

[Users \(/activity/\)](#) » [hudakz \(/users/hudakz/\)](#) » [Code \(/users/hudakz/code/\)](#) » [MapleMini\\_Hello \(/users/hudakz/code/MapleMini\\_Hello/\)](#)



## [Zoltan Hudak \(/users/hudakz/\)](#) / [MapleMini\\_Hello \(/users/hudakz/code/MapleMini\\_Hello/\)](#) ✓ Featured

Using low cost LeafLab Maple Mini boards with mbed.

 [Dependencies: \(/users/hudakz/code/MapleMini\\_Hello/dependencies\)](#)  [mbed-MapleMini \(/users/hudakz/code/mbed-MapleMini/\)](#)  [mbed \(/users/mbed\\_official/code/mbed/\)](#)

[Home \(/users/hudakz/code/MapleMini\\_Hello/\)](#) [History \(/users/hudakz/code/MapleMini\\_Hello/shortlog\)](#)  
[Graph \(/users/hudakz/code/MapleMini\\_Hello/graph\)](#)  
[API Documentation \(/users/hudakz/code/MapleMini\\_Hello/docs/\)](#)  
[Wiki \(/users/hudakz/code/MapleMini\\_Hello/wiki/\)](#)  
[Pull Requests \(/users/hudakz/code/MapleMini\\_Hello/pull-requests/\)](#)

# Using Maple Mini board with mbed

Sparkfun used to sell a [board \(https://www.sparkfun.com/products/retired/11280\)](https://www.sparkfun.com/products/retired/11280) (now available on eBay and from other online sellers for less than \$4) which provides an affordable and flexible way for users to try out new ideas and build prototypes. The board is equipped with an [STM32F103CBT6 \(http://www.st.com/content/st\\_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f1-series/stm32f103/stm32f103cb.html\)](http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f1-series/stm32f103/stm32f103cb.html) microcontroller compatible with the mbed [NUCLEO-F103RB \(https://developer.mbed.org/platforms/ST-Nucleo-F103RB/\)](https://developer.mbed.org/platforms/ST-Nucleo-F103RB/) platform.

## Microcontroller features

- STM32F103CBT6 in LQFP48 package
- ARM®32-bit Cortex®-M3 CPU
- 72 MHz max CPU frequency
- VDD from 2.0 V to 3.6 V
- 128 KB Flash
- 20 KB SRAM
- GPIO (34) with external interrupt capability
- 12-bit ADC (2) with 10 channels
- RTC
- Timers (4)
- I2C (2)
- USART (3)
- SPI (2)
- USB 2.0 full-speed
- CAN


## Board features

- Small foot-print
- Flexible board power supply: USB VBUS or external source (3.3V, 5V)
- User LED: LED1
- Two push buttons: RESET, USER BUTTON
- Mini-B USB connector

## Maple Mini board pinout

## Repository toolbox

[Import into Compiler \(https://os.mbed.c](#)

 [Export to desktop IDE](#)

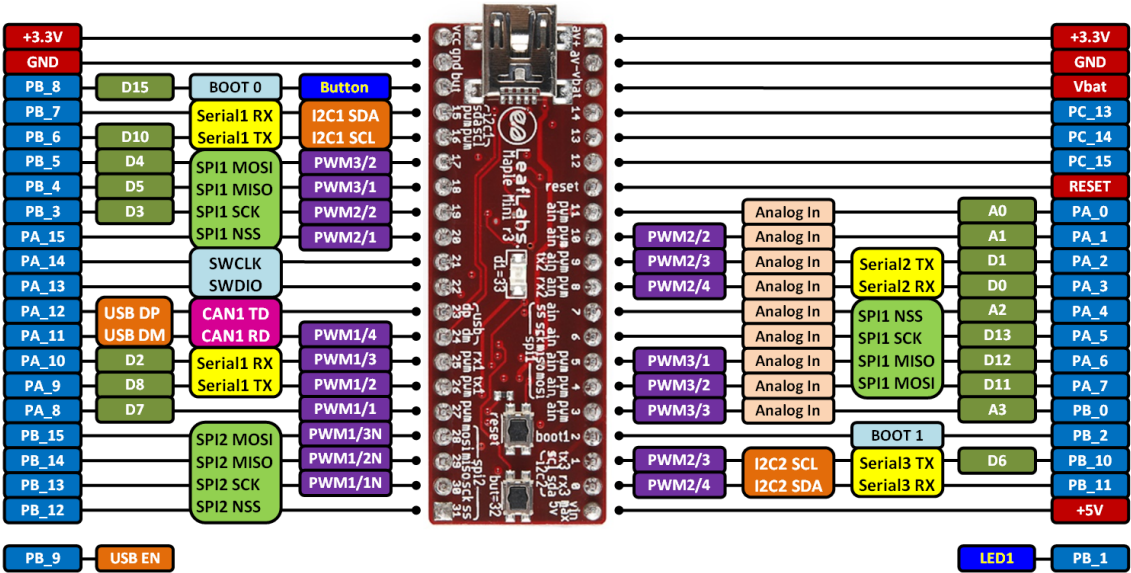
 [Build repository \(/u:](#)

