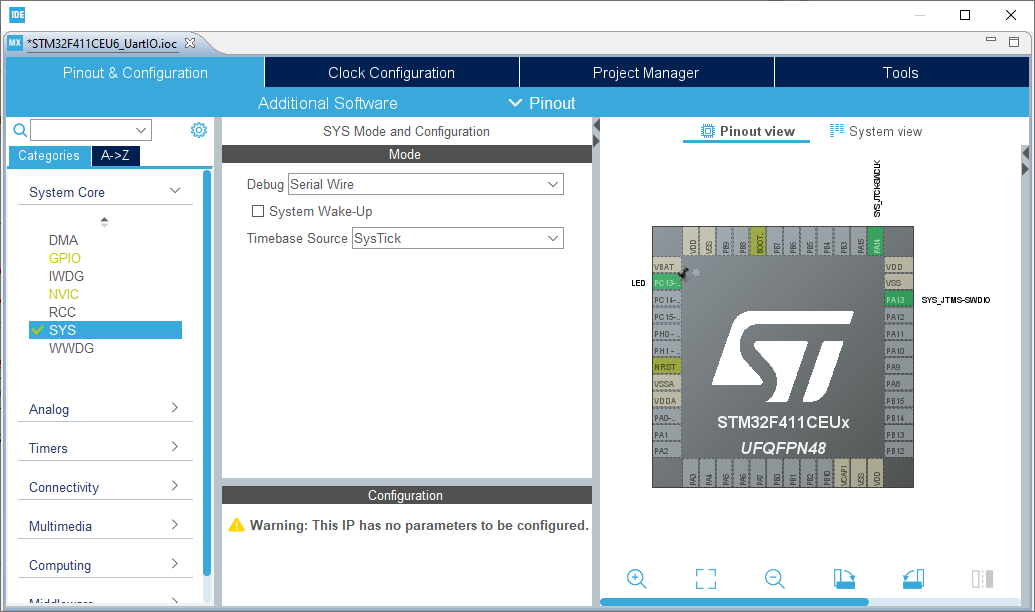
## Project Settings

Open CubeMX by double clicking the \*.ioc file in your project.

### Serial Debugging

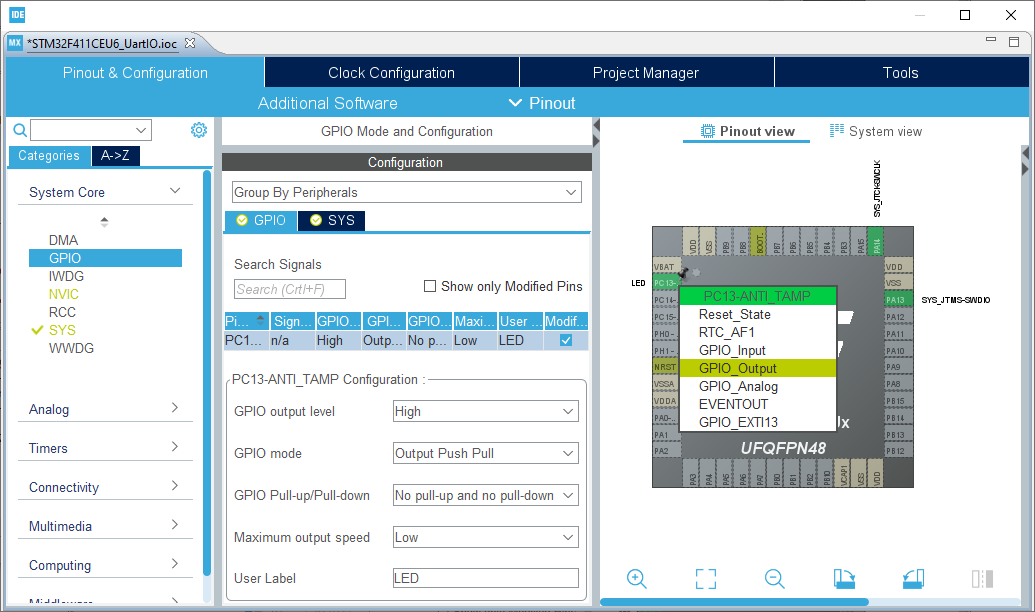
Enable the serial debugging feature (for use with ST-Link) so that later you’ll be able to step through your code as needed.

System Core >> SYS >> Debug >> Serial Wire

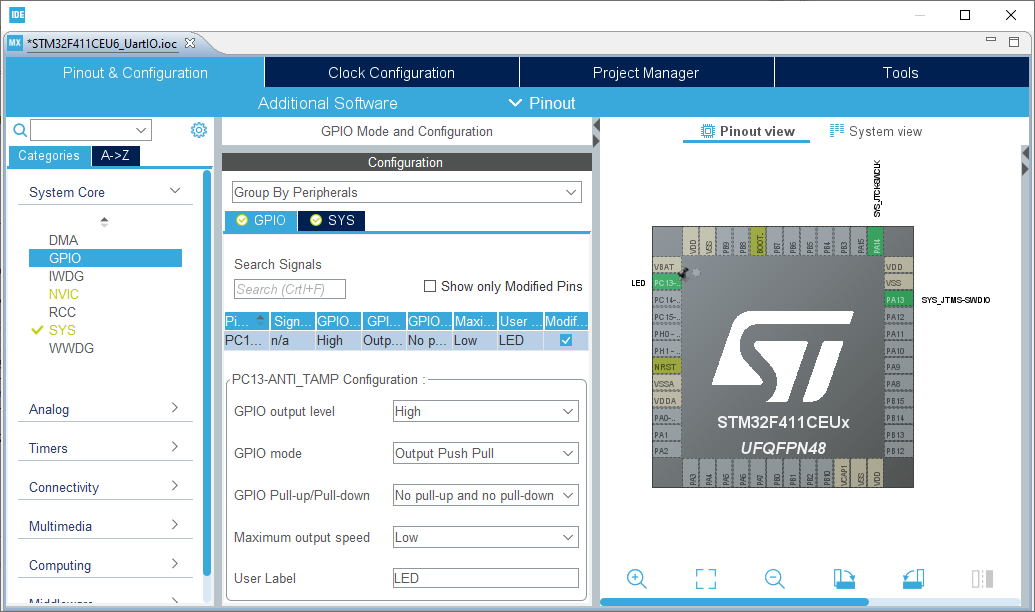


## Define the builtin LED

System Core >> GPIO



Left click pin PC13 in the diagram and select GPIO\_Output from the drop down list. Next, set up the drive characteristics of the pin.



System Core >> GPIO

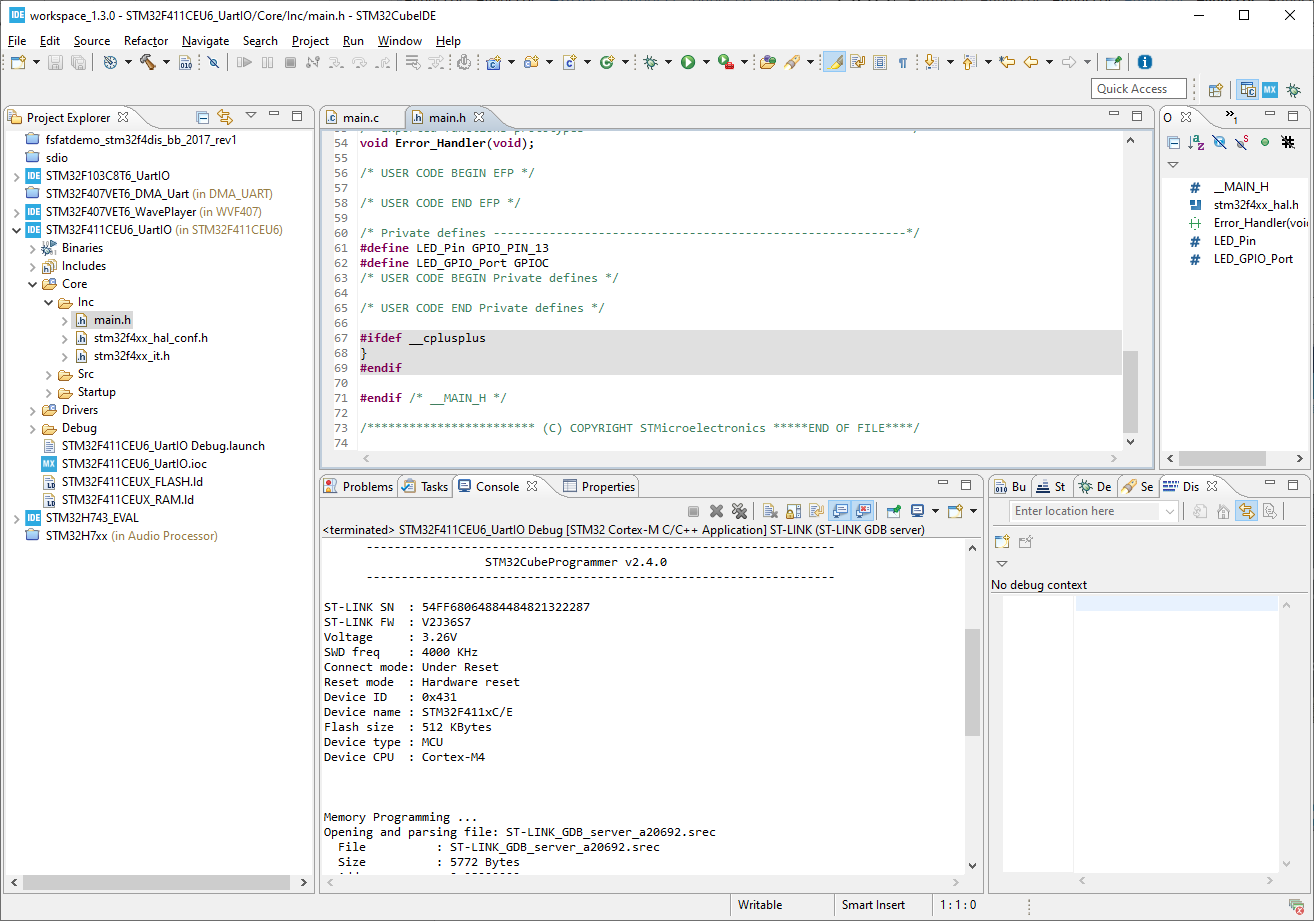
Select PC13-ANTI\_TAMP from the grid display:

