Homepage (/users/hudakz/code/Blackpill_Hello/wiki/Homepage)

Using Blackpill (STM32F401CCU6 or STM32F411CEU6) boards with mbed

It provides an affordable (about $3 / $6 on eBay) and flexible way for users to try out new ideas and build prototypes. The board is equipped with an STM32F401CCU6 (https://www.st.com/resource/en/datasheet/stm32f401cc.pdf) or an STM32F411CEU6 (https://www.google.com/url?

sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjn4_eQ8v7rAhUHAcAKHQVmBRQQFjABegQIBxAC&url=https%3A%2F%2Fwww.st.com%2Fresourc microcontroller compatible with the NUCLEO-F401CE (https://os.mbed.com/platforms/ST-Nucleo-F401RE) or the NUCLEO-F411RE (https://developer.mbed.org/platforms/ST-Nucleo-F411RE/) platform, respectively.

Microcontroller features

- STM32F401CCU6 or STM32F411CEU6 in UFQFPN48 package
- ARM® 32-bit Cortex® -M4 CPU with FPU
- 100 MHz max CPU frequency
- VDD from 1.7 V to 3.6 V
- 256 or 512 KB Flash
- 64KB or 128 KB SRAM
- GPIO with external interrupt capability
- 1x12-bit, 2.4 MSPS ADC with 16 channels
- DMA Controller
- Up to 11 Timers (six 16-bit, two 32-bit, two watchdog timers and a SysTick timer)
- USART/UART (3)
- I2C (3)

- SPI/I2S (3 or 5)
- SDIO
- USB 2.0 full-speed device/host/OTG controller with on-chip PHY
- CRC calculation unit
- 96-bit unique ID
- RTC

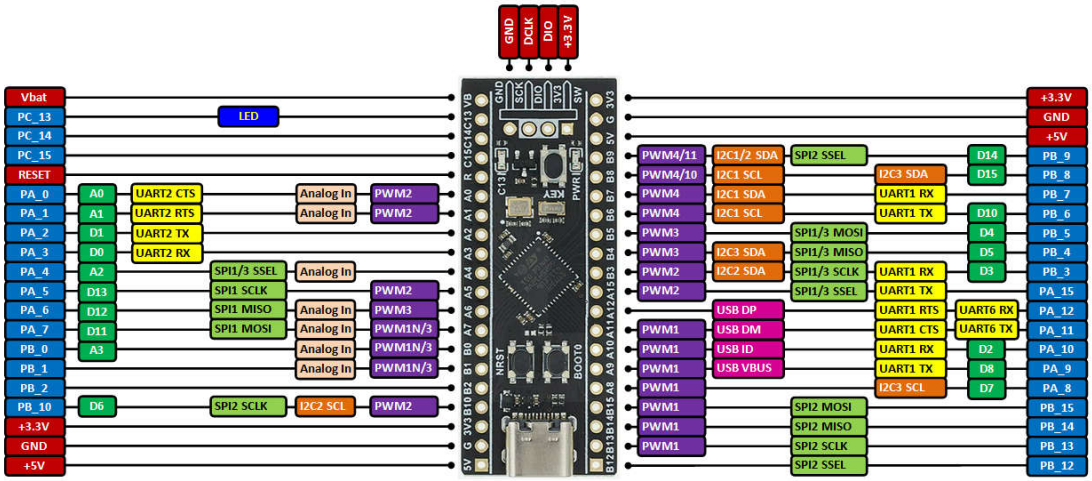
Board features

- Small foot-print
- Flexible board power supply: USB VBUS or external source (3.3V, 5V)
- User LED: LED1
- Three push buttons: RESET, BOOT0, KEY
- Programming/Debug port
- USB-C connector