 [Embed u](#)

 [Clone repository :](#)

## Repository details

Type:	 Program
Created:	17 Jul 2016 (17 Jul
Imports:	 87 (/users/hudal
Forks:	 0 (/users/hudakz
Commits:	 6 (shortlog)
Dependents:	 0 (/users/hudakz/coc
Dependencies:	 2 (/users/hudakz/coc
Followers:	 5 (/users/hudakz



Zoom in ([/media/uploads/hudakz/maplemini\\_pinout01.png](/media/uploads/hudakz/maplemini_pinout01.png))

Information

Only the labels printed in **blue/white** or **green/white** (i.e. PB\_8, PC\_13, D15, A0 ...) must be used in your code. The other labels are given as information (alternate-functions, power pins, ...). You can also use these additional pin names:

```
1 LED1=PB_1    SERIAL_TX=PA_2    I2C_SCL=PB_6    SPI_MOSI=PA_7    PWM_OUT=PB_3
2              SERIAL_RX=PA_3    I2C_SDA=PB_7    SPI_MISO=PA_6
3              SPI_SCK =PA_5
4              SPI_CS  =PB_6
```

## Schematic

The schematic is available [here \(http://cdn.sparkfun.com/datasheets/Dev/ARM/maplemini.pdf\)](http://cdn.sparkfun.com/datasheets/Dev/ARM/maplemini.pdf).

## Using the mbed online compiler to build programs for Maple Mini

- Create a program as if it was for a NUCLEO-F103RB board (select NUCLEO-F103RB as target platform in the online compiler).  
Or click [here \(https://developer.mbed.org/compiler/#import:/users/hudakz/code/MapleMini\\_Hello/\)](https://developer.mbed.org/compiler/#import:/users/hudakz/code/MapleMini_Hello/) to import this demo into your online compiler (then you can skip the following two steps).
- Import the [mbed-MapleMini \(https://developer.mbed.org/users/hudakz/code/mbed-MapleMini/\)](https://developer.mbed.org/users/hudakz/code/mbed-MapleMini/) library into your project.
- Add `#include "MapleMini.h"` to main.cpp before `#include "mbed.h"` (the position matters!).

Blinking on-board LED:

```
1 #include "MapleMini.h"
2 #include "mbed.h"
3
4 int main() {
5     confSysClock();    //Configure system clock (72MHz HSE clock, 48MHz USB clock)
6
7     Serial    pc(PA_2, PA_3);
8     DigitalOut myled(LED1);
9
10    while(1) {
11        myled = 1;    // turn the on-board LED on
12        wait_ms(200); // wait 200 milliseconds
13        myled = 0;    // turn the on-board LED off
14        wait_ms(1000); // wait 1000 milliseconds
15        pc.printf("Blink\r\n");
16    }
17 }
```