* GPIO Output Level – High
* GPIO mode – Output Push Pull
* GPIO Pull-up / Pull-down – No pull-up, no pull-down
* Maximum output speed – Low (no other choice)
* User Label – LED

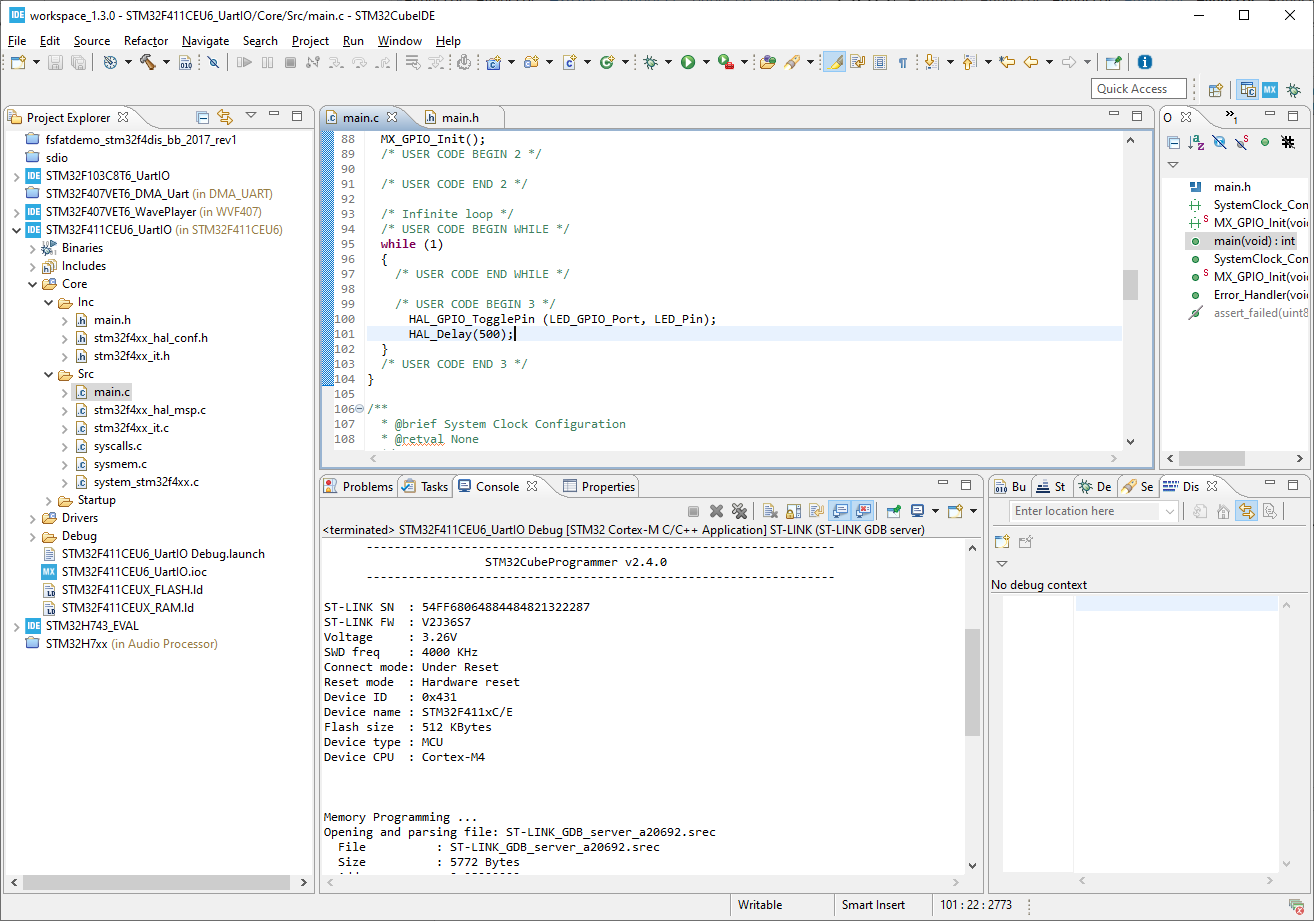
Save and close your project .ioc file

# Blinky

Open your main.h source file and observe the declarations for the LED port and pin.