Board pinout

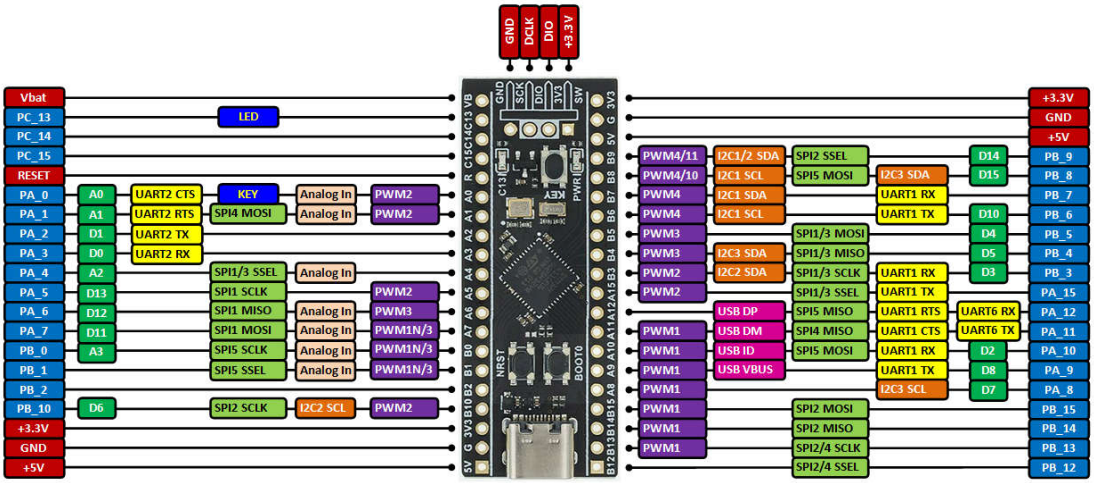
When equipped with STM32F401CCU6:

Zoom in (https://os.mbed.com/media/uploads/hudakz/blackpill_f401cc.png)



When equipped with STM32F411CEU6:

Zoom in (https://os.mbed.com/media/uploads/hudakz/blackpill-pinout.png)



For more details on pin definitions see Table 8 in the Datasheet (https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjn4_eQ8v7rAhUHAcAKHQVmBRQQFjABegQIBxAC&url=https%3A%2F%2Fwww.st.com%2Fresourc

Information

Only the labels printed in **blue/white** or **green/white** (i.e. PC_13, PB_9, A0, D14 ...) 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: Please notice that in order to fit the small size board, the leading 'P' and the '_' characters are omitted from

labels indicated on the board (e.g. Instead of 'PA_1' you can find the label 'A1' on the board). Arduino (green/white) and the additional naming labels are not indicated on the board.
Also notice that the on-board LED is connected to pin PC_13 and, via a resistor, to +3.3V. So to turn the LED `on` or `off` you have to set the DigitalOut to `0` or `1` respectively (active on 0).

Using the mbed online compiler to build programs for the Blackpill board

Create a program as if it was for a NUCLEO-F401RE or NUCLEO-F411RE board (select NUCLEO-F401RE or NUCLEO-F411RE as target platform for the online compiler).
Or click here (https://developer.mbed.org/compiler/#import:/users/hudakz/code/Blackpill_Hello/) to import this demo into your online compiler.

Blinking on-board LED:

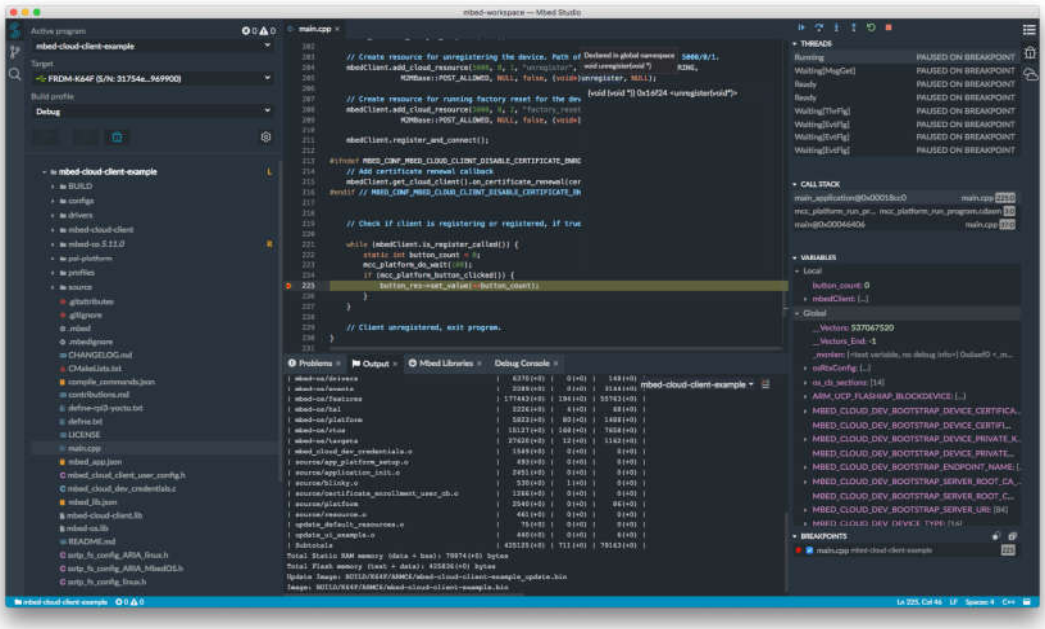
```
#include "mbed.h"

DigitalOut led(PC_13);

int main()
{
    while (true) {
        led = !led;
        ThisThread::sleep_for(500ms);
    }
}
```