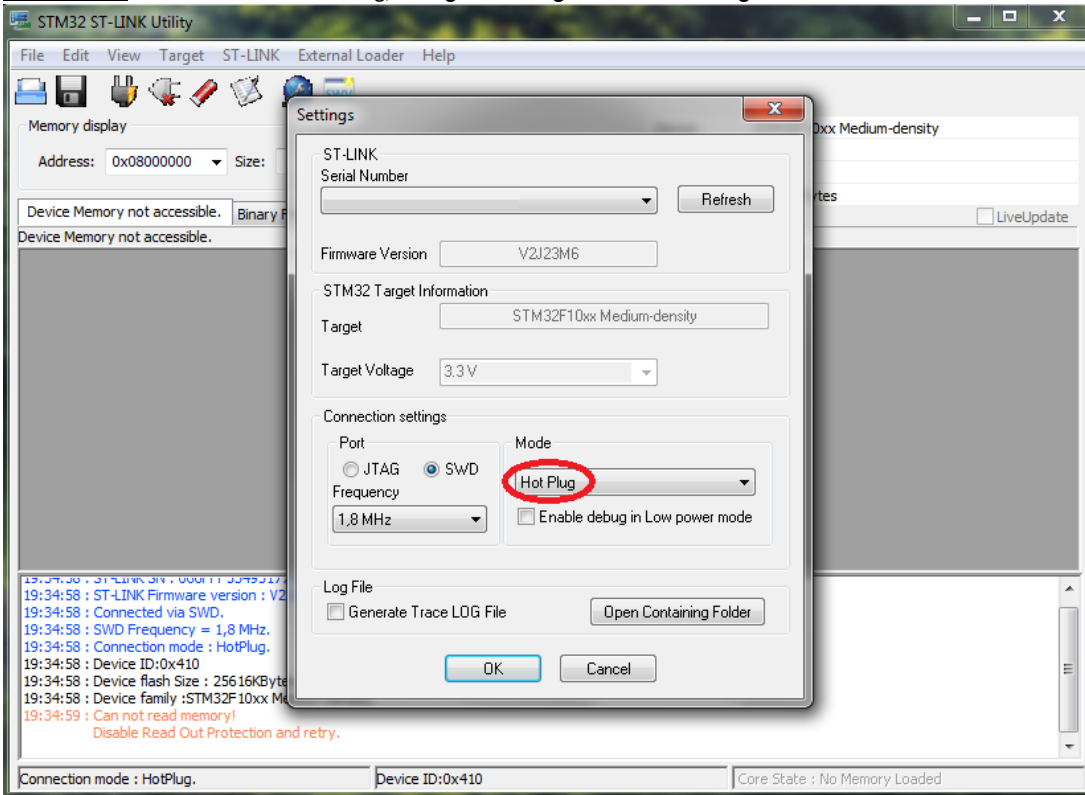


NRST

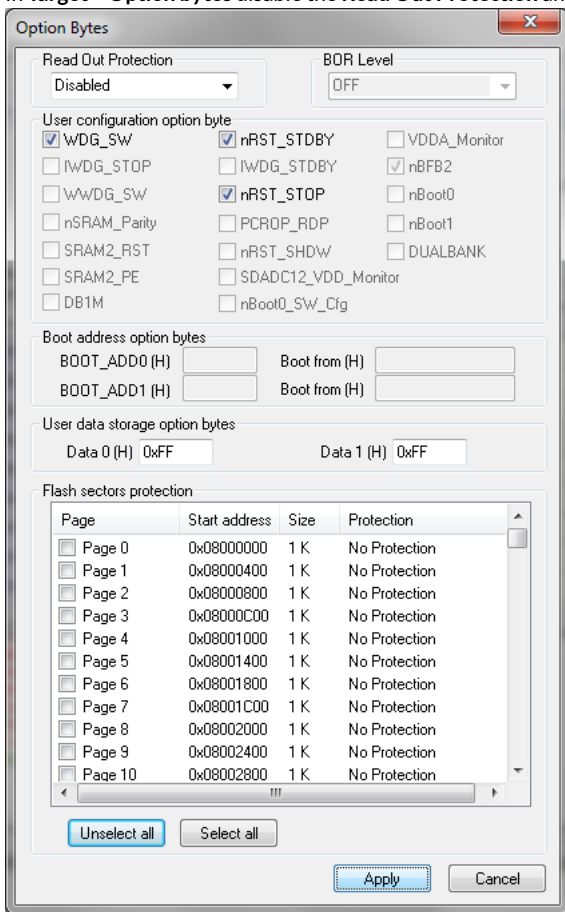
&lt;=&gt;

RESET

- Provide power for the Maple Mini board.
- Connect the NUCLEO board to your PC using a USB cable.
- Maple's flash memory is protected. In order to start downloading programs to the board the protection has to be removed. That could be accomplished with the **STM32 ST-LINK Utility** ([http://www.st.com/content/st\\_com/en/products/embedded-software/development-tool-software/stsw-link004.html](http://www.st.com/content/st_com/en/products/embedded-software/development-tool-software/stsw-link004.html)). Once installed and running, in **Target->Settings** switch to **Hot Plug mode**.



- In **Target->Option bytes** disable the **Read Out Protection** and turn the **Flash sectors protection** off as well.



- After that you'll be able to fully erase the chip (including the LeafLabs Maple boot loader).

#### ⚠ Warning

Once the chip has been erased the board cannot be used with LeafLabs Maple IDE anymore!

- Exit the STM32 ST-LINK Utility. From now on you'll be able to download programs to the board.
- To program the Maple Mini board, click on the **Compile** button and save the binary to the NUCLEO virtual disk .