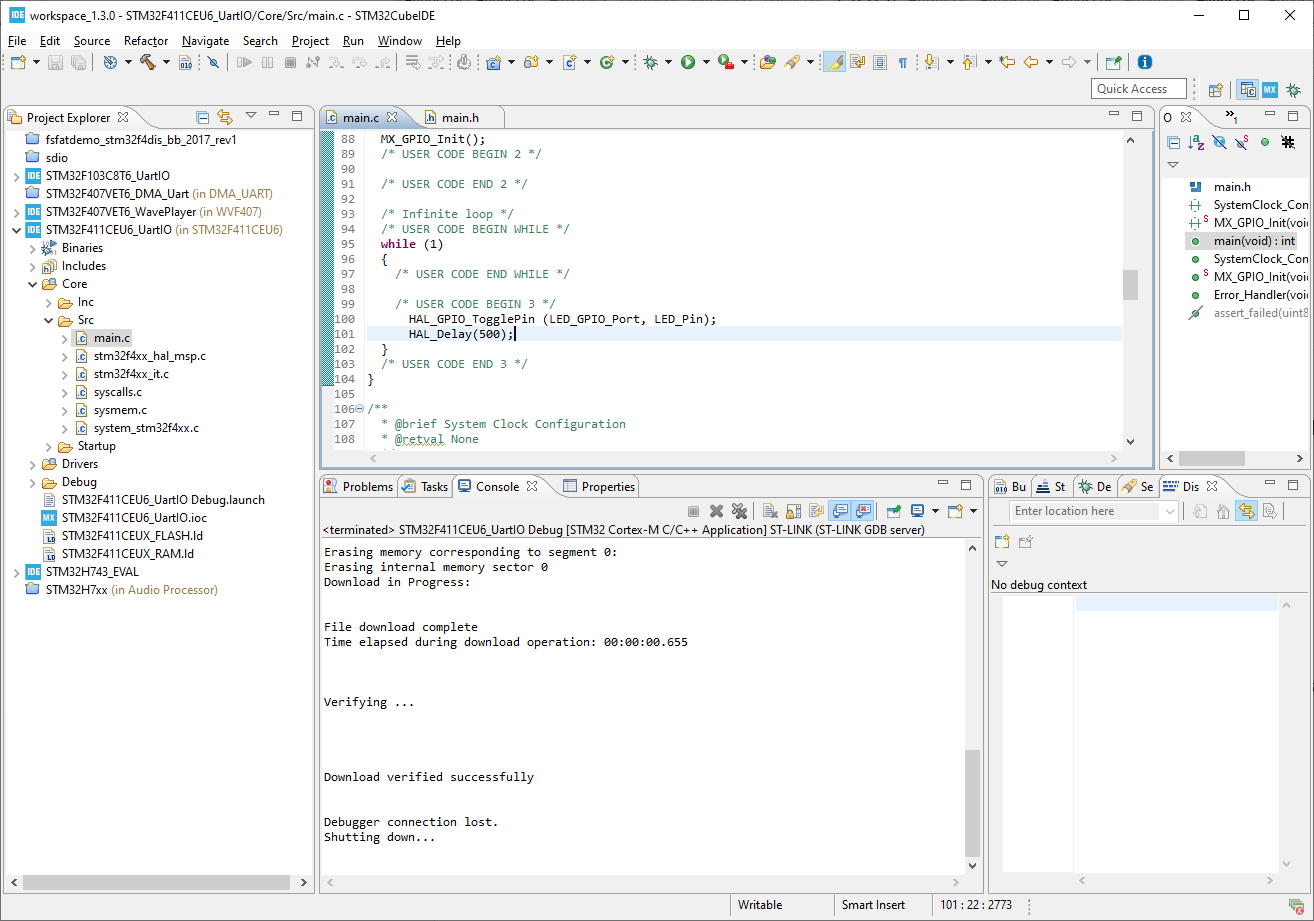


Open your main.c file and locate the while(1) loop. Using the LED Port and Pin declarations that you just confirmed in main.h, create some simple LED flash code:

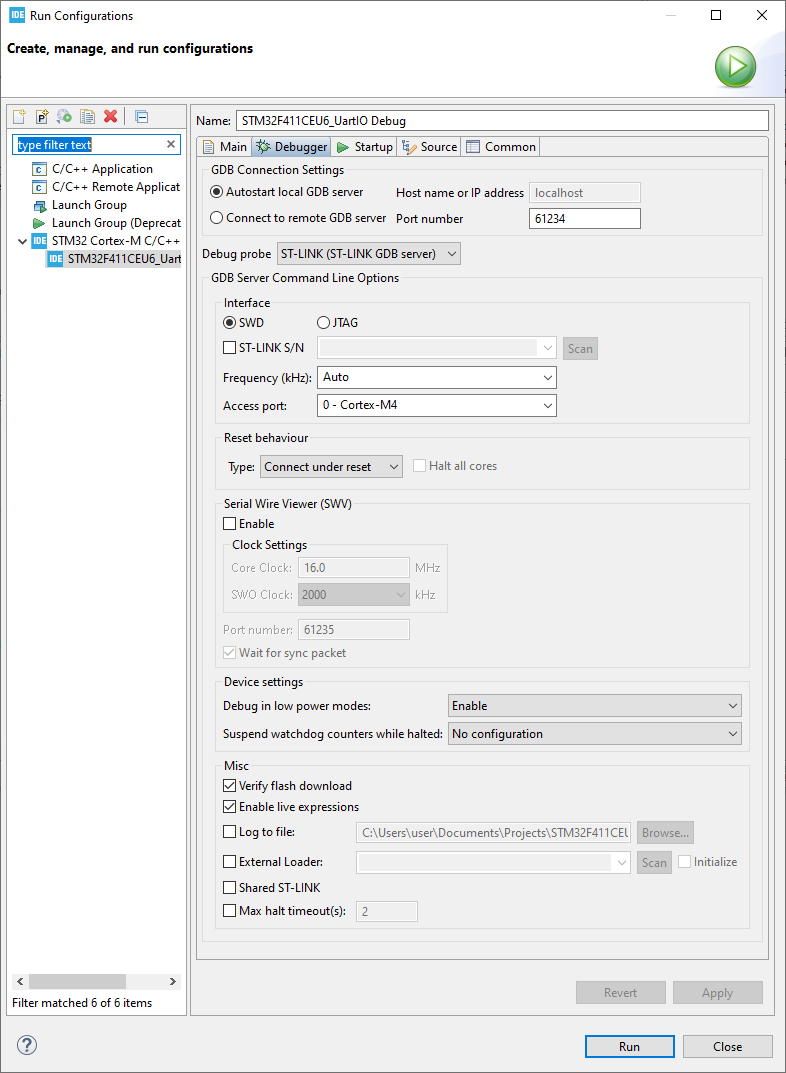


Ensure your code is placed after the /\* USER CODE BEGIN \*/ comment. As a general rule, all updates to the source code need to sit within the USER CODE areas otherwise you risk having your edits overwritten the next time CubeIDE needs to update some of the source code.

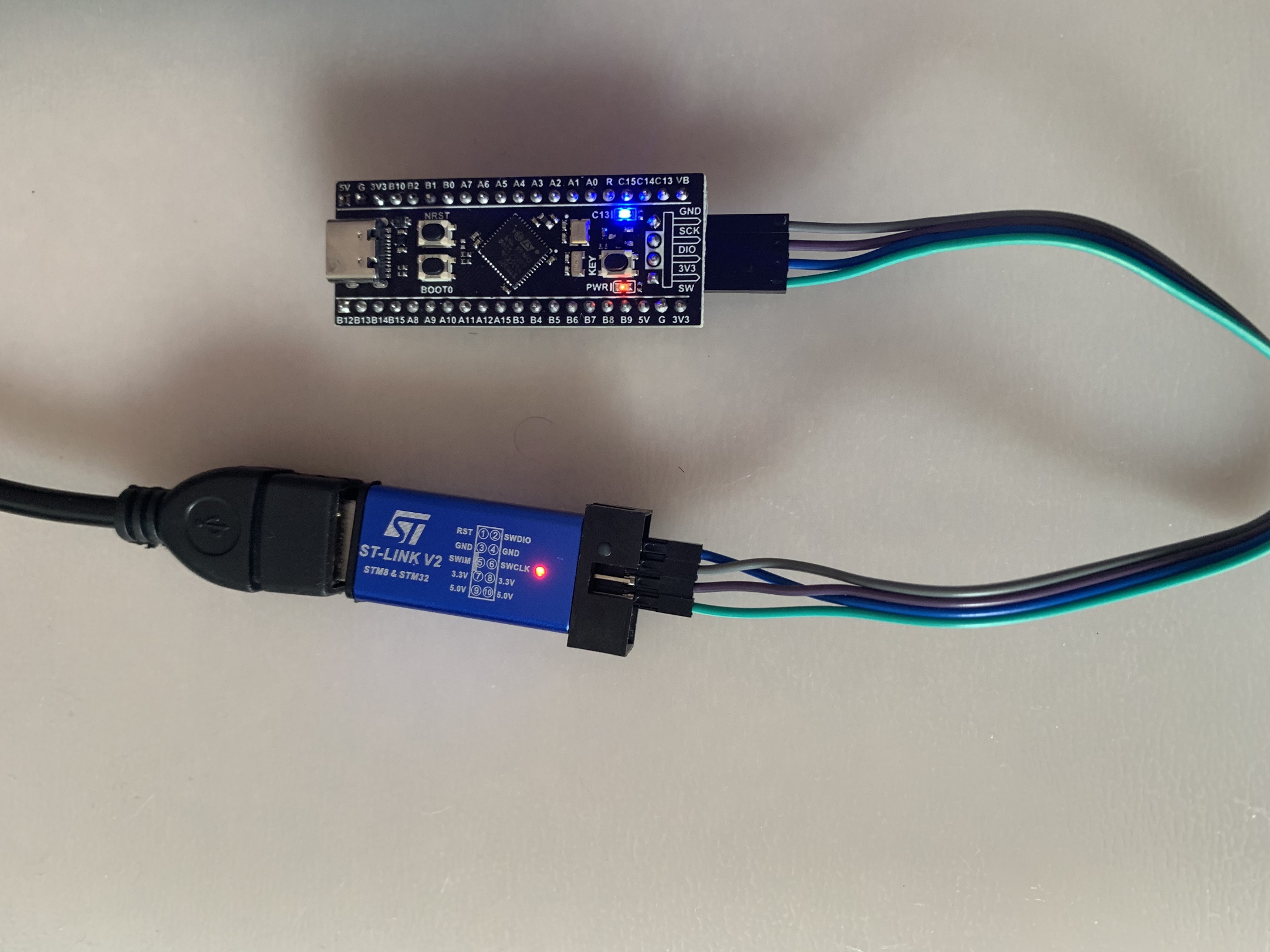
Click on the Run icon:



As this is the first time you’ve run your code, you’ll be asked to specify the settings for downloading and debugging the code. If you’re using ST-Link, I’ve found the defaults seem to work without modification, but if at any point you need to modify these settings, click the drop down arrow beside the run icon and select the Run Configurations menu item:

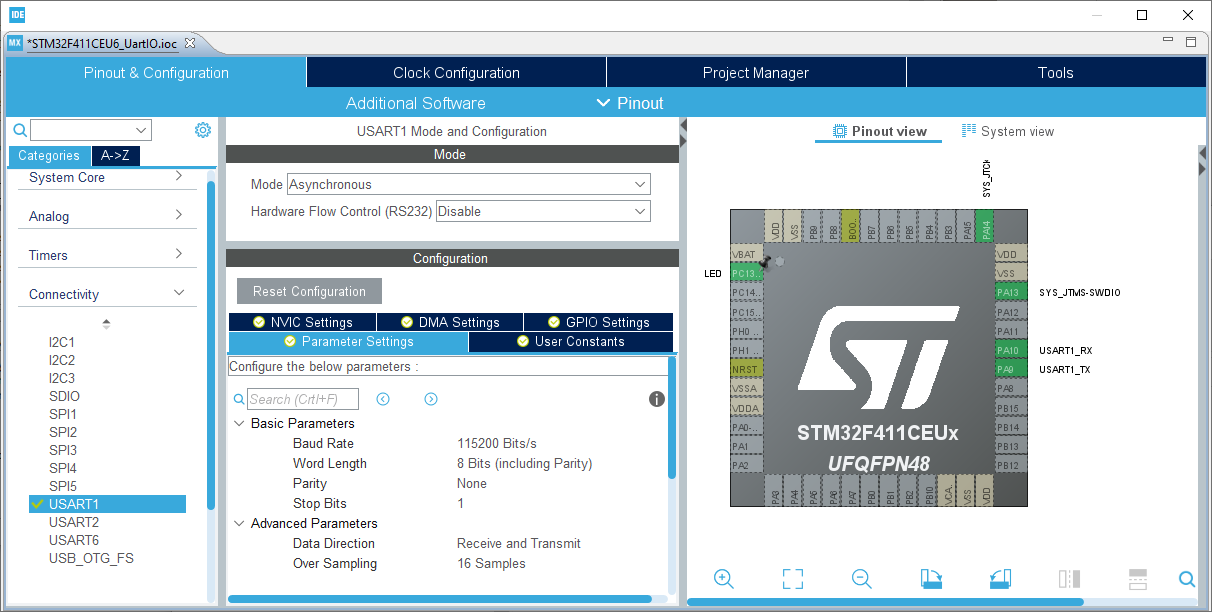


Assuming everything has gone to plan, you should see the LED flashing something like this:



# Adding a USART Interface

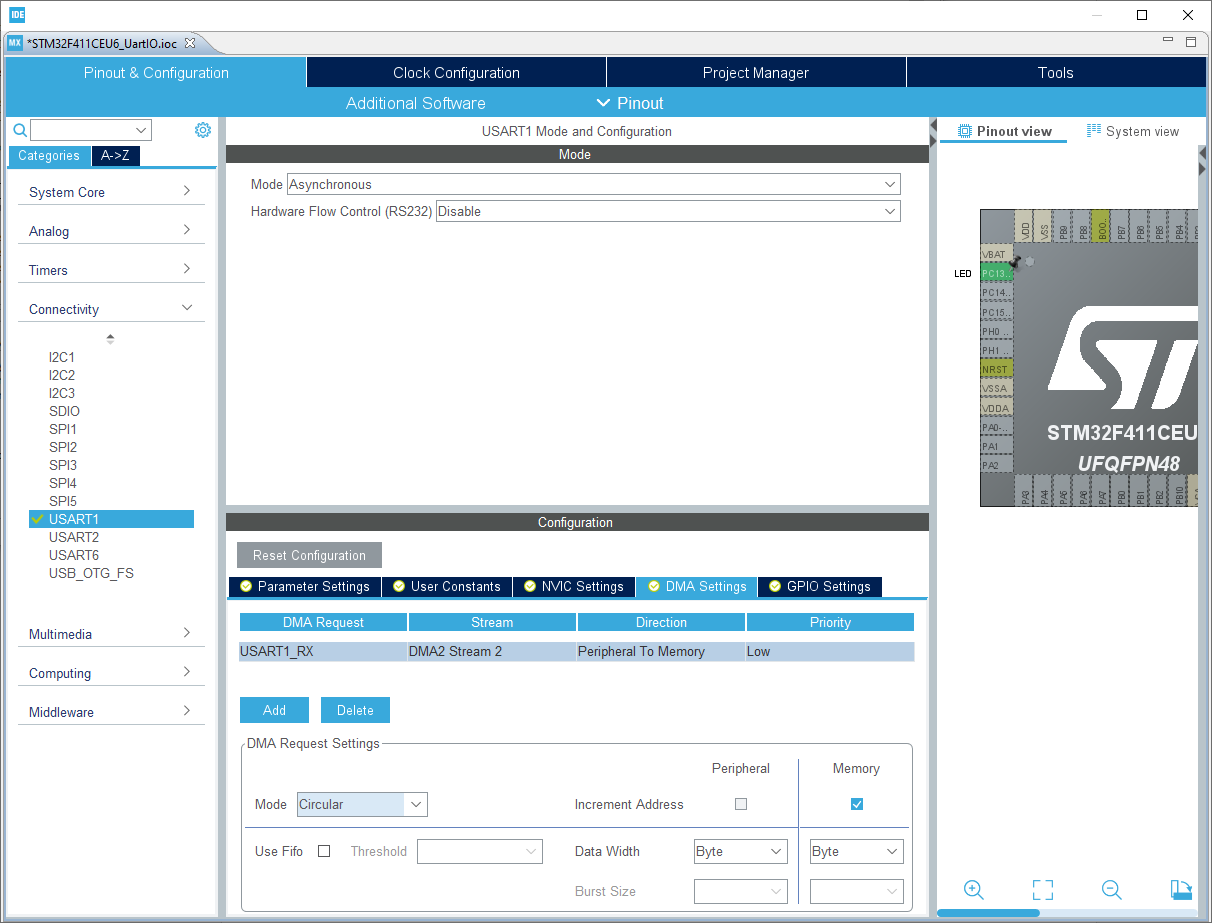
Open the project .ioc file and select Connectivity >> USART1. Using the Mode drop down list, enable the USART as follows:



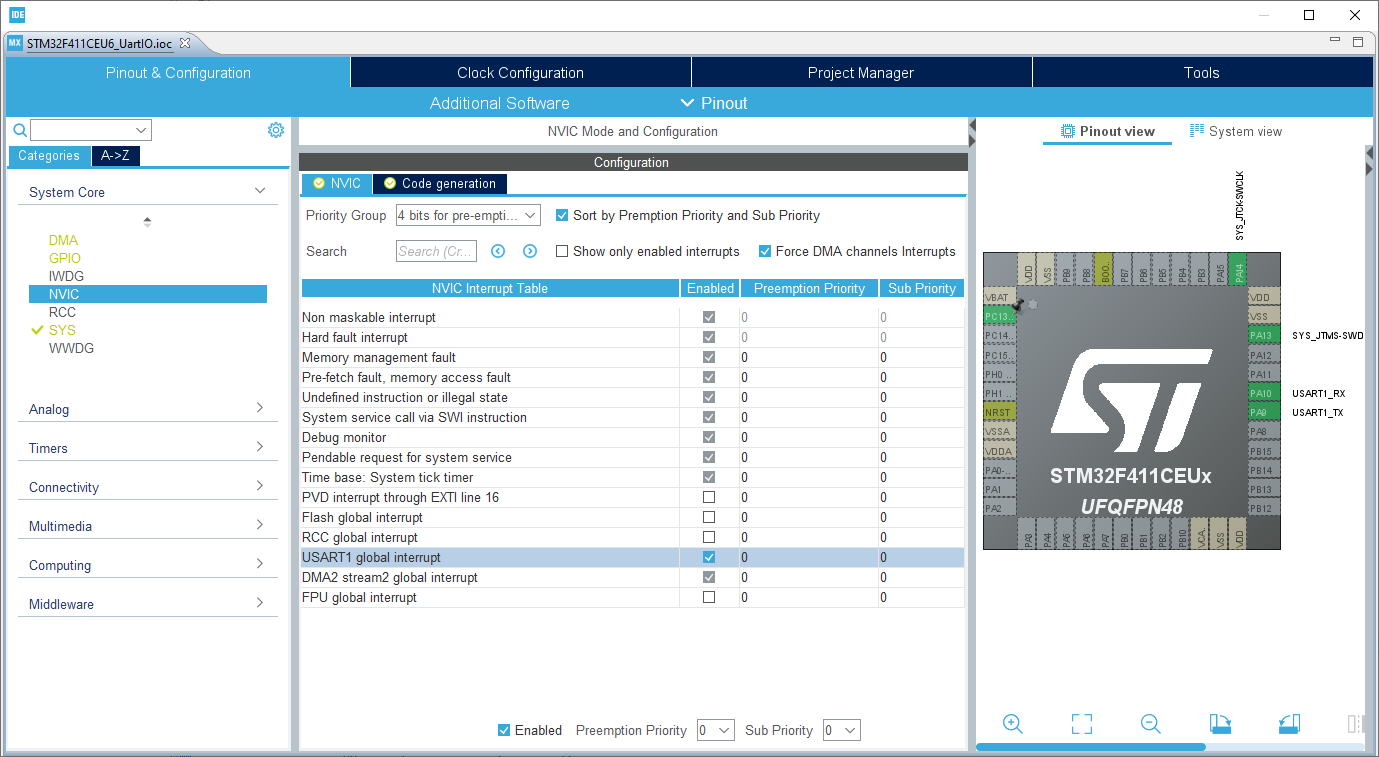
As we will be using one of the DMA channels for automating serial reception, select the DMA Settings Tab.