Building programs with Mbed Studio IDE

The Mbed Studio IDE (https://os.mbed.com/docs/mbed-studio/current/introduction/index.html) is a great tool for working offline.



- It supports Mbed OS 6 (with built-in Mbed RTOS threads, mutexes, semafores ...)
- and also Mbed OS 6 Bare metal profile (https://os.mbed.com/docs/mbed-os/v6.3/bare-metal/index.html) (no RTOS, similar to Mbed OS 2) which is suitable for small or medium size projects requiring less RAM and Flash. To use it, add a `mbed_app.json` configuration file to your project:

med_app.json

```
{
    "requires": ["bare-metal"]
}
```

As target you can select

- either the NUCLEO_F401RE (https://os.mbed.com/platforms/ST-Nucleo-F401RE) or the NUCLEO_F411RE (https://os.mbed.com/platforms/ST-Nucleo-F411RE)
- or the BLACKPILL (https://os.mbed.com/users/hudakz/code/BLACKPILL_Custom_Target) custom target.

Advantages of the BLACKPILL custom target over the NUCLEO_F401RE/ NUCLEO_F411RE

- The **onboard external 25 MHz crystal** is used as system clock source rather than the less precise internal 16 MHz RC oscillator.
- The **onboard LED** works as **LED1** in programs.
- The **onboard KEY** on STM32F411CEU6 boards works as **USER_BUTTON** pin in programs.
- You can use the **USB peripheral** in your programs and connect the board to the PC **over the onboard USB connector**. An example of using the USB peripheral as USBSerial (12 Mbit/s) is available here. (https://os.mbed.com/users/hudakz/code/Blackpill_USBSerial/)

Building programs for the BLACKPILL custom target in Mbed Studio

- Connect an STM32 ST-Link programmer to your BLACKPILL board and PC (see below for more details).
- Create a new program in the Mbed Studio IDE.
- Right-click on the program's root folder and in the popup window select `Add library...`
- Copy&paste the text **https://os.mbed.com/users/hudakz/code/BLACKPILL_Custom_Target (https://os.mbed.com/users/hudakz/code/BLACKPILL_Custom_Target)** into the `Git or os.mbed.com URL` edit box and click on the `Next` button.
- Open the drop-list and select `default` as `Branch or tag` and click on the `Finish` button.
- Open the `BLACKPILL_Custom_Target` folder and according to you board drag&drop the `TARGET_BLACKPILL_F401CC` or the `TARGET_BLACKPILL_F411CE` folder and the `custom_targets.json` file one by one to the root folder of your program.
- Delete the `BLACKPILL_Custom_Target` folder from your project. (Right-click and select delete).
- Open the `Target` drop-list and click on the button with a "chip" icon on it (Manage custom targets) .
- Open the `USB device` drop-list and select your STM32 ST-Link programmer.
- Open the `Build target` drop-list and according to your board select `BLACKPILL_F401CC` or `BLACKPILL_F411CE` .
- Click on the `Save All` button.
- Build your program (click on hammer button).

Programming the Blackpill board

NUCLEO ST-LINK/V2-1 and drag & drop

You can use the NUCLEO virtual disk to program the Blackpii board (drag and drop programming). To do that, an additional NUCLEO board is needed (any type equipped with ST-LINK/V2-1 will do).