For more details have a look at the [User Manual](#)

([http://www.st.com/content/ccc/resource/technical/document/user\\_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/tra:chapter 6.2.4 Using ST-LINK/V2-1 to program and debug an external STM32 application](http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/tra:chapter 6.2.4 Using ST-LINK/V2-1 to program and debug an external STM32 application)).

## Using serial port (not just for debugging)

- Connect an FTDI or similar USB-Serial converter to your PC and to an on-board serial port (for example PA\_2, PA\_3). Make sure you connect the on-board TX pin to the converter's RX pin and the on-board RX pin to the converter's TX pin.
- In your code, create a `Serial` object (using TX and RX pin names of the connected serial port).
- Use `printf` function to send serial messages to the connected PC.

#### 📄 Sending debug messages over the ST-Link virtual com port

In case you would like to spare the external USB-Serial converter for other purposes then there is available an alternative solution proposed by [X.M \(bitman\)](https://developer.mbed.org/users/bitman/) (<https://developer.mbed.org/users/bitman/>). You can use the ST-Link virtual com port also for debugging of programs running on the Maple Mini board. However, that will require a soldering iron (and probably some soldering skills). According to the [User Manual](#) ([http://www.st.com/content/ccc/resource/technical/document/user\\_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/tra:chapter 6.8 USART communication](http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/tra:chapter 6.8 USART communication)), solder bridges (on the back side of the NUCLEO board) SB62 and SB63 should be

ON, SB13 and SB14 should be OFF. In such case it is possible to connect another USART to the NUCLEO (ST-Link) CN3 connector using flying wires. For instance on STM32F103C8T6 board it is possible to use USART2 available on PA\_2 (TX) and PA\_3 (RX). Two flying wires shall be connected as follows:

Maple Mini board, pin PA\_2 (Serial2 TX)    <=>    NUCLEO board CN3 connector, pin RX

Maple Mini board, pin PA\_3 (Serial2 RX)    <=>    NUCLEO board CN3 connector, pin TX

A smart trick proposed by **Nothing Special** (<https://developer.mbed.org/users/mega64/>) makes even soldering needless.

The point is to redirect the UART on the NUCLEO board by software (without modifying the solder bridges on the back side of the NUCLEO board) and convert it into a "Debugger". On the NUCLEO board that you are going to use as programmer/ debugger, choose any Serial port other than Serial2 (other than the default port used for standard UART) to be initialized as standard UART. In the program below (using NUCLEO-F103RB as programmer/debugger) Serial1 (PA\_9, PA\_10) was selected.

#### Debugger

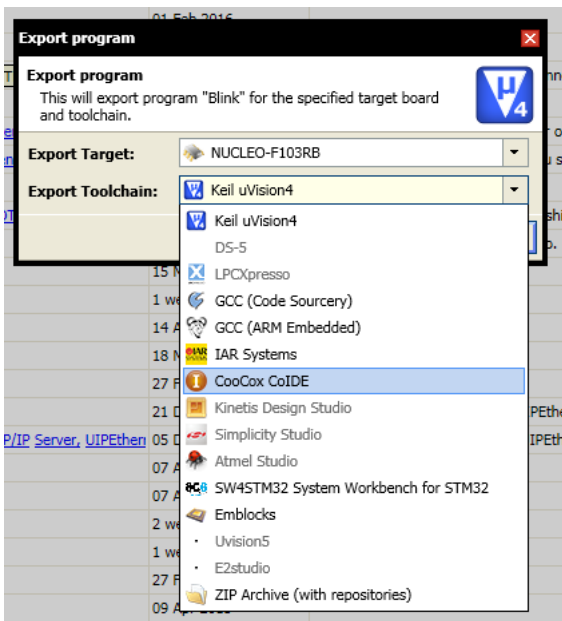
```
1 #include "mbed.h"
2
3 // declarations needed to change the parameters of stdio UART
4 extern serial_t      stdio_uart;
5 extern int           stdio_uart_initied;
6
7 int main() {
8     serial_init(&stdio_uart, PA_9, PA_10); // other than Serial2
9     stdio_uart_initied = 1;
10    printf("Ready for debugging\r\n");
11 }
```

Once compiled (remember to select the NUCLEO board used for programing/debugging as target for the online compiler), download the "Debugger" program to the NUCLEO board. Please make sure you have the two jumpers in place on the CN2 connector when programming the NUCLEO board. Once downloaded to the NUCLEO board, remove the two jumpers again.