Click the Add button and select USART1\_RX from the drop down list.

Under the DMA Request Settings, set the Mode to Circular.



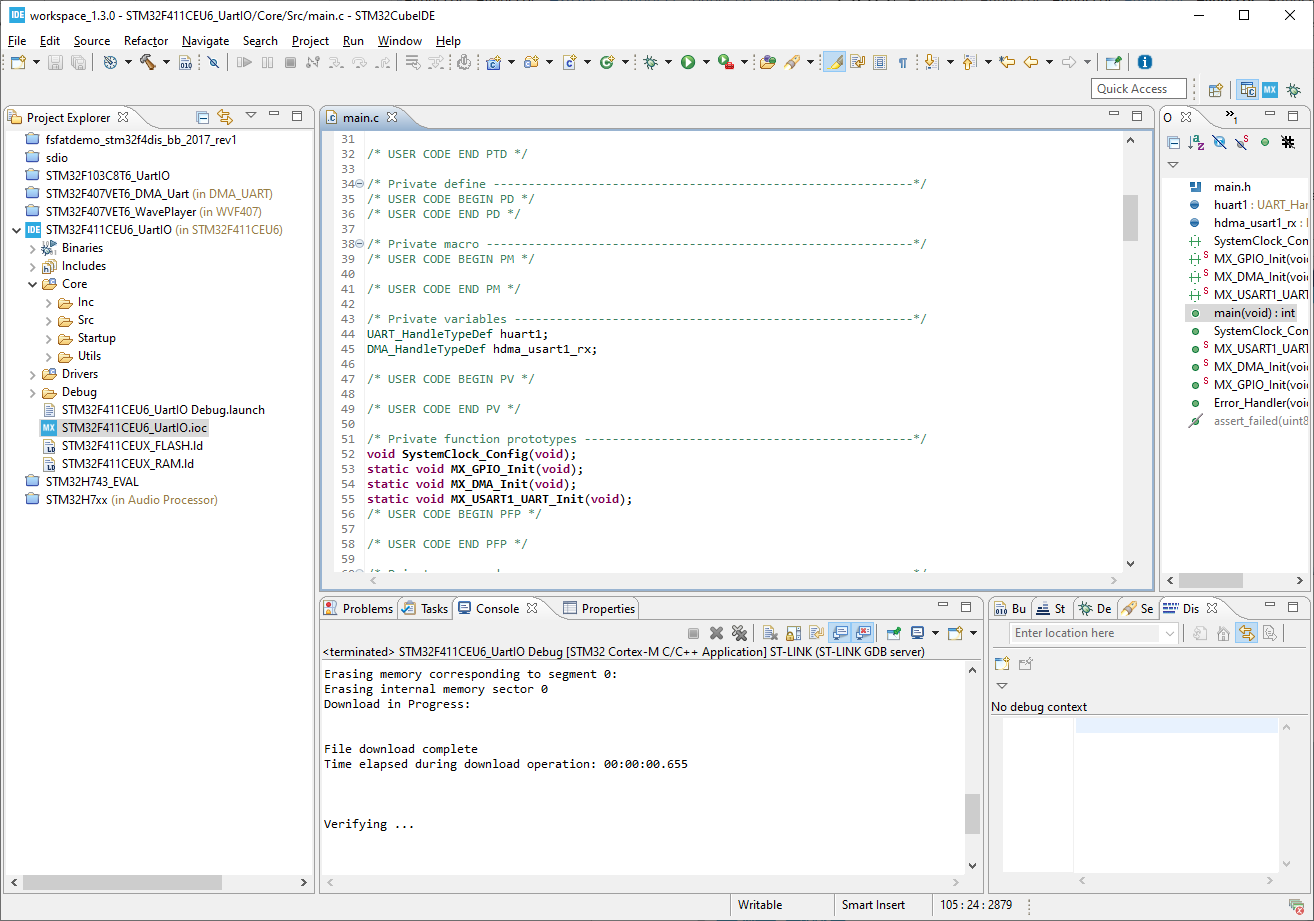
Next, because we plan to use the USART Interrupt to help with sending out characters in the background, a quirk of CubeIDE is that we need to enable it globally via the CubeMX interface.



Open System Core >> NVIC and ensure the USART1 global interrupt is enabled.

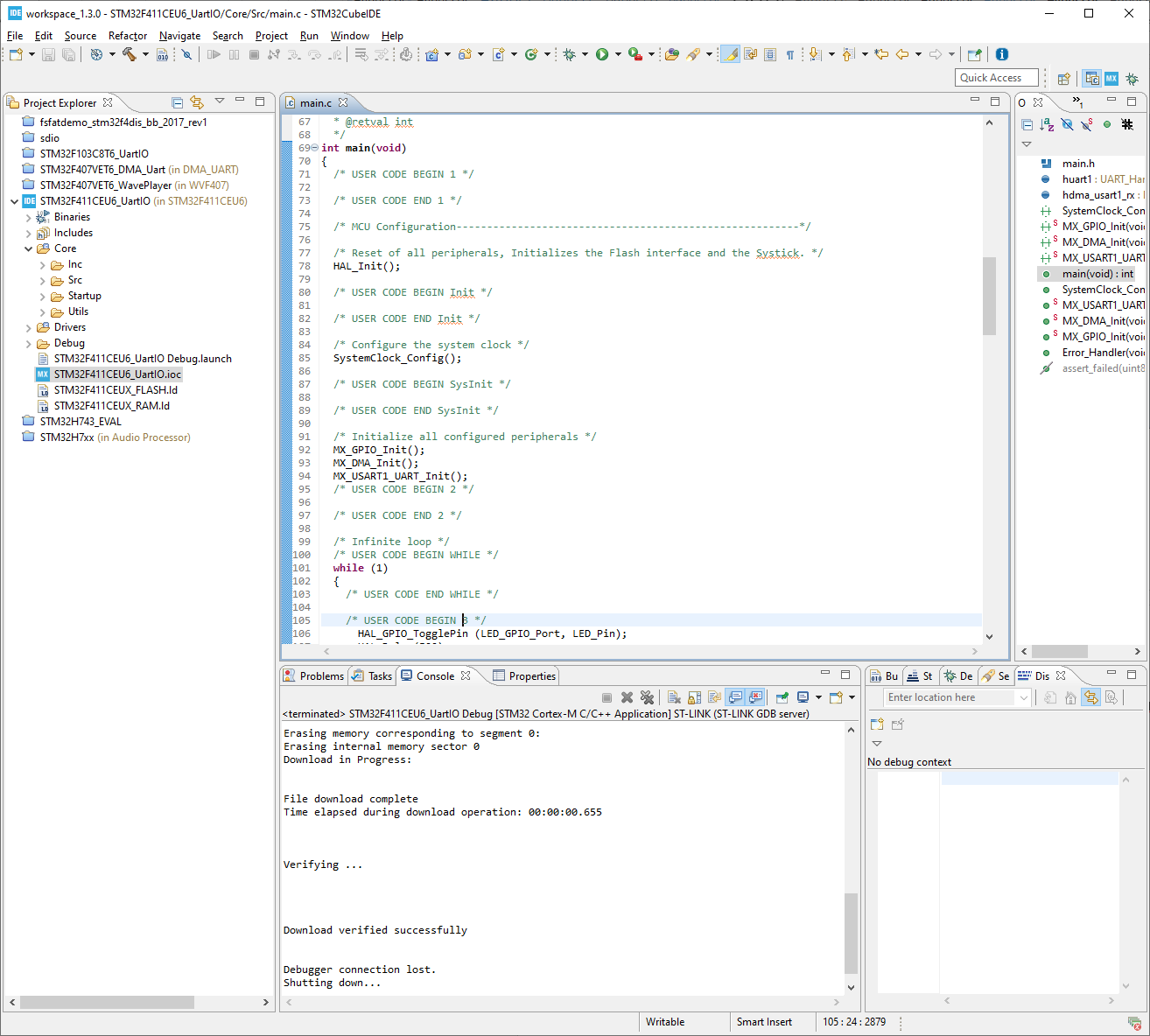
Save your settings and close the project .ioc file.