- Remove the two jumpers from the CN2 connector as illustrated in Figure 8:

Figure 8. Using ST-LINK/V2-1 to program the STM32 on an external application

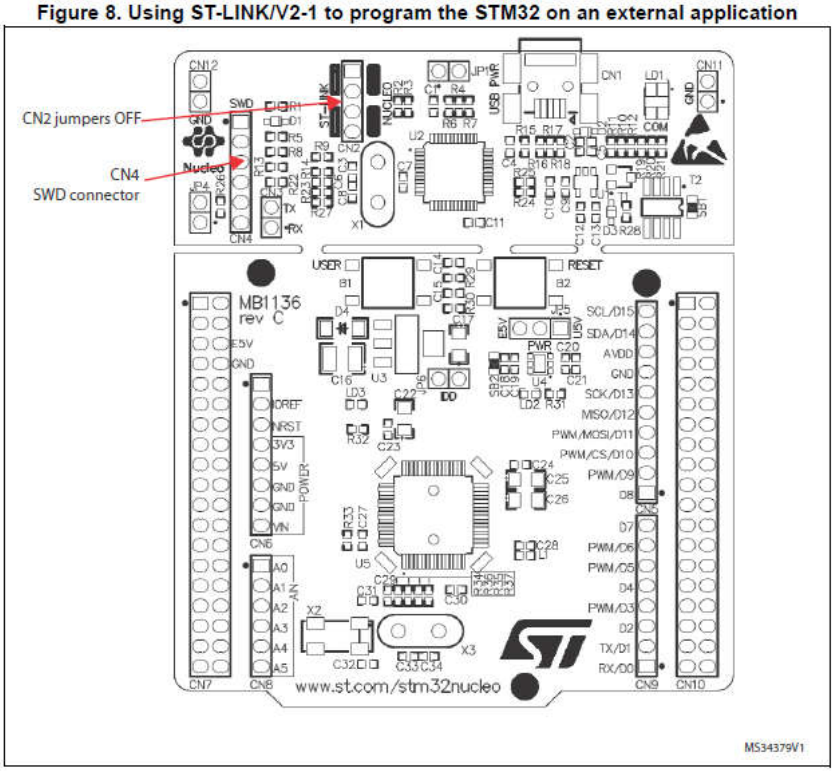
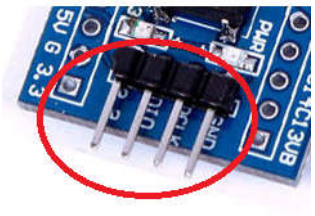


Table 4. Debug connector CN4 (SWD)

Pin	CN4	Designation
1	VDD_TARGET	VDD from application
2	SWCLK	SWD clock
3	GND	Ground
4	SWDIO	SWD data input/output
5	NRST	RESET of target MCU
6	SWO	Reserved

STM32F103C8T6 debug connector



3V3, DIO, DCLK, GND

- Connect the NUCLEO board CN4 connector to the Blackpill board using flying wires as follows:

NUCLEO board CN4 connector		Blackpill debug connector		Blackpill board
SWCLK	<=>	DCLK		
GND	<=>	GND		
SWDIO	<=>	DIO		
NRST			<=>	RESET

> Warning
>
> Please notice that `VDD_TARGET` is not connected. That works with the ST-Link programmer but could potentially damage the target micro controller in case it's running at a lower voltage (e.g. 2V5) than the programmer (e.g. 3V3). That's why it's recommended to connect also the `VDD_TARGET` line when an external programmer such as a Segger J-Link is hooked up to program the board.

- Provide power for the Blackpill board through a 3.3V pin, 5V pin or over a USB cable. (The VDD_TARGET pin on the NUCLEO board CON4 does not work as source of power).

- Connect the NUCLEO board to your PC over a USB cable.

- To program the Blackpill 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:conten

chapter 6.2.4 *Using ST-LINK/V2-1 to program and debug an external STM32 application.*

ST-Link V2 USB dongle and STM32 ST-LINK utility

If you would like to use an ST-Link V2 USB dongle (aka ST-Link V2 Programming Unit) to program the board apply the same wiring as specified above. If not done yet install an ST-Link/V2 driver (https://os.mbed.com/teams/ST/wiki/ST-Link-Driver) onto your PC. Plug the ST-Link V2 dongle into your PC. Then click on the `Compile` button and save the binary to your local disk. Install and run the STM32 ST-LINK utility (http://www.st.com/en/development-tools/stsw-link004.html). Once the program is running open the binary built with the online compiler and click on the `Program verify` button.