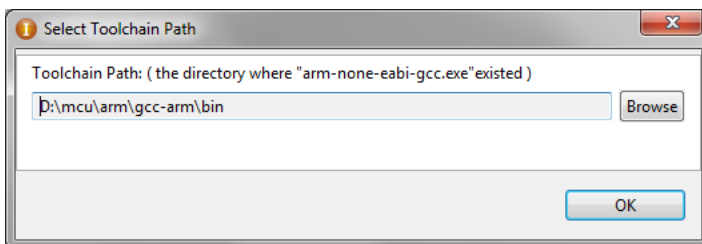
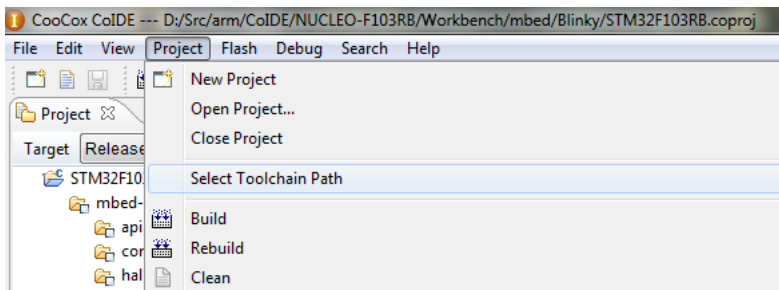
## Offline compilation and debugging with CooCox CoIDE (<http://www.coocox.org/>)

NOTE: The pictures below are just for illustration (some belong to the STM32F103C8T6 - Blue Pill ([https://developer.mbed.org/users/hudakz/code/STM32F103C8T6\\_Hello/](https://developer.mbed.org/users/hudakz/code/STM32F103C8T6_Hello/)) project).

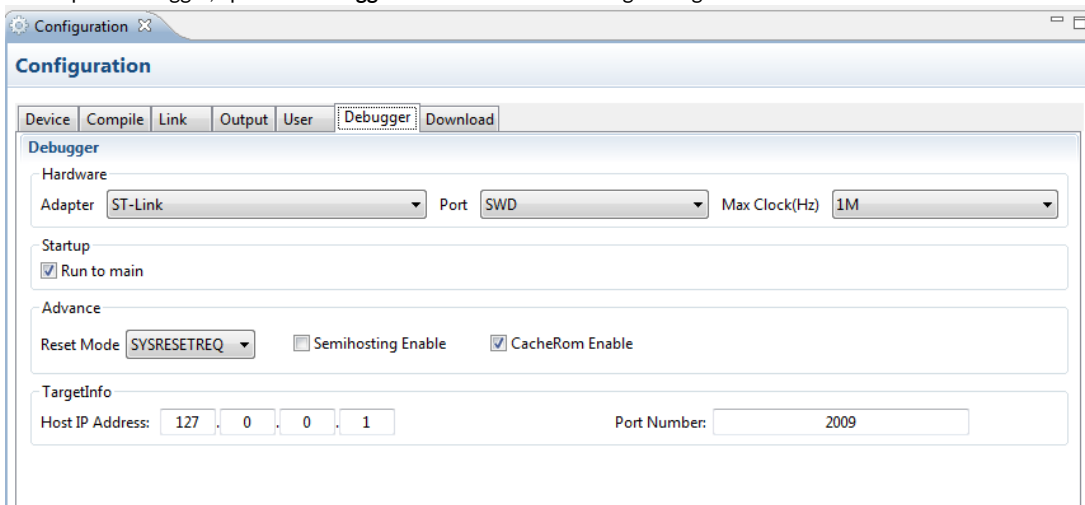
- I do not recommend to use the Beta version but rather install CoIDE V1.7.8 (<http://www.coocox.org/download/Tools/CoIDE-1.7.8.exe>).
- If not done yet then download and install the GNU Arm Embedded Toolchain (<https://developer.arm.com/open-source/gnu-toolchain/gnu-rm>) onto your PC.
- Keep the NUCLEO board ( ST-LINK) connected to the Maple Mini board as in the set-up for online programming.
- Export your program from the **mbed online compiler** to **CooCox CoIDE**.