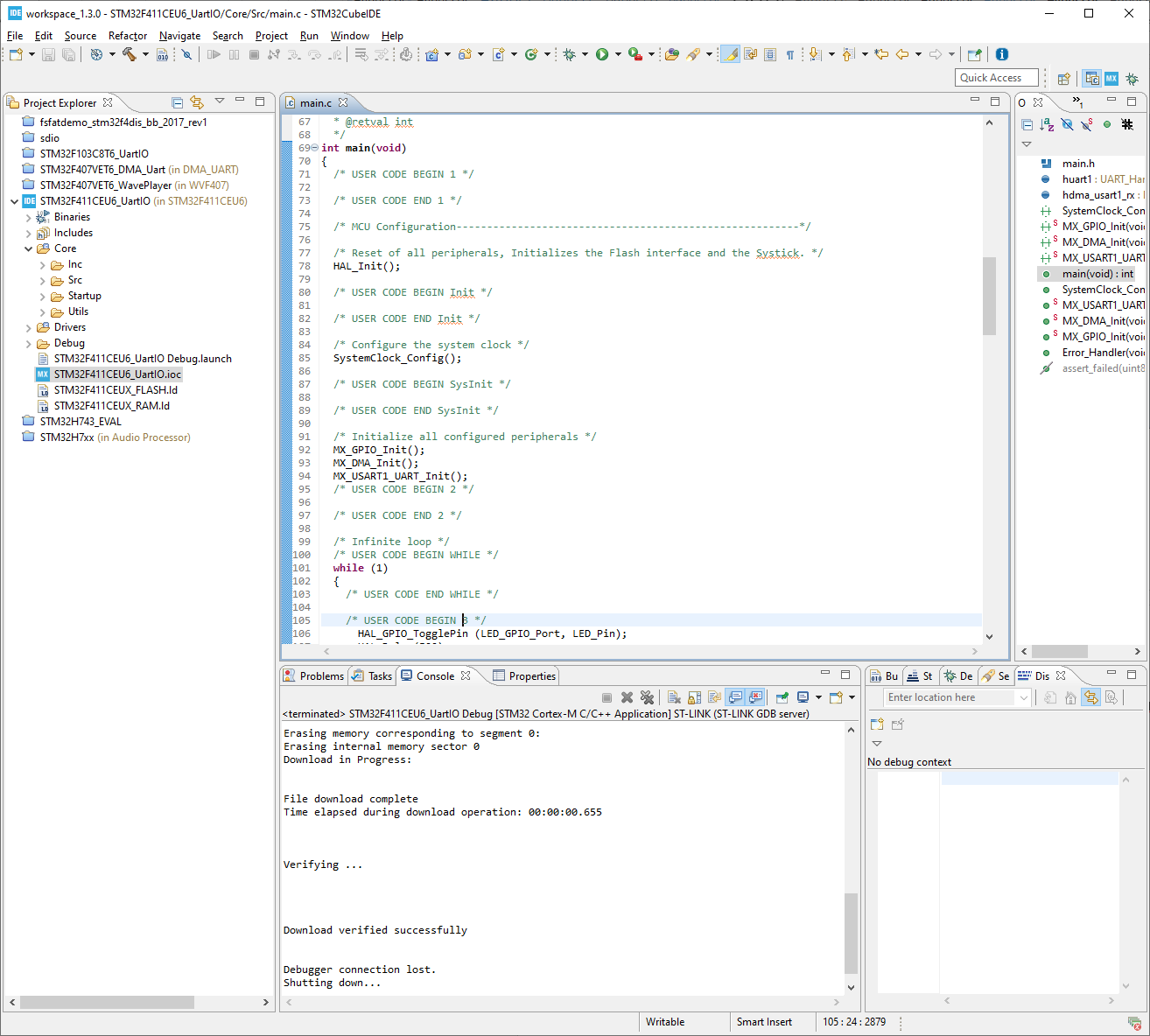
In your main.c file, take note that the following lines have been added automatically by CubeIDE:



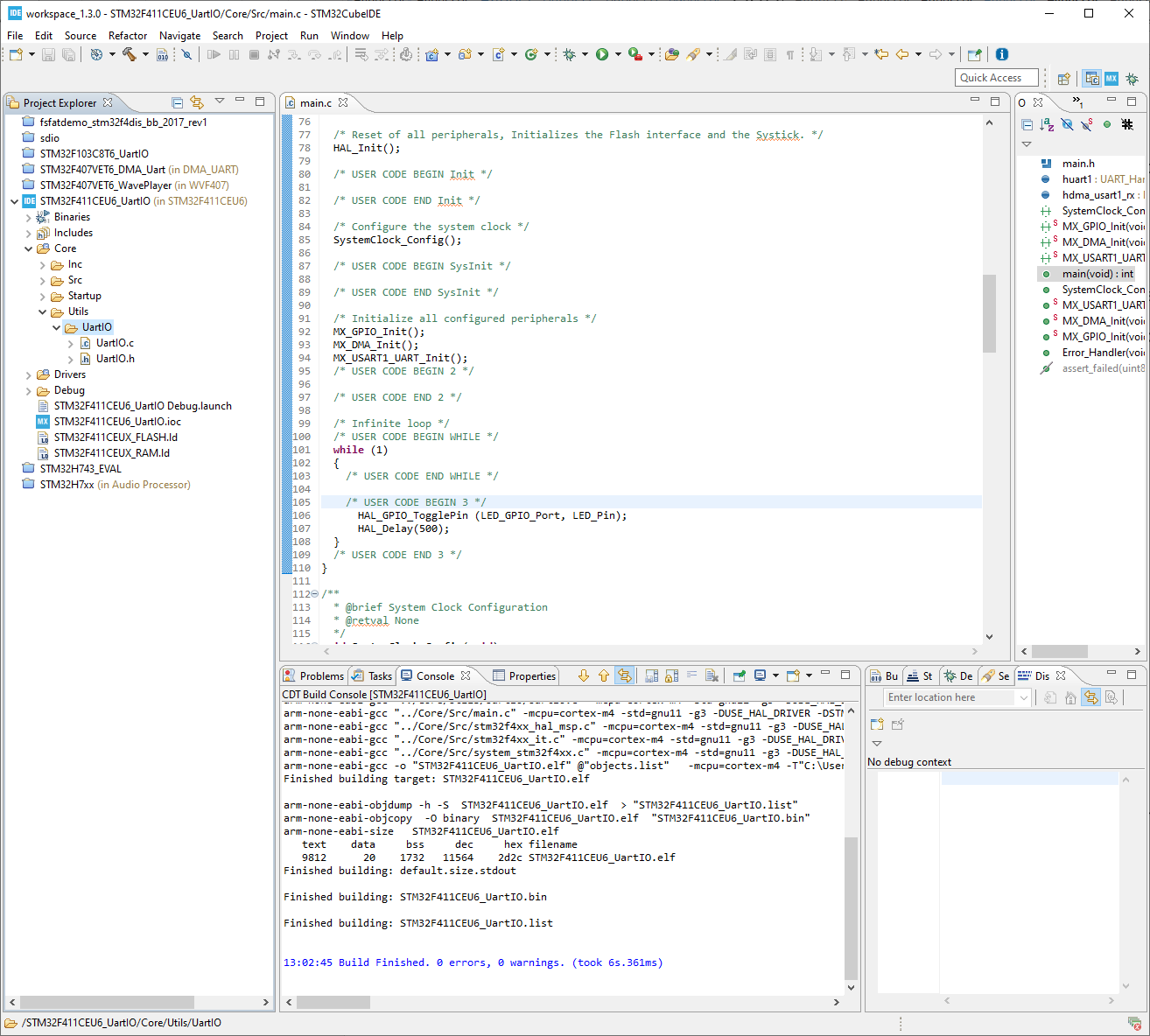
These are the ‘handles’ that are used for accessing the UART and DMA resources.

You will also notice that some extra initialization functions for each of these resources has been added to the mainline code:

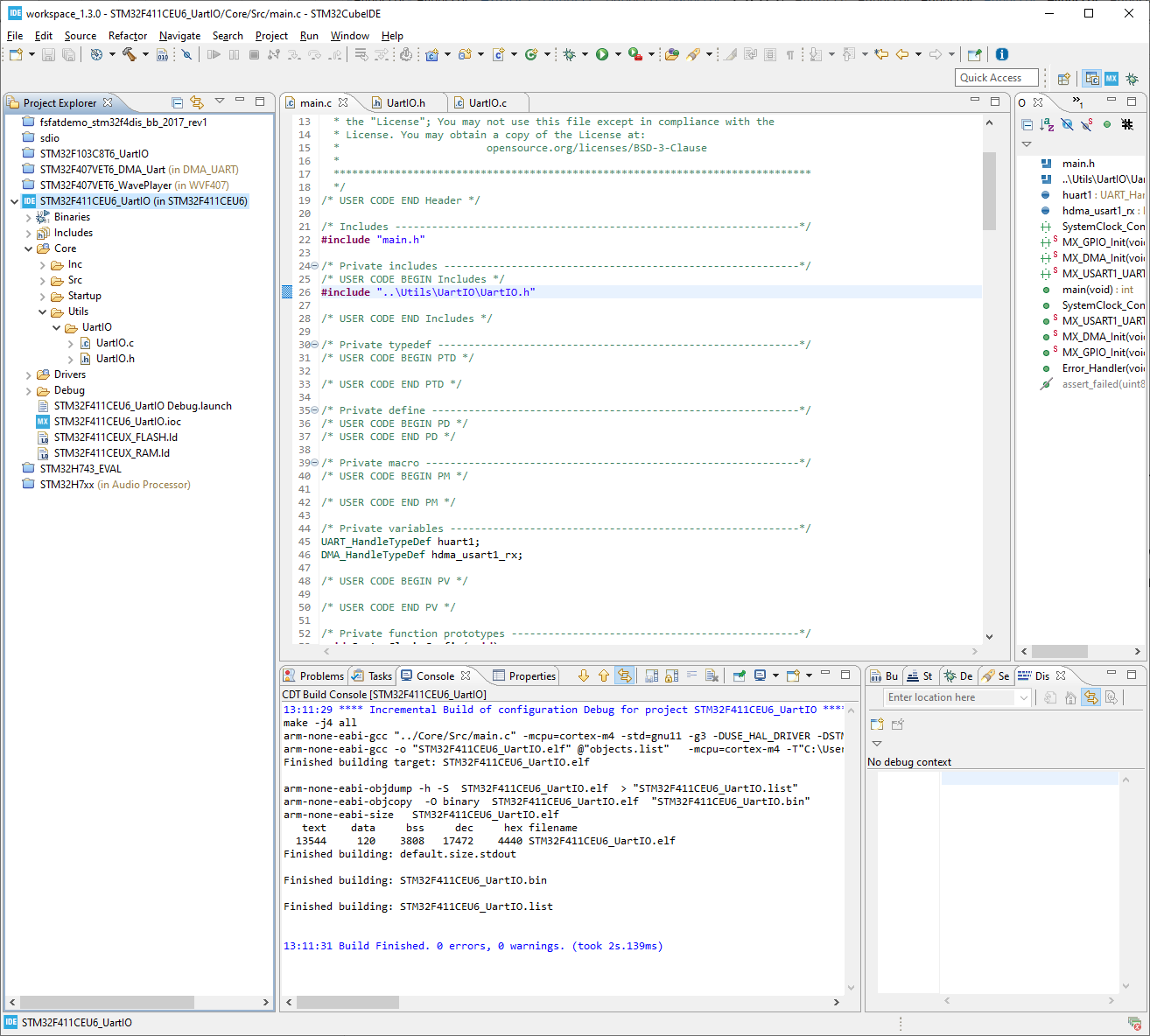


As an interim step, confirm that your code still builds without error by clicking on the ‘build’ icon

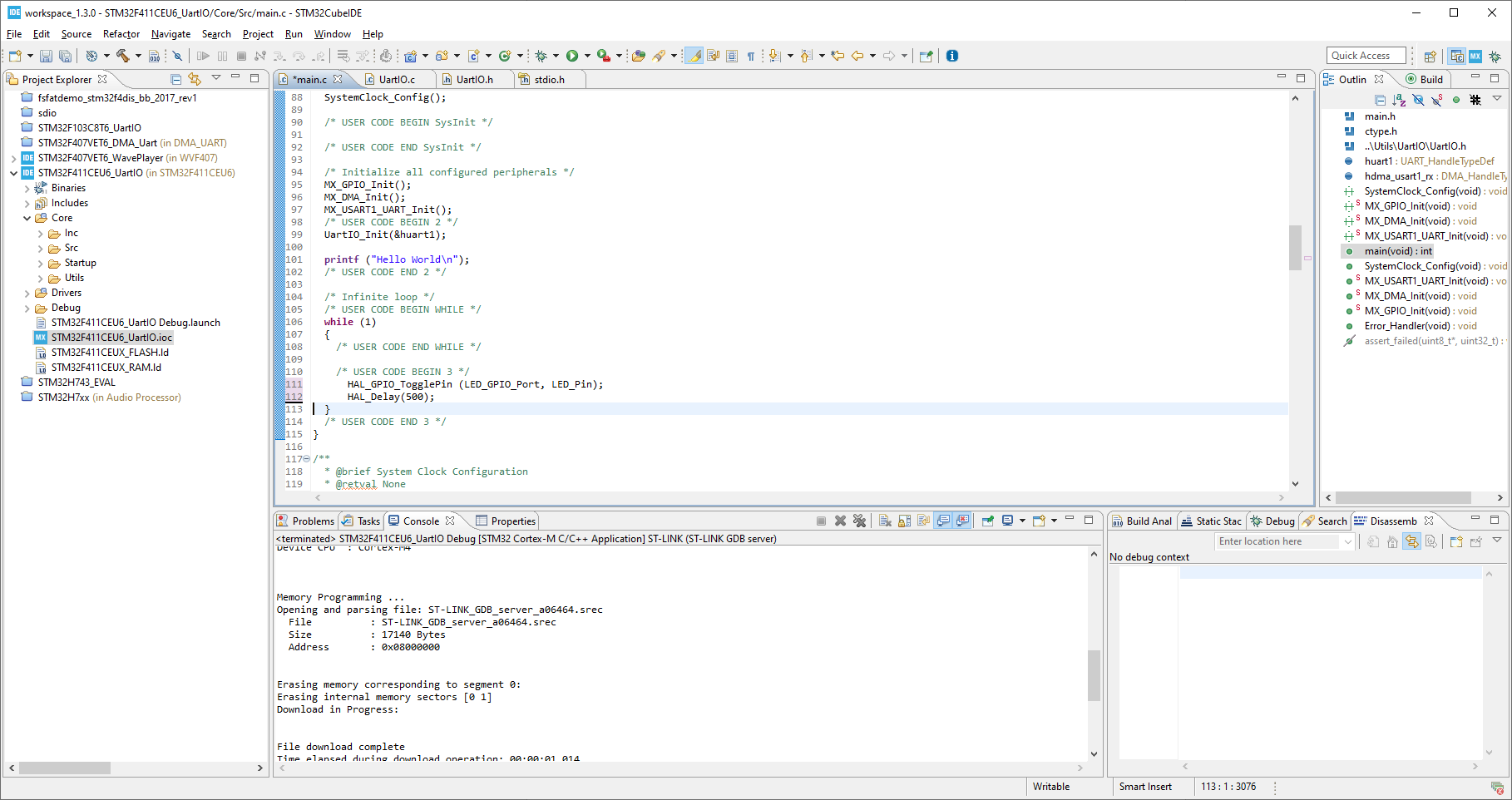
Using Windows Explorer (or right-click the Core folder in CubeIDE and select New >> Folder to) create a Utils >> UartIO folder. Copy the UartIO.c and UartIO.h files into this location.



Go to the top of your main.c file and, in the USER CODE section, include the UartIO.h file into the project.

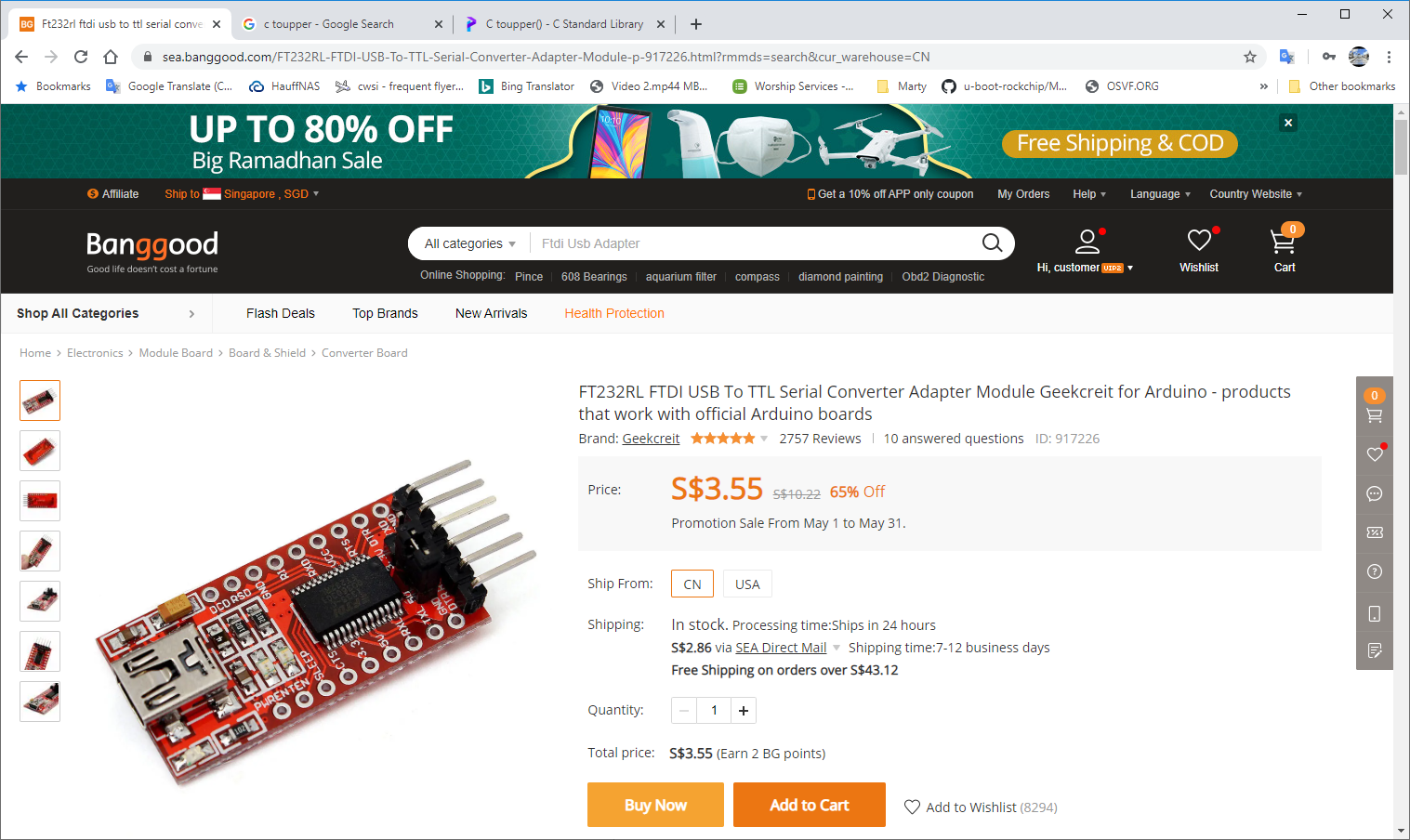


Next, initialize the custom UartIO code by calling its UartIO\_init() function and pass the address of the UART handle to it (huart1):



To test the USART output, you can also include the classic “Hello World” message.

To receive the UART data, you’ll need to hook up a serial port device. One of the simply FTDI USB/TTL devices (or similar) would be an excellent choice.

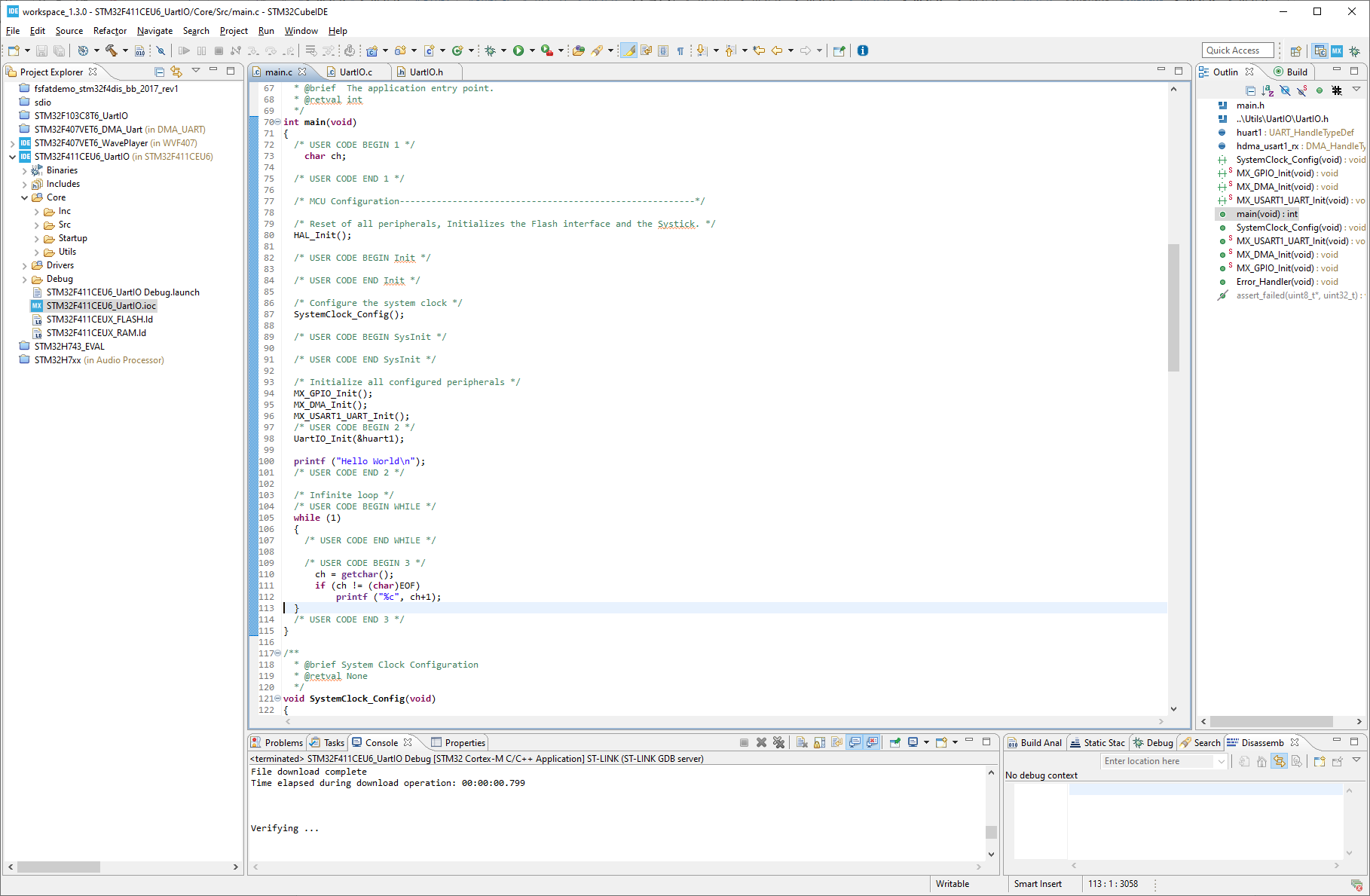


Connect the boards as follows:

|  |  |
| --- | --- |
| STM32 Board | USB Serial Board |
| A9 (USART1\_TX) | RX |
| A10 (USART1\_RX) | TX |
| GND | GND |

If there is a voltage selector jumper on the USB Serial device then you should probably set that to 3.3V, although we are currently not powering the board via this device so it is not critical.

Update your main.c file as follows and run it:



Use PuTTY (or a similar) serial terminal program to transmit and receive the data from a PC.



## Source Files

Source files for UartIO can be found at: <https://github.com/DrMarty/STM32Utils/tree/master/UartIO>

An example project illustrating the content of this document can be found at:

<https://github.com/DrMarty/STM32Examples/tree/master/STM32F411CEU6_UartIO>