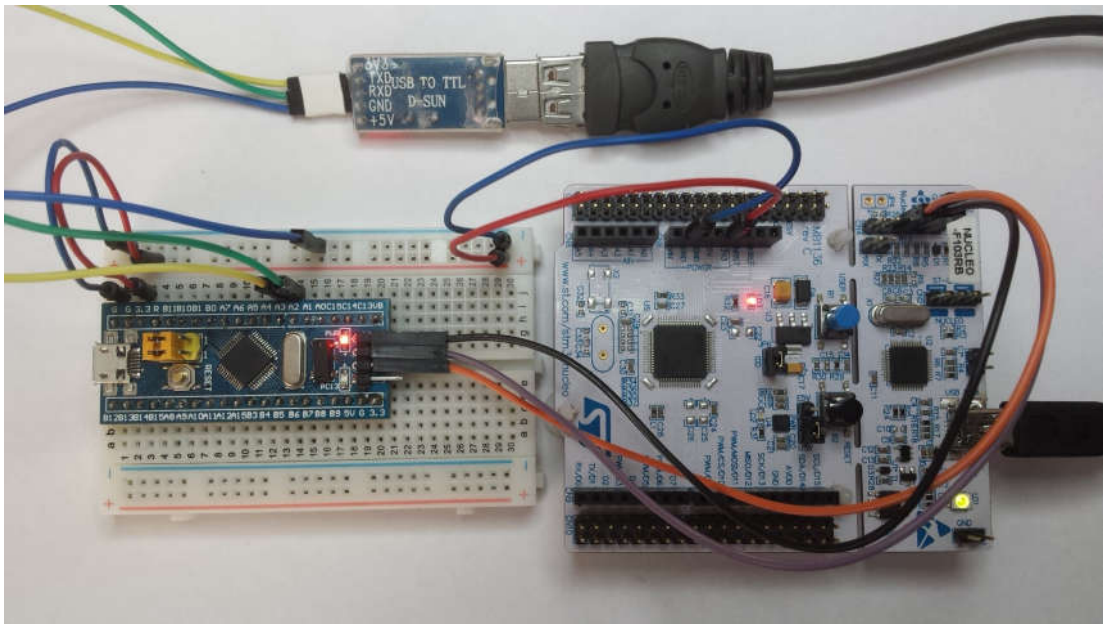
STM32 USART system memory bootloader and Flasher-STM32

Have a look at STM32 Embedded Bootloader (https://scienceprog.com/flashing-programs-to-stm32-embedded-bootloader/).

Download: FLASHER-STM32 (http://www.st.com/en/development-tools/flasher-stm32.html)
For more details read: Appliction Note AN2606 (http://www.st.com/content/ccc/resource/technical/document/application_note/b9/9b/16/3a/12/1e/40/0c/CD00167594.pdf/files/CD00167594.pdf/jcr:content

Using serial port (not just for debugging)

- Connect an FTDI or similar USB to Serial TTL 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.



Zoom in (/media/uploads/hudakz/stm32f103c8t6_hookup.jpg)

- 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/). You can use the ST-Link virtual com port also for debugging of programs running on the Blackpill 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/DM0010582 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 Blackpill board it is possible to use USART2 available on PA_2 (TX) and PA_3 (RX). Two flying wires shall be connected as follows:

Blackpill board, pin PA_2 (Serial2 TX)	<=>	NUCLEO board CN3 connector, pin RX
Blackpill 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

```
#include "mbed.h"

// declarations needed to change the parameters of stdio UART
extern serial_t     stdio_uart;
extern int          stdio_uart_inited;

int main() {
    serial_init(&stdio_uart, PA_9, PA_10); // other than Serial2
    stdio_uart_inited = 1;
    printf("Ready for debugging\r\n");
}
```

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 the "Debugger" binary has been downloaded to the NUCLEO board, remove the two jumpers again.

Happy coding :-)