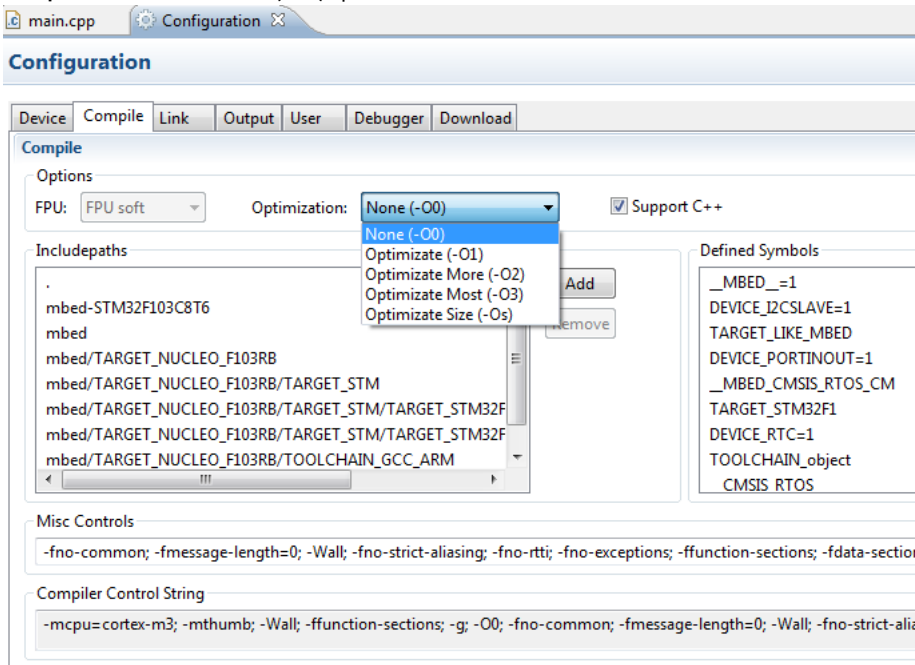
- Save the file into a folder on you PC's hard drive and unzip.
- Run **CoIDE** and open the project.
- On the menu bar click on **Project**. Choose **Select Toolchain Path** and select the path to the **GNU Arm Embedded Toolchain**. Once completed the path will apply to all projects. (It is not necessary to repeat this for each project but rather do it only once after installing the toolchain.)



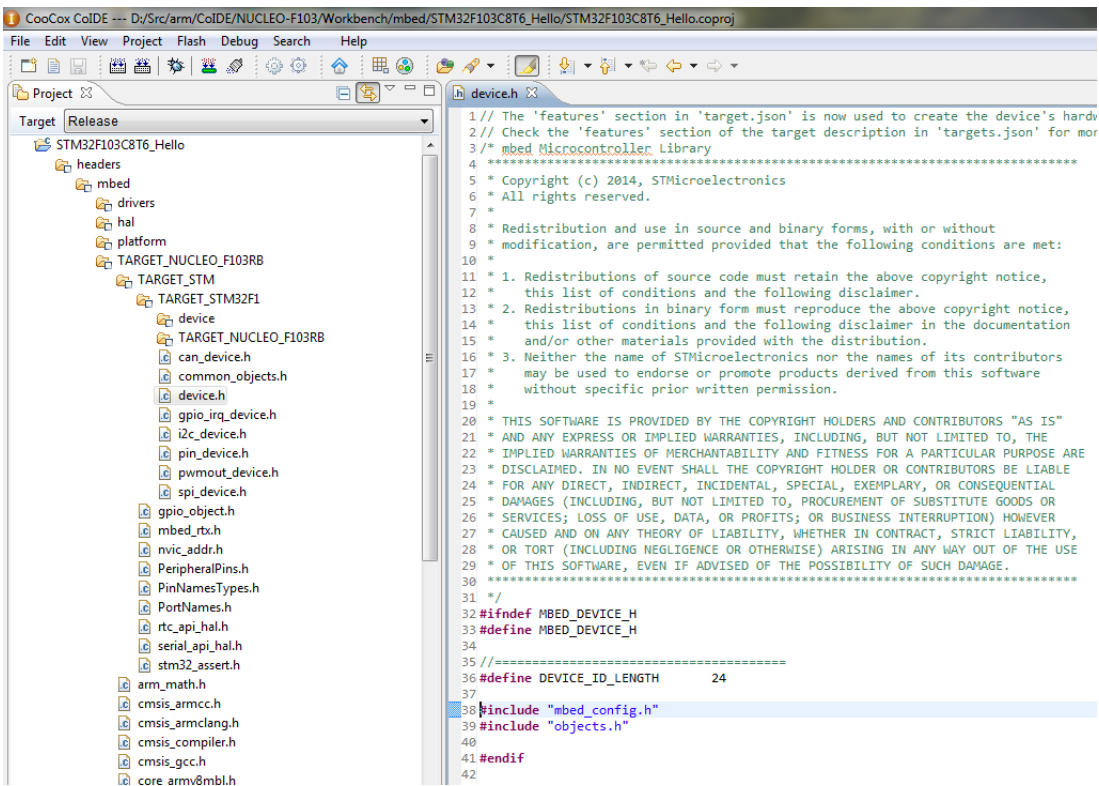
- To setup the debugger, open the **Debugger** tab and use the following settings:



- To ensure that the execution of instructions during debugging is performed in the foreseen sequence, go to the **Compile** tab and select **None (-O0)** optimization.



- To be able to use the `Serial` class in application programs built offline, in the **Project** side bar navigate to `headers/mbd/TARGET_NUCLEO_F103RB/TARGET_STM/TARGET_STM32F1` and open the `device.h` header file for editing by double-clicking on it. Then add a new line saying `#include "mbd_config.h"` as below:



After such modification the program should compile with no errors.

- Another option is to use the default serial port which is automatically created by the mbed library. In such case no `Serial` object (like `Serial pc(PA_2, PA_3);`) has to be created. Instead the global `printf` function is called as below. In this case also the binary code becomes smaller. The default settings are 9600 bits/s, one start bit, eight data bits, no parity bit, one stop bit.

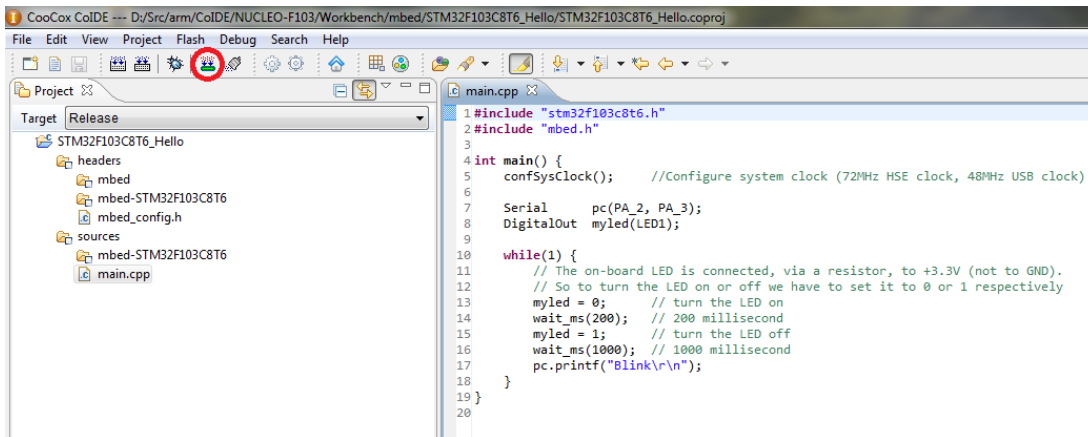


```

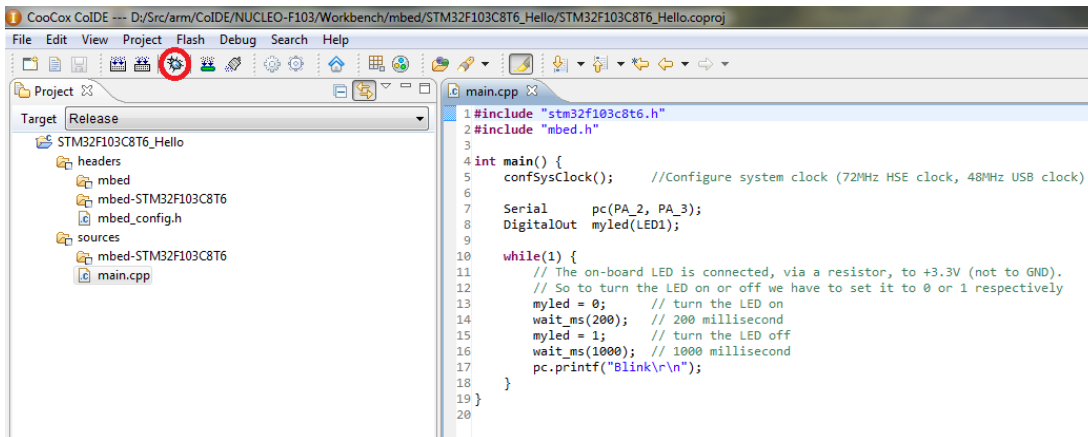
1 #include "MapleMini.h"
2 #include "mbed.h"
3
4 int main() {
5     confSysClock();    //Configure system clock (72MHz HSE clock, 48MHz USB clock)
6
7     //Serial    pc(PA_2, PA_3);
8     DigitalOut myled(LED1);
9
10    while(1) {
11        myled = 1;      // turn the on-board LED on
12        wait_ms(200);   // wait 200 milliseconds
13        myled = 0;      // turn the on-board LED off
14        wait_ms(1000);  // wait 1000 milliseconds
15        //pc.printf("Blink\r\n");
16        printf("Blink\r\n");
17    }
18 }

```

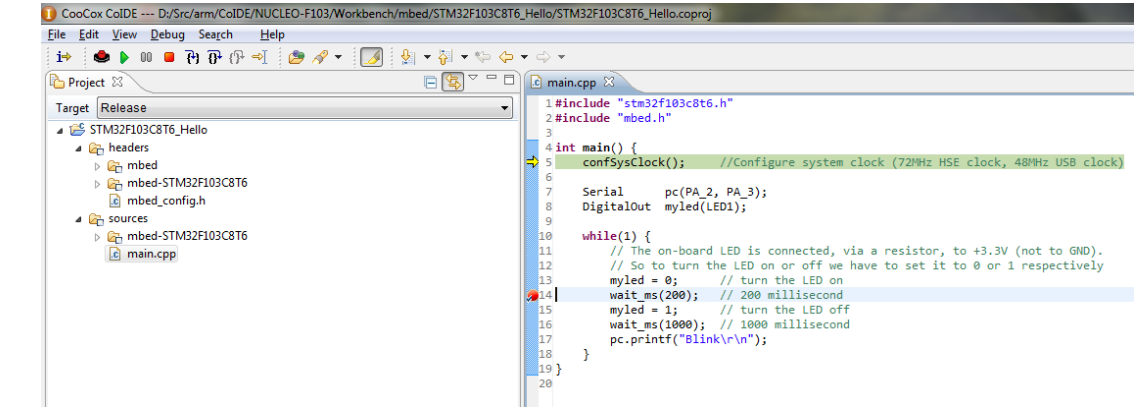
- Open **main.cpp** for editing and rebuild the project by clicking on the **Rebuild** button.
- Reprogram the board by clicking on the **Download Code To Flash** button:  
NOTE: Any time you launch the debugger the program is getting recompiled and the board reprogrammed. So you can skip this step. I included it just to illustrate how easily you can (re)program the board offline.



- Set up some debug points and launch the debugger by clicking on the **Start Debug** button:



- You can execute the instructions step by step (or run to the next debug point) and observe the variables and outputs whether they are changing as expected...



Happy coding and debugging :-)

Download repository: [zip \(/users/hudakz/code/MapleMini\\_Hello/archive/a82043222b61.zip\)](zip://users/hudakz/code/MapleMini_Hello/archive/a82043222b61.zip) [tar.gz \(/users/hudakz/code/MapleMini\\_Hello/archive/a82043222b61.tar.gz\)](tar.gz://users/hudakz/code/MapleMini_Hello/archive/a82043222b61.tar.gz)

## Files at revision 5:a82043222b61

default

tip

Name	Size	Actions
<div> <div> <div></div> <div>lupl (/users/hudakz/code/MapleMini_Hello/file/a82043222b61/)</div> </div> </div>		
<div> <div> <div></div> <div>main.cpp</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/file/a82043222b61/main.cpp)</div> </div> </div> </div>	489	<div> <div> <div></div> <div>Revisions</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/log/a82043222b61/main.cpp)</div> </div> <div> <div></div> <div>Annotate</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/annotate/a82043222b61/main.cpp)</div> </div> </div> </div> </div>
<div> <div> <div></div> <div>mbed-MapleMini.lib</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/file/a82043222b61/mbed-MapleMini.lib)</div> </div> </div> </div>	74	<div> <div> <div></div> <div>Revisions</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/log/a82043222b61/mbed-MapleMini.lib)</div> </div> <div> <div></div> <div>Annotate</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/annotate/a82043222b61/mbed-MapleMini.lib)</div> </div> </div> </div> </div>
<div> <div> <div></div> <div>mbed.bld</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/file/a82043222b61/mbed.bld)</div> </div> </div> </div>	66	<div> <div> <div></div> <div>Revisions</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/log/a82043222b61/mbed.bld)</div> </div> <div> <div></div> <div>Annotate</div> </div> <div> <div> <div></div> <div>(/users/hudakz/code/MapleMini_Hello/annotate/a82043222b61/mbed.bld)</div> </div> </div> </div> </